LEARNING DISCRETE REPRESENTATIONS VIA MUTUALLY EXCLUSIVE RANDOM VARIABLES

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel approach, named Mutually Exclusive Random Variable Autoencoder (MERVar-AE), to learn discrete representations based on Mutually Exclusive Random Variables (MERVar). Mutually Exclusive Random Variable (MERVar) arises from the observation that deterministic mappings (e.g., neural decoders) involving random variables (noise injection) tend to reduce overlap coefficients among involved random variables, thereby inducing exclusivity. This exclusivity can be utilized to learn binary latents. Specifically, by injecting symmetric noise into the output of a bounded activation function (e.g., $\tanh$), the output values are pushed toward the boundaries (-1 or 1 for $\tanh$), effectively producing binary latent representations. Compared to previous approaches based on Straight-Through Estimator (STE), such as VQ-VAE, the proposed approach is simpler, with only an activation and noise injection operation. The forward and backward computations are aligned, as gradients are propagated through the same continuous latent representation. Comprehensive experiments across nine standard datasets demonstrate competitive reconstruction performance against representative state-of-the-art quantization techniques under a unified experimental protocol. Controlled experiments further demonstrate the benefit of avoiding surrogate-gradient mismatch during encoder optimization.

1 INTRODUCTION

Learning discrete representations is an important task in generative autoregressive models, with VQ-VAE (van den Oord et al., 2017) being one of the most popular methods. VQ-VAE employs a discrete codebook to map encoder outputs to embeddings before they are fed into the decoder, with the Straight-Through Estimator (STE) being utilized to bypass the non-differentiable quantization step. Although there is a mismatch between the forward and backward computations, VQ-VAE remains one of the most popular methods for discrete representation learning due to its elegant simplicity and efficiency. In this work, we propose a simpler alternative based on injecting symmetric noise into the bounded activation outputs (e.g., $\tanh$) of the latent space. During training, we observe that the outputs of the activation function gradually become bipolar, with values being driven toward the boundaries of the activation range. Interestingly, after adding noise, these latent variables behave as if they are mutually exclusive. We investigated and named this phenomenon the Mutually Exclusive Random Variable (MERVar), and the autoencoder utilizing MERVar to learn binary latents is named the Mutually Exclusive Random Variable Autoencoder (MERVar-AE).

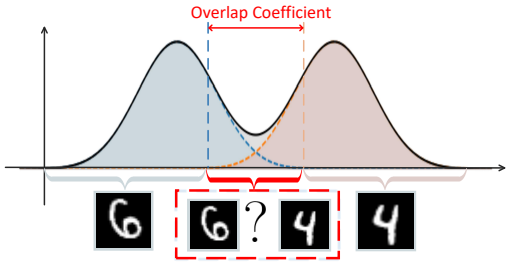

Figure 1: Selecting images for values within the overlapping range is ambiguous. This ambiguity is incompatible with a deterministic mapping system (e.g., a neural decoder), and is empirically observed to decrease as the random variables move apart during optimization.

Mutually Exclusive Random Variable (MERVar) arises from the operation of adding symmetric noise ϵ_i (e.g., Gaussian noise) into the latent $\mathbf{z}_i$ with respect to an image $\mathbf{x}_i$ in an autoencoder: $\tilde{\mathbf{z}}_i = \mathbf{z}_i + \sigma \cdot \epsilon_i$. This operation extends the variable $\mathbf{z}_i$ into a random variable $\tilde{\mathbf{z}}_i$ whose values vary in different training updates. During training, the decoder tries to map a random variable $\tilde{\mathbf{z}}_i$ to a

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.

deterministic variable $\mathbf{x}_i$ (an image). Therefore, it is possible that two random variables $\tilde{\mathbf{z}}_i$ and $\tilde{\mathbf{z}}_j$ produce similar or the same realizations that are used to reconstruct different images as shown in Fig. 1. Unfortunately, most decoders are deterministic mapping systems, and a deterministic mapping system is incompatible with cases where two similar or identical values are mapped to different images. Therefore, achieving low reconstruction error requires reducing such ambiguity. Empirically, we observe that autoencoder optimization reduces this ambiguity by pushing the values of $\mathbf{z}_i$ and $\mathbf{z}_j$ away from each other. Based on this phenomenon, the MERVar Autoencoder is proposed for effective discrete representation learning.

The proposed MERVar Autoencoder is straightforward, and the MERVar theory behind it provides a principled foundation. We first define MERVar in Definition 3.1 and mathematically establish its exclusivity in Theorem 3.3 and Theorem 3.5. Furthermore, comprehensive validation experiments are conducted to support our theoretical claims. For MERVar-AE, simplicity is a core advantage, and most autoencoder architectures can be extended into a MERVar-AE by passing the encoder outputs into a bounded activation function (e.g., $\tanh$) and adding symmetric noise (e.g., Gaussian). A MERVar-AE based on the $\tanh$ function and Gaussian noise ϵ_i is formulated as:

$$\text{MERVar-AE}(\mathbf{X};\theta,\phi) = \sum_{\mathbf{x}_i \in \mathbf{X}} \mathbb{E}_{\epsilon_i}\left[\|d_\phi\Big(\tanh\left(e_\theta(\mathbf{x}_i)\right) + \sigma \cdot \epsilon_i\Big) - \mathbf{x}_i\|^2\right], \quad \epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (1)$$

where $\mathbf{z}_i = \tanh(e_\theta(\mathbf{x}_i))$ is the latent representation that gradually binarizes during training. e_θ and d_ϕ denote the encoder and decoder with parameters θ and ϕ, respectively. $\mathbf{x}_i$ denotes the training images, and σ is used to control the noise intensity. As illustrated above, the forward and backward computations are aligned, avoiding the mismatch introduced by the Straight-Through Estimator (STE). During training, the latent values gradually converge toward ± 1. Once most latent values are sufficiently close to the bounds, sign-based binarization is applied and the noise is removed. The decoder is then optimized for a few additional updates, while the encoder remains fixed since gradients are not backpropagated through the binarization operation. The main contributions are:

- We introduce the Mutually Exclusive Random Variable Autoencoder (MERVar-AE). By injecting noise into the outputs of bounded activation functions, the model learns signed binary latents.Compared to previous methods, the proposed approach is simpler, achieves strong reconstruction performance, and maintains aligned forward and backward computations during training.
- We establish the Mutually Exclusive Random Variable (MERVar) (Definition 3.1) by observing the emergence of exclusivity in optimizable deterministic mapping systems involving random variables. Specifically, we mathematically define MERVar to investigate the underlying mechanisms of exclusivity, supported by our MSE Lower Bound Theorem (Theorem 3.3) and Mutual Exclusivity Theorem (Theorem 3.5).
- Comprehensive experiments and discussions are conducted to demonstrate the efficacy and properties of the proposed approach, including: (a) ablation studies to empirically validate the emergence of Mutually Exclusive Random Variables (Sec. 5.1); (b) ablation studies to investigate the performance degradation associated with introducing the Straight-Through Estimator (STE) into MERVar-AE (Sec. 5.3); and (c) comparative analyses between MERVar-AE and prominent discrete representation learning methods across multiple standard datasets under a unified training and evaluation protocol (Sec. 5.5).

2 Related Work

Learning discrete tokens is an important task in computer vision and machine learning (Li et al., 2026b). VQ-VAE (van den Oord et al., 2017) learns a discrete codebook and replaces encoder outputs with their nearest-neighbor embeddings, using the Straight-Through Estimator (STE) (Bengio et al., 2013) to provide surrogate gradients for backpropagation. It has inspired many extensions (Huh et al., 2023; Hsu et al., 2023; Xu et al., 2026). For example, VQ-GAN (Esser et al., 2021) incorporates adversarial and perceptual losses. RQ-VAE (Lee et al., 2022) employs residual quantization to progressively refine latent representations. LooC (Li et al., 2026a) splits feature vectors into multiple segments and applies the same codebook to each segment. SQ-VAE (Takida et al., 2022) introduces self-annealed stochastic quantization within a variational Bayesian framework, using a Gumbel-Softmax relaxation to avoid STE during training while retaining a learned codebook. However,

maintaining and accessing learned codebooks introduces additional computational and memory costs. Lookup-Free Quantization (LFQ) (Yu et al., 2024) avoids an explicit learned codebook by quantizing latent representations into binary codes. Finite Scalar Quantization (FSQ) (Mentzer et al., 2024) eliminates the need for a learned codebook by quantizing each latent dimension to a fixed set of scalar levels, while BSQ (Zhao et al., 2025) performs binary quantization on a spherical latent space. Recent advancements such as VP-VAE (Zhai et al., 2026) bypass explicit codebooks and surrogate gradients through adaptive vector perturbations, while TokenBridge (Wang et al., 2025) adopts a decoupled post-training quantization strategy to bridge continuous and discrete tokens for visual generation. Another direction is the Gumbel-Softmax trick (Jang et al., 2017), which provides a differentiable continuous relaxation of categorical sampling. Rather than representing discreteness through categorical probabilities, MERVar-AE directly injects symmetric noise into bounded continuous latent representations. This provides a simple, codebook-free training mechanism without surrogate gradients. A comparison with representative discrete representation learning methods is provided in Tab. 1.

Table 1: Comparison between the proposed MERVar-AE and representative discrete representation learning methods, where $\mathbf{z}_i^{\mathrm{e}} = e_\theta(\mathbf{x}_i)$ denotes the output of the encoder.

Method	Quantization	Gradient	Core Aux. Loss	Parameters	Binary
VQ-VAE	$\mathrm{argmin}_{c\in\mathcal{C}} \|\mathbf{z}_i^{\mathrm{e}} - c\|$	STE	Commit.+Codeb.	Codebook	$\times$
VP-VAE	$\mathbf{z}_i^{\mathrm{e}} + \delta_i^{\mathrm{MH}}$	Direct BP	Latent Norm.	–	$\times$
SQ-VAE	$c_k,\ k \sim \mathrm{Cat}(\boldsymbol{\pi}_i)$	Gumbel-Softmax	Latent KL	Codebook	$\times$
Gumbel-Softmax	$\mathrm{argmax}_i(\log \pi_i + g_i)$	STE(Hard-mode)	Entropy	–	$\times$
FSQ	$\mathrm{round}(f(\mathbf{z}_i^{\mathrm{e}}))$	STE	–	–	$\times$
TokenBridge	$\mathrm{PTQ}_{\mathrm{dim}}(\mathbf{z}_i^{\mathrm{e}})$	Direct BP	–	–	$\times$
LFQ, BSQ	$\mathrm{sign}(f(\mathbf{z}_i^{\mathrm{e}}))$	STE	–	–	✓
MERVar-AE	$\tanh(\mathbf{z}_i^{\mathrm{e}}) + \sigma \cdot \epsilon_i$	Stoch. Smooth.	–	–	✓

STE: Straight-Through Estimator; BP: Backpropagation; PTQ: Post-Training Quantization; Stoch. Smooth.: Stochastic smoothing; Commit. + Codeb.: Commitment + Codebook; Latent Norm.: Latent normalization; Latent KL: Latent variational terms.

Although our MERVar is newly proposed, its underlying principle is related to optimization inconsistency induced by noise injection, which has also been observed in previous work. β-VAE (Higgins et al., 2017) observed optimization inconsistency when noise is injected. Burgess et al. (2018) explain this inconsistency through the lens of the information bottleneck, while Lucas et al. (2019) suggest that it leads to posterior collapse. This inconsistency can also be observed in the "prior hole" problem (Aneja et al., 2021; Xiao et al., 2020; Nalisnick et al., 2018), considering that a single Gaussian prior is often insufficient for modeling complex data distributions (Vahdat & Kautz, 2020). Semantic hashing (Salakhutdinov & Hinton, 2009) also observed a similar phenomenon and utilized it for document retrieval with near-binary representations, providing an intuitive explanation of why noise can induce binary latents. In contrast, we formulate this phenomenon through Mutually Exclusive Random Variables (MERVar), mathematically characterizing how the overlap between noise-perturbed latent distributions is related to reconstruction ambiguity, and validate this mechanism through comprehensive experiments. Building on this theoretical foundation, we propose the MERVar Autoencoder (MERVar-AE), where discrete representations are learned through overlap reduction in a bounded latent space without relying on STE or a learned codebook.

3 Theoretical Foundations

Notation: Let $\tilde{\mathbf{z}}_i = \mathbf{z}_i + \sigma\epsilon_i$ be a random variable [1] where $\epsilon_i \sim p(\epsilon)$ represents additive noise, $\mathbf{z}_i^e = e_\theta(\mathbf{x}_i)$ denotes the encoder output, and $\mathbf{z}_i = \tanh(\mathbf{z}_i^e)$ denotes the output of the tanh activation. σ is a parameter to control the standard deviation of noise. For brevity, $\|\cdot\|$ denotes the ℓ_2 norm throughout this paper. For any deterministic mapping d_ϕ, the expression $d_\phi(\tilde{\mathbf{z}}_i)$ is understood as the mapping of a realization (sample) of $\tilde{\mathbf{z}}_i$. Optimization objectives involving $d_\phi(\tilde{\mathbf{z}}_i)$ aim to train d_ϕ such that samples drawn from $\tilde{\mathbf{z}}_i$ are mapped to the corresponding target $\mathbf{x}_i$. This ensures that the mapping stays consistent regardless of the noise.

[1] Formally, the random variable $\tilde{\mathbf{z}}_i$ is defined on a common probability space $(\Omega, \mathcal{F}, P)$ as the mapping $\tilde{\mathbf{z}}_i(\omega) = \mathbf{z}_i + \sigma\epsilon_i(\omega)$. For notational simplicity and following common practice in machine learning literature, we omit the sample space element ω in the subsequent sections.

Definition 3.1 (Mutually Exclusive Random Variables (MERVar)) *Let $\tilde{\mathbf{Z}} = \{\tilde{\mathbf{z}}_i\}_{i=1}^N$ be a set of random variables in the latent space $\mathbb{R}^D$, where each $\tilde{\mathbf{z}}_i$ has a probability density function $p_{\tilde{\mathbf{z}}_i}(\mathbf{z})$ centered at a deterministic vector $\mathbf{z}_i$. We say that $\tilde{\mathbf{Z}}$ is a set of ε-mutually exclusive random variables if for every pair $i \neq j$, their overlap coefficient (OC) satisfies:*

$$\mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = \int_{\mathbb{R}^D} \min\left(p_{\tilde{\mathbf{z}}_i}(\mathbf{z}), p_{\tilde{\mathbf{z}}_j}(\mathbf{z})\right) d\mathbf{z} \leq \varepsilon. \tag{2}$$

Given a deterministic decoder $d_\phi : \mathbb{R}^D \to \mathbb{R}^K$ and a set of deterministic targets $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^K$, each latent random variable $\tilde{\mathbf{z}}_i$ is associated with a target $\mathbf{x}_i$, such that the decoder is optimized to map realizations of $\tilde{\mathbf{z}}_i$ to $\mathbf{x}_i$.

- *When $\varepsilon = 0$, the variables are strictly mutually exclusive, implying their densities have disjoint supports.*
- *When $\varepsilon > 0$ is small, they are approximately mutually exclusive, where their probability masses are concentrated in non-overlapping regions of the latent space.*

Remark 3.2 *The concept of MERVar characterizes the dynamic tension between latent random variables and deterministic reconstruction. By injecting noise into the decoder's input, deterministic latent codes are expanded into stochastic random variables. Because the decoder $d_\phi(\cdot)$ is a deterministic mapping, a stochastic input theoretically yields a stochastic output. However, since the optimization objective (reconstruction loss) targets a fixed deterministic variable $\mathbf{x}_i$, a fundamental contradiction arises in regions where the probability distributions of different variables $\tilde{\mathbf{z}}_i$ and $\tilde{\mathbf{z}}_j$ overlap. In these overlapping regions, a single input sample is "pulled" toward multiple distinct targets $\mathbf{x}_i \neq \mathbf{x}_j$, creating a "one-to-many" mapping ambiguity that prevents convergence. To minimize the loss, the optimizer is encouraged to suppress these overlap regions by driving the latent distributions away from one another. Consequently, these random variables emerge as mutually exclusive entities during training, hence the name Mutually Exclusive Random Variables (MERVar).*

Theorem 3.3 (MSE Lower Bound Theorem) *Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^K$ be a set of deterministic targets. Let $\{\tilde{\mathbf{z}}_i\}_{i=1}^N$ be a set of random variables taking values in $\mathbb{R}^D$, defined as $\tilde{\mathbf{z}}_i = \mathbf{z}_i + \sigma\epsilon_i$. Here, $\mathbf{z}_i \in \mathbb{R}^D$ are deterministic latent vectors, and $\{\epsilon_i\}_{i=1}^N$ are independent and identically distributed (i.i.d.) noise vectors drawn from a shared, unimodal, and symmetric distribution $p(\epsilon)$. Let $d_\phi : \mathbb{R}^D \to \mathbb{R}^K$ be a deterministic mapping (e.g., a neural decoder) that maps realizations of $\tilde{\mathbf{z}}_i$ to the corresponding target $\mathbf{x}_i$. The total expected reconstruction loss across all random variables satisfies:*

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_\epsilon\left[\|d_\phi(\tilde{\mathbf{z}}_i) - \mathbf{x}_i\|^2\right] \;\geq\; \frac{1}{4N^2}\sum_{i,j=1}^{N} \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)\,\|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{3}$$

where $\mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)$ denotes the Overlap Coefficient between the probability density functions of $\tilde{\mathbf{z}}_i$ and $\tilde{\mathbf{z}}_j$. For any prescribed reconstruction loss level $L_0 > 0$, achieving a reconstruction loss no greater than L_0 necessarily requires $\sum_{i,j=1}^N \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)\,\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq 4N^2 L_0$. Thus, sufficiently low reconstruction loss necessarily requires sufficiently small distance-weighted aggregate overlap among the latent distributions. In particular, overlap is more strongly penalized for more distinct targets, while this separation pressure vanishes for identical targets. The proof and empirical validation are provided in the supplementary material (Sec. A.1.1 and Sec. A.3.1, respectively).

Remark 3.4 *This lower bound implies a necessary condition for achieving low reconstruction error. Since the reconstruction MSE is always greater than or equal to the overlap-dependent lower bound, achieving a sufficiently low reconstruction error requires this lower bound to also become sufficiently small. Our primary contribution is to characterize this lower bound explicitly using the Overlap Coefficient (OC) and pairwise target distances. Empirically, we observe that this lower bound decreases during reconstruction optimization, consistent with reduced overlap and the emergence of mutual exclusivity among the random variables $\tilde{\mathbf{z}}_i$. In addition, although this lower bound is derived under the mean squared error (MSE), a similar overlap-dependent lower bound can be established for other strongly convex loss functions, with the constant depending on the corresponding convexity parameter.*

Theorem 3.5 (Mutual Exclusivity Theorem) *Let $\{\tilde{\mathbf{z}}_i\}_{i=1}^N$ be isotropic Gaussian random variables with shared covariance $\sigma^2\mathbf{I}$ and centers $\{\mathbf{z}_i\}_{i=1}^N \subset [-1,1]^D$. Then, for any pair $i \neq j$ with $\mathbf{z}_i \neq \mathbf{z}_j$, minimizing their pairwise overlap induces an increase in the pairwise separation of their centers:*

$$\frac{\partial \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)}{\partial \|\mathbf{z}_i - \mathbf{z}_j\|} < 0 \Rightarrow -\nabla_{\mathbf{z}_i} \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) \propto \mathbf{z}_i - \mathbf{z}_j \Rightarrow \|\mathbf{z}_i - \mathbf{z}_j\| \uparrow, \tag{4}$$

The proof is provided in the supplementary material (Sec. A.1.2), with empirical validation provided in Sec. 5.1.

Remark 3.6 *Reducing pairwise overlaps induces the random variables to separate in expectation, thereby encouraging the formation of mutual exclusivity between random variables whose distributions exhibit overlap. Since the latent centers $\mathbf{z}_i$ are bounded within $[-1,1]^D$, maximizing this separation drives them toward the two boundary values. Consequently, each latent coordinate tends toward either -1 or $+1$, naturally yielding a binary representation. We emphasize that the theorem establishes a tendency toward such boundary-seeking behavior rather than guaranteeing coordinate-wise convergence to the binary values. Although the theorem is established under Gaussian noise, the emergence of mutual exclusivity is not restricted to the Gaussian case. For more general symmetric and unimodal noise distributions, reducing the overlap coefficient similarly promotes separation between the corresponding random variables, although the induced notion of separation may not necessarily correspond to the Euclidean distance. We empirically verified this point using different noise distributions in Sec. A.3.3.*

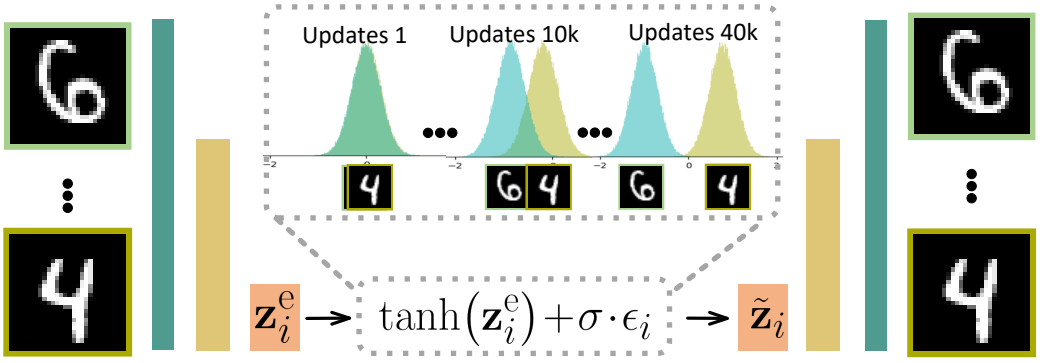

Figure 2: Illustration of the proposed MERVar-AE. The encoder output is passed through a $\tanh$ function, followed by the addition of Gaussian noise controlled by σ. Due to the exclusivity among these Mutually Exclusive Random Variables (MERVar), the latent values are gradually pushed toward the boundaries $\{-1, 1\}$.

4 MERVar Autoencoder

The framework of the proposed MERVar Autoencoder (MERVar-AE) is shown in Fig. 2. Starting from a common autoencoder architecture, images are input into an encoder to obtain the encoder outputs $\mathbf{z}_i^e$. The outputs $\mathbf{z}_i^e$ are then passed through an activation function with a bounded output range, such as the hyperbolic tangent (tanh). Next, noise ϵ_i from a symmetric and unimodal distribution (Gaussian noise) is added to the activation function's output $\mathbf{z}_i$, thereby extending the output into random variables $\tilde{\mathbf{z}}_i = \mathbf{z}_i + \sigma \cdot \epsilon_i$, with the aim of forming MERVar. Due to exclusivity, the network implicitly pushes these latent variables toward distinct, non-overlapping regions at the limits of the $\tanh$ activation (± 1), resulting in an autoencoder with discrete signed binary latents. In our implementation, a latent is considered to have reached a binary limit when its absolute value lies within a numerical tolerance $\delta = 10^{-1}$ of 1, i.e., $|\tilde{\mathbf{z}}| \geq 1 - \delta$. When the percentage of such latents

Table 2: Continuous training and hard-binarization procedures of MERVar-AE.

Continuous Training		**Hard Binarization**	
$\mathbf{z}_i^{\mathrm{e}} \leftarrow e_\theta(\mathbf{x}_i)$	▷ Encode	$\mathbf{z}_i^{\mathrm{e}} \leftarrow e_\theta(\mathbf{x}_i)$	▷ Encode
$\mathbf{z}_i \leftarrow \tanh(\mathbf{z}_i^{\mathrm{e}})$	▷ Bound	$\mathbf{z}_i \leftarrow \tanh(\mathbf{z}_i^{\mathrm{e}})$	▷ Bound
$\tilde{\mathbf{z}}_i \leftarrow \mathbf{z}_i + \sigma \cdot \epsilon_i$	▷ Noise	$\mathbf{z}_i^{\pm} \leftarrow \mathrm{sign}(\mathbf{z}_i)$	▷ Binarize
$\hat{\mathbf{x}}_i \leftarrow d_\phi(\tilde{\mathbf{z}}_i)$	▷ Decode	$\hat{\mathbf{x}}_i \leftarrow d_\phi(\mathbf{z}_i^{\pm})$	▷ Decode
$\theta, \phi \leftarrow \mathrm{Update}(\theta, \phi)$		$\phi \leftarrow \mathrm{Update}(\phi)$	▷ No gradient to θ

exceeds a threshold (e.g., 90%), hard binarization is applied to map all latent values exactly into the discrete set $-1, +1$. Meanwhile, the noise injection is disabled. In particular, since most of the latent values are already close to the boundaries, the reconstruction quality does not exhibit obvious changes when this binarization step is introduced. The ablation study regarding hard binarization is presented in Sec. 5.2. The pseudocode of the proposed MERVar-AE is shown in Tab. 2, in which STE is not used at any stage. After hard binarization, gradients are no longer backpropagated to the encoder, and thus the encoder is no longer updated. The additional training is used solely to improve the decoder quality, which may have been degraded by the injected noise during the continuous training stage.

5 Experiments

5.1 Mutually Exclusive Random Variables

Table 3: Evaluation of the overlap coefficient (OC) and pairwise separation of latent centers during training of MERVar-AE with and without the tanh activation. Values reported as 0 indicate numerical underflow in FP32.

	Updates	0	2k	4k	6k	8k	10k	12k	14k	16k
With tanh	$\mathbb{E}(\mathrm{OC})$	0.99	0.873	0.076	0.030	0.054	0.042	0.036	0.016	0.009
	$\min(\mathrm{OC})$	0.98	0.465	$3.43_{\times 10^{-24}}$	$7.54_{\times 10^{-24}}$	$3.99_{\times 10^{-21}}$	$1.05_{\times 10^{-23}}$	$2.84_{\times 10^{-23}}$	$5.98_{\times 10^{-23}}$	$1.02_{\times 10^{-20}}$
	$\max(\mathrm{OC})$	0.99	0.996	0.995	0.992	0.961	0.946	0.929	0.906	0.910
	$\mathbb{E}\lVert\mathbf{z}_i-\mathbf{z}_j\rVert$	0.004	0.097	2.65	3.25	2.09	2.36	2.30	2.49	2.64
	$\min\lVert\mathbf{z}_i-\mathbf{z}_j\rVert$	0.001	0.003	0.004	0.006	0.029	0.041	0.053	0.071	0.068
	$\max\lVert\mathbf{z}_i-\mathbf{z}_j\rVert$	0.01	0.44	6.09	6.04	5.66	6.02	5.96	5.92	5.60
Without tanh	$\mathbb{E}(\mathrm{OC}^{\mathrm{e}})$	0.99	0.009	0.001	$2.22_{\times 10^{-5}}$	$6.51_{\times 10^{-6}}$	$2.28_{\times 10^{-6}}$	$3.19_{\times 10^{-7}}$	$2.52_{\times 10^{-9}}$	$1.17_{\times 10^{-9}}$
	$\min(\mathrm{OC}^{\mathrm{e}})$	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	$\max(\mathrm{OC}^{\mathrm{e}})$	0.99	0.94	0.79	0.25	0.20	0.16	0.03	$3.26_{\times 10^{-4}}$	$1.52_{\times 10^{-4}}$
	$\mathbb{E}\lVert\mathbf{z}_i^{\mathrm{e}}-\mathbf{z}_j^{\mathrm{e}}\rVert$	0.004	9.27	13.62	18.32	19.79	22.85	26.57	31.88	34.59
	$\min\lVert\mathbf{z}_i^{\mathrm{e}}-\mathbf{z}_j^{\mathrm{e}}\rVert$	0.001	0.05	0.16	0.70	0.77	0.84	1.33	2.16	2.27
	$\max\lVert\mathbf{z}_i^{\mathrm{e}}-\mathbf{z}_j^{\mathrm{e}}\rVert$	0.01	46.60	53.56	66.02	72.53	82.22	97.44	118.69	129.28

The evaluation of the pairwise overlap coefficients among latent random variables throughout training is shown in Tab. 3. Specifically, images from CIFAR-10 are passed through the encoder to obtain the encoder outputs, which are used as the centers of the latent distributions. We evaluate both the standard setting with the $\tanh$ activation and the setting without $\tanh$. The latter removes the bounded constraint on the latent centers, allowing us to better investigate the relationship between the overlap coefficient and pairwise separation. Since Gaussian noise with a fixed standard deviation is injected during training, the overlap coefficient between each pair of latent random variables can be analytically computed from the Euclidean distance between their latent centers. Mathematically:

$$\mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = \mathrm{erfc}\left(\frac{\lVert\mathbf{z}_i - \mathbf{z}_j\rVert}{2\sqrt{2}\sigma}\right), \qquad \tilde{\mathbf{z}}_i = \mathbf{z}_i + \sigma\epsilon_i, \quad \tilde{\mathbf{z}}_j = \mathbf{z}_j + \sigma\epsilon_j. \tag{5}$$

where $\mathrm{erfc}(\cdot)$ denotes the complementary error function. In Tab. 3, OC and $\lVert\mathbf{z}_i - \mathbf{z}_j\rVert$ denote the overlap coefficient and pairwise distance with the $\tanh$ activation, respectively, while OC^{e} and $\lVert\mathbf{z}_i^{\mathrm{e}} - \mathbf{z}_j^{\mathrm{e}}\rVert$ denote their counterparts without the $\tanh$ activation.

5.2 Hard Binarization

The evaluation of hard binarization under different binarization rates is presented in Tab. 4. Images from CIFAR-10 are used for the experiments, with Gaussian noise of $\sigma = 0.3$. Hard binarization is applied when MERVar-AE reaches a predefined binarization rate ranging from 0.1 to 0.9. We report the PSNR at four stages: before hard binarization, immediately after hard binarization with noise, immediately after hard binarization without noise, and after subsequent decoder optimization. The second measurement reflects the reconstruction degradation under hard binarization with noise injection, while the third directly isolates the degradation caused by replacing the continuous latent representations with exact binary codes. When the binarization rate reaches approximately 0.8, the PSNR gap before and immediately after hard binarization without noise becomes negligible. After hard binarization, the decoder is further optimized directly on the exact binary codes used during inference, without noise injection, resulting in a slight improvement in reconstruction performance.

Table 4: Training-set reconstruction performance before and after hard binarization at different binarization rates, using the CIFAR-10 dataset.

Binarization Rate	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Before Hard Binarization	28.57	28.29	27.61	27.45	27.36	26.74	26.52	26.34	25.88
After Hard Binarization (w/ Noise)	15.15	18.29	18.83	18.48	23.28	24.76	24.38	24.43	23.96
After Hard Binarization (w/o Noise)	16.51	21.71	22.53	23.57	25.07	25.49	25.72	26.31	25.89
After Decoder Optimization	21.25	22.38	23.68	24.87	25.45	26.69	26.90	26.52	26.13

Table 5: Ablation study of MERVar and STE-based optimization on CIFAR-10. Training and test PSNR (dB) are reported, where Δ denotes the change from initialization to 7K updates.

Method	Metric	Init.	1K	2K	3K	4K	5K	6K	7K	Δ
STE (Fix Dec.)	Test	25.75	24.99	24.72	24.76	24.72	24.61	20.19	24.35	-1.39
	Train	26.02	24.99	24.83	24.92	24.91	24.77	20.22	24.41	-1.61
STE (Fix Enc.)	Test	25.75	25.70	25.67	25.65	25.63	25.62	25.61	25.60	-0.14
	Train	26.02	26.10	26.13	26.15	26.16	26.17	26.18	26.19	+0.18
STE	Test	25.75	23.25	23.61	23.91	23.95	24.16	24.27	24.22	-1.53
	Train	26.02	23.31	23.73	24.07	24.07	24.31	24.44	24.38	-1.63
MERVar-AE (Fix Dec.)	Test	25.75	25.75	25.75	25.75	25.75	25.75	25.75	25.75	0.00
	Train	26.02	26.02	26.02	26.02	26.02	26.02	26.02	26.02	0.00
MERVar-AE (Fix Enc.)	Test	25.75	25.70	25.67	25.65	25.63	25.62	25.61	25.60	-0.14
	Train	26.02	26.10	26.13	26.15	26.16	26.17	26.18	26.19	+0.18
MERVar-AE	Test	25.75	25.70	25.67	25.65	25.63	25.62	25.61	25.60	-0.14
	Train	26.02	26.10	26.13	26.15	26.16	26.17	26.18	26.19	+0.18

Fix Dec.: Fixed decoder parameters; Fix Enc.: Fixed encoder parameters; Init.: Initialization checkpoint;

5.3 Effect of Straight-Through Gradient Estimation

The investigation of STE on reconstruction optimization is conducted through a controlled ablation using the same trained MERVar-AE checkpoint, as shown in Tab. 5. We compare regular MERVar-AE without STE and MERVar-AE with STE-based hard-binarization training, and additionally freeze either the encoder or decoder to isolate the effects of their updates. All other training configurations, including the random seed, are kept unchanged. Note that after hard binarization, MERVar-AE does not propagate gradients to the encoder; therefore, MERVar-AE and MERVar-AE (Fix Enc.) follow the same optimization trajectory, while MERVar-AE (Fix Dec.) contains no trainable parameters and remains unchanged.

As shown in Tab. 5, when the encoder is frozen, STE (Fix Enc.) and MERVar-AE (Fix Enc.) exhibit identical reconstruction behavior, as expected, since no gradient is propagated through the quantization operation to the encoder. In contrast, enabling STE-based encoder updates substantially degrades reconstruction performance: after 7K updates, the test PSNR decreases from 25.75 dB to 24.22 dB, compared with 25.60 dB for MERVar-AE. A similar degradation is observed when the decoder is frozen and only the encoder is optimized with STE, yielding 24.35 dB. These controlled results localize the degradation primarily to STE-based encoder updates, supporting the observation that the forward-backward gradient mismatch introduced by STE contributes to the performance gap. Interestingly, introducing STE into our encoder optimization reduces performance toward that of STE-based binary quantizers such as BSQ. This suggests that surrogate-gradient mismatch may partly explain this gap. More broadly, we observe a similar phenomenon in both BSQ and VQ-VAE (Sec. A.3.6). In particular, STE-based encoder optimization causes clear degradation when the encoder is optimized without additional latent regularization, whereas freezing the encoder largely avoids this degradation. For VQ-VAE, this degradation is substantially mitigated by the commitment loss, suggesting a reliance on additional continuous constraints to stabilize encoder optimization under STE.

5.4 Generalization Evaluation

The generalization of the proposed MERVar-AE under both in-distribution (ID) and out-of-distribution (OOD) settings on ImageNet and COCO2017 is demonstrated in Tab. 6. For the ID evaluation, MERVar-AE is trained on the training set and evaluated on the corresponding validation set of each dataset. For the OOD evaluation, MERVar-AE is trained on the ImageNet training set and evaluated on the COCO2017 validation set, and vice versa. All images used for evaluation are unseen during training.

Table 6: Generalization evaluation of our MERVar-AE under in-distribution and out-of-distribution settings across COCO2017 and ImageNet using PSNR, SSIM, and LPIPS.

	Codebook Size		2^1	2^2	2^4	2^8	2^{16}	2^{32}	2^{64}	2^{128}
In-Distribution	COCO2017	PSNR	19.50	21.36	24.79	28.07	29.95	32.08	36.22	35.44
		SSIM	0.540	0.634	0.757	0.849	0.894	0.932	0.968	0.968
		LPIPS	0.588	0.508	0.367	0.255	0.194	0.139	0.076	0.075
	ImageNet	PSNR	19.00	20.69	23.73	26.68	28.36	30.27	33.96	33.30
		SSIM	0.498	0.584	0.697	0.801	0.855	0.900	0.948	0.947
		LPIPS	0.623	0.546	0.411	0.298	0.228	0.160	0.088	0.087
Out-of-Distribution	ImageNet - COCO2017	PSNR	19.42	21.23	24.78	28.02	29.86	31.93	36.02	35.16
		SSIM	0.535	0.630	0.751	0.849	0.892	0.929	0.966	0.964
		LPIPS	0.598	0.514	0.371	0.256	0.203	0.148	0.081	0.085
	COCO2017 - ImageNet	PSNR	19.01	20.77	23.72	26.70	28.35	30.23	33.74	33.13
		SSIM	0.501	0.587	0.702	0.801	0.854	0.900	0.946	0.945
		LPIPS	0.618	0.543	0.411	0.299	0.228	0.161	0.091	0.087

ImageNet-COCO2017: Trained on ImageNet and evaluated on COCO2017; COCO2017-ImageNet: Trained on COCO2017 and evaluated on ImageNet;

5.5 Comparison with SOTA Discrete Representation Learning Methods

The proposed approach is compared with state-of-the-art discrete representation learning methods under a unified training and evaluation protocol, with the same learning rate, optimizer, batch size, number of training updates, and encoder-decoder architecture. Detailed experimental settings are provided in the supplementary material (Sec. A.2). The comparisons are conducted across nine datasets, including MNIST, Fashion-MNIST, CIFAR-10, ImageNet, COCO2017, CelebA-HQ, LSUN Bedroom (10k), FFHQ Thumbnail, and Kodak. All comparison experiments are repeated three times, with the mean values reported in Tab. 7. Due to the page limitation, the complete results are provided in the supplementary material (Tabs. 15, 16, 17, 18, 19, 20, 21, 22, 23).

6 Limitations and Discussion

The performance of MERVar-AE is sensitive to the noise intensity. When the noise intensity is too small (e.g., 0.01), the exclusivity between random variables may not be strong enough to push the latent variables toward binary values. When the noise intensity is too large (e.g., 2.0), the reconstruction quality can be substantially degraded. Practicallly, we found that Gaussian noise with σ between 0.1 and 0.3 generally provides desirable results. Another limitation is that the latent variables may move toward the boundaries too quickly when a high noise level and a large learning rate are used. Once the outputs approach ± 1, the gradients of the $\tanh$ function become nearly zero, which may slow down or temporarily stall the learning process. To mitigate this issue, an auxiliary loss such as an L_2 penalty $|\mathbf{z}_i|_2^2$ can be introduced to regularize the latent space. Alternatively, the noise intensity can be gradually increased during training. In our experiments, we used a linear noise schedule that increases the noise intensity from 0.0 to 0.3 as the number of training updates increases.

7 Conclusion

In this work, we introduce the Mutually Exclusive Random Variable (MERVar), a novel theoretical framework that explains how exclusivity emerges in deterministic mapping systems when subject to

Table 7: Comparisons between the proposed MERVar-AE and state-of-the-art discrete representation learning methods, including VQ-VAE (van den Oord et al., 2017), VP-VAE (Zhai et al., 2026), SQ-VAE (Takida et al., 2022), FSQ (Mentzer et al., 2024), BSQ (Zhao et al., 2025), LFQ (Yu et al., 2024), TokenBridge (Wang et al., 2025), Gumbel-Softmax (Jang et al., 2017), and LooC (Li et al., 2026a), across nine image datasets. Results are averaged over three runs, with details provided in the supplementary material.

Method	CIFAR-10			ImageNet			COCO2017		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
VQ-VAE	24.57	0.819	0.241	27.08	0.805	0.294	28.80	0.859	0.241
VP-VAE	22.57	0.750	0.294	24.61	0.730	0.383	25.02	0.756	0.372
SQ-VAE	24.78	0.855	**0.185**	25.78	0.795	0.294	27.89	0.860	0.242
FSQ	23.45	0.790	0.268	25.21	0.759	0.358	25.86	0.796	0.338
BSQ	24.45	0.819	0.231	24.65	0.747	0.376	26.35	0.814	0.321
LFQ	23.75	0.796	0.259	25.71	0.775	0.341	26.79	0.820	0.305
Gumbel.	23.38	0.784	0.285	25.48	0.752	0.359	27.44	0.826	0.293
TokenBridge	21.04	0.615	0.435	23.39	0.659	0.532	24.62	0.716	0.430
LooC	25.02	0.836	0.220	26.73	0.796	0.310	27.79	0.829	0.293
Ours	**25.68**	**0.858**	0.188	**27.66**	**0.826**	**0.262**	**29.07**	**0.870**	**0.231**

Method	CelebA-HQ			LSUN Bedroom (10k)			Kodak		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
VQ-VAE	30.25	0.862	0.295	28.17	0.838	0.314	26.96	0.885	0.376
VP-VAE	27.06	0.796	0.391	25.73	0.779	0.378	23.74	0.799	0.455
SQ-VAE	30.88	0.893	**0.186**	**29.93**	**0.897**	**0.208**	26.97	0.882	0.351
FSQ	29.09	0.870	0.266	26.63	0.827	0.321	24.85	0.850	0.443
BSQ	28.99	0.861	0.281	27.41	0.843	0.297	21.09	0.724	0.383
LFQ	29.95	0.885	0.236	26.17	0.811	0.342	27.31	0.890	0.358
Gumbel.	30.55	0.881	0.240	27.91	0.834	0.302	24.00	0.820	0.456
TokenBridge	27.96	0.821	0.353	25.04	0.743	0.430	22.53	0.767	0.539
LooC	29.73	0.864	0.297	28.25	0.847	0.312	26.91	0.885	0.377
Ours	**31.69**	**0.900**	0.217	29.67	0.882	0.240	**28.67**	**0.919**	**0.300**

Method	FFHQ Thumbnail			Fashion-MNIST			MNIST		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
VQ-VAE	27.24	0.836	0.261	25.10	0.866	0.080	27.90	0.958	0.022
VP-VAE	24.81	0.771	0.325	20.52	0.705	0.175	21.08	0.841	0.084
SQ-VAE	24.56	0.835	0.225	26.22	**0.910**	**0.047**	28.38	0.980	0.011
FSQ	25.83	0.817	0.280	20.55	0.727	0.148	21.60	0.899	0.041
BSQ	26.29	0.834	0.253	23.69	0.843	0.086	25.97	0.966	0.019
LFQ	26.42	0.836	0.255	22.03	0.794	0.109	24.08	0.945	0.028
Gumbel.	26.36	0.819	0.271	23.13	0.830	0.092	25.45	0.963	0.021
TokenBridge	23.94	0.730	0.431	19.48	0.633	0.222	20.43	0.800	0.116
LooC	27.42	0.844	0.249	24.45	0.861	0.074	27.99	0.976	0.016
Ours	**28.57**	**0.875**	**0.190**	**26.60**	0.909	0.049	**29.75**	**0.987**	**0.010**

Gumbel.: Gumbel-Softmax

noise injection. Through rigorous mathematical proof and the proposed MSE Lower Bound theorem, we demonstrate that, under the stated assumptions, achieving sufficiently low reconstruction error requires a sufficiently small overlap coefficient among the latent random variables. Empirically, we observe that the latent variables move toward non-overlapping regions during reconstruction training. Based on this principle, the MERVar-AE is proposed to learn binary latent representations. Compared with previous methods such as VQ-VAE and BSQ, our MERVar-AE eliminates the need for complex codebook management and Straight-Through Estimator (STE), thereby maintaining aligned forward and backward computations during training. Comprehensive experiments on standard datasets demonstrate that the proposed approach achieves competitive results, suggesting that MERVar-based exclusivity is a promising direction for future research in discrete 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 derivations underlying MERVar are presented in Sec. 3, with complete proofs provided in Sec. A.1. The MERVar-AE architecture and its continuous-training and hard-binarization procedures are described in Sec. 4 and Tab. 2. Sec. 5.5 presents the evaluation protocols and controlled comparisons, while Sec. A.2 provides the model architecture, optimization settings, dataset splits, baseline configurations, activation functions, and noise distributions. Additional validation, ablation, downstream-generation, robustness, and qualitative results are reported in Sec. A.3. The main comparisons are repeated over three runs, with complete results reported in Tabs. 15, 16, 17, 18, 19, 20, 21, 22, 23. The supplementary material also provides the CIFAR-10 evaluation code and pretrained model, together with an analysis of the performance degradation caused by surrogate gradients in STE-based training. The full source code of the proposed MERVar-AE will be publicly released upon acceptance of this work.

REFERENCES

Jyoti Aneja, Alex Schwing, Jan Kautz, and Arash Vahdat. A contrastive learning approach for training variational autoencoder priors. *Advances in neural information processing systems*, 34:480–493, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta -vae. *arXiv preprint arXiv:1804.03599*, 2018.

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.

Kyle Hsu, William Dorrell, James Whittington, Jiajun Wu, and Chelsea Finn. Disentanglement via latent quantization. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 45463–45488. Curran Associates, Inc., 2023.

Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning*, pp. 14096–14113. PMLR, 2023.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

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 (CVPR)*, pp. 11523–11532, June 2022.

Jie Li, Kwan-Yee K. Wong, and Kai Han. Looc: Effective low-dimensional codebook for compositional vector quantization. In *Proc. Winter Conference on Applications of Computer Vision*, Tucson, Arizona, USA, March 2026a.

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*, 2026b.

James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

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

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2018.

Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

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 Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20987–21012. PMLR, 17–23 Jul 2022.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Yuqing Wang, Zhijie Lin, Yao Teng, Yuanzhi Zhu, Shuhuai Ren, Jiashi Feng, and Xihui Liu. Bridging continuous and discrete tokens for autoregressive visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18596–18605, 2025.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*, 2020.

Tongda Xu, Wendi Zheng, Jiajun He, José Miguel Hernández-Lobato, Yan Wang, Ya-Qin Zhang, and Jie Tang. Training-free vector quantization via gaussian VAEs. In *Forty-third International Conference on Machine Learning*, 2026.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Linwei Zhai, Han Ding, Mingzhi Lin, Cui Zhao, Fei Wang, Ge Wang, Wang Zhi, and Wei Xi. Vp-vae: Rethinking vector quantization via adaptive vector perturbation. *arXiv preprint arXiv:2602.17133*, 2026.

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.

A Appendix

A.1 Mathematical Derivation

A.1.1 Proof of Reconstruction MSE Lower Bound in Theorem 3.3

Given a set of images $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$, an encoder e_θ and a decoder d_ϕ, let $\mathbf{z}_i = \tanh(e_\theta(\mathbf{x}_i))$ and inject symmetric, unimodal noise to obtain $\tilde{\mathbf{z}}_i = \mathbf{z}_i + \sigma \cdot \epsilon_i$ with density $p_{\tilde{\mathbf{z}}_i}(\cdot)$. For any pair (i, j) define:

$$p_m^{(i,j)}(\mathbf{z}) \;=\; \min\big(p_{\tilde{\mathbf{z}}_i}(\mathbf{z}),\, p_{\tilde{\mathbf{z}}_j}(\mathbf{z})\big). \tag{6}$$

Then:

$$\begin{aligned}
\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\big[\|d(\tilde{\mathbf{z}}_i)-\mathbf{x}_i\|^2\big] &= \frac{1}{2N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\Big(\mathbb{E}\|d(\tilde{\mathbf{z}}_i)-\mathbf{x}_i\|^2+\mathbb{E}\|d(\tilde{\mathbf{z}}_j)-\mathbf{x}_j\|^2\Big)\\
&= \frac{1}{2N^2}\sum_{i,j}\Big(\int p_{\tilde{\mathbf{z}}_i}(\mathbf{z})\|d(\mathbf{z})-\mathbf{x}_i\|^2\,d\mathbf{z}+\int p_{\tilde{\mathbf{z}}_j}(\mathbf{z})\|d(\mathbf{z})-\mathbf{x}_j\|^2\,d\mathbf{z}\Big)\\
&= \frac{1}{2N^2}\sum_{i,j}\int p_m^{(i,j)}(\mathbf{z})\Big(\|d(\mathbf{z})-\mathbf{x}_i\|^2+\|d(\mathbf{z})-\mathbf{x}_j\|^2\Big)\,d\mathbf{z}\\
&\quad+\frac{1}{2N^2}\sum_{i,j}\zeta_{ij}(\mathbf{Z},\mathbf{X}),
\end{aligned} \tag{7}$$

where the expression of $\zeta_{ij}(\mathbf{Z}, \mathbf{X})$ is:

$$\zeta_{ij}(\mathbf{Z},\mathbf{X}) := \int \big(p_{\tilde{\mathbf{z}}_i}(\mathbf{z})-p_m^{(i,j)}(\mathbf{z})\big)\|d(\mathbf{z})-\mathbf{x}_i\|^2\,d\mathbf{z}+\int \big(p_{\tilde{\mathbf{z}}_j}(\mathbf{z})-p_m^{(i,j)}(\mathbf{z})\big)\|d(\mathbf{z})-\mathbf{x}_j\|^2\,d\mathbf{z}. \tag{8}$$

Let $A_{ij} = \{\mathbf{z} : p_{\tilde{\mathbf{z}}_i}(\mathbf{z}) \geq p_{\tilde{\mathbf{z}}_j}(\mathbf{z})\}$ and $B_{ij} = A_{ij}^{\complement}$. On A_{ij}, $p_{\tilde{\mathbf{z}}_i} - p_m^{(i,j)} = p_{\tilde{\mathbf{z}}_i} - p_{\tilde{\mathbf{z}}_j} \geq 0$ and $p_{\tilde{\mathbf{z}}_j} - p_m^{(i,j)} = 0$; on B_{ij}, the roles swap. Since the weights are nonnegative and the squared terms are nonnegative, we have:

$$\boxed{\zeta_{ij}(\mathbf{Z},\mathbf{X}) \geq 0}.$$

Using the following inequality,

$$\|d(\mathbf{z})-\mathbf{x}_i\|^2+\|d(\mathbf{z})-\mathbf{x}_j\|^2 \;\geq\; \tfrac{1}{2}\|\mathbf{x}_i-\mathbf{x}_j\|^2, \tag{9}$$

hence:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\big[\|d(\tilde{\mathbf{z}}_i)-\mathbf{x}_i\|^2\big] \;\geq\; \frac{1}{4N^2}\sum_{i,j}\|\mathbf{x}_i-\mathbf{x}_j\|^2\int p_m^{(i,j)}(\mathbf{z})\,d\mathbf{z} \;+\; \frac{1}{2N^2}\sum_{i,j}\zeta_{ij}(\mathbf{Z},\mathbf{X}). \tag{10}$$

Since $\int p_m^{(i,j)}(\mathbf{z})\,d\mathbf{z} = \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)$ and $\zeta_{ij} \geq 0$, we obtain the lower bound

$$\boxed{\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\big[\|d(\tilde{\mathbf{z}}_i)-\mathbf{x}_i\|^2\big] \;\geq\; \frac{1}{4N^2}\sum_{i,j=1}^{N}\mathrm{OC}(\tilde{\mathbf{z}}_i,\tilde{\mathbf{z}}_j)\,\|\mathbf{x}_i-\mathbf{x}_j\|^2}.$$

Remark A.1 *Although the above lower bound is derived under the squared error loss, the key structure does not rely on the specific quadratic form. The overlap coefficient* $\mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)$ *arises solely from the probabilistic overlap of the perturbed latent codes and is independent of the loss. The coefficient* $\frac{1}{2}$ *multiplying* $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ *in Eq. 9 originates from the strong convexity of the squared norm. For a general strongly convex loss, one may obtain an analogous lower bound where this coefficient changes according to the corresponding convexity parameter. Thus, the phenomenon that reconstruction error is fundamentally limited by the overlap coefficient in MERVar is not specific to the squared loss, but extends to a broader family of strongly convex losses.*

A.1.2 Proof of Mutual Exclusivity in Theorem 3.5

The expression of Theorem 3.5 is shown as:

$$\frac{\partial \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)}{\partial \|\mathbf{z}_i - \mathbf{z}_j\|} < 0 \Rightarrow -\nabla_{\mathbf{z}_i} \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) \propto \mathbf{z}_i - \mathbf{z}_j \Rightarrow \|\mathbf{z}_i - \mathbf{z}_j\| \uparrow . \tag{11}$$

where the noise follows an isotropic Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with $f(\cdot)$ as its probability density function. Therefore, the probability density functions of $\tilde{\mathbf{z}}_i$ and $\tilde{\mathbf{z}}_j$ are:

$$p_{\tilde{\mathbf{z}}_i}(\mathbf{z}) = \frac{1}{\sigma^D} f\left(\frac{\mathbf{z} - \mathbf{z}_i}{\sigma}\right), \quad p_{\tilde{\mathbf{z}}_j}(\mathbf{z}) = \frac{1}{\sigma^D} f\left(\frac{\mathbf{z} - \mathbf{z}_j}{\sigma}\right), \tag{12}$$

where D denotes the dimensionality of $\mathbf{z}$. Then, by plugging this expression into the overlap coefficient in Eq. 2, we have:

$$\mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = \int \min\big(p_{\tilde{\mathbf{z}}_i}(\mathbf{z}), p_{\tilde{\mathbf{z}}_j}(\mathbf{z})\big)\, d\mathbf{z}. \tag{13}$$

For any pair $i \neq j$ with distinct centers, let

$$d_{ij} = \|\mathbf{z}_i - \mathbf{z}_j\| > 0. \tag{14}$$

Since the two Gaussian distributions have the same isotropic covariance, their overlap coefficient depends solely on the Euclidean distance between their centers and can be written as

$$\mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = 2\Phi\left(-\frac{d_{ij}}{2\sigma}\right) = \mathrm{erfc}\left(\frac{d_{ij}}{2\sqrt{2}\sigma}\right) =: h(d_{ij}), \tag{15}$$

where $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution. Therefore,

$$\frac{\partial \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)}{\partial d_{ij}} = h'(d_{ij}) = -\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{d_{ij}^2}{8\sigma^2}\right) < 0. \tag{16}$$

Thus, the overlap coefficient is strictly decreasing with respect to the Euclidean distance between the two latent centers.

By the chain rule,

$$\nabla_{\mathbf{z}_i} \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = \frac{\partial \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j)}{\partial d_{ij}} \nabla_{\mathbf{z}_i} d_{ij} = h'(d_{ij}) \frac{\mathbf{z}_i - \mathbf{z}_j}{\|\mathbf{z}_i - \mathbf{z}_j\|}. \tag{17}$$

Let

$$\alpha_{ij} = -\frac{h'(d_{ij})}{d_{ij}} > 0. \tag{18}$$

Then, the negative gradient direction is

$$-\nabla_{\mathbf{z}_i} \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = \alpha_{ij}(\mathbf{z}_i - \mathbf{z}_j) \propto \mathbf{z}_i - \mathbf{z}_j. \tag{19}$$

Since $\alpha_{ij} > 0$, this direction points from $\mathbf{z}_j$ toward $\mathbf{z}_i$, and therefore pushes $\mathbf{z}_i$ away from $\mathbf{z}_j$. Specifically, under a gradient-descent update with step size $\eta > 0$, the updated center $\mathbf{z}_i'$ satisfies:

$$\mathbf{z}_i' = \mathbf{z}_i - \eta \nabla_{\mathbf{z}_i} \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j) = \mathbf{z}_i + \eta \alpha_{ij}(\mathbf{z}_i - \mathbf{z}_j). \tag{20}$$

Hence,

$$\|\mathbf{z}_i' - \mathbf{z}_j\| = \|(1 + \eta\alpha_{ij})(\mathbf{z}_i - \mathbf{z}_j)\| = (1 + \eta\alpha_{ij})\|\mathbf{z}_i - \mathbf{z}_j\| > \|\mathbf{z}_i - \mathbf{z}_j\|. \tag{21}$$

Therefore, minimizing the pairwise overlap induces a repulsive update between the corresponding latent centers and increases their pairwise Euclidean separation. For the full pairwise overlap objective

$$J(\mathbf{z}) = \sum_{i<j} \mathrm{OC}(\tilde{\mathbf{z}}_i, \tilde{\mathbf{z}}_j), \tag{22}$$

the negative gradient with respect to $\mathbf{z}_i$ is

$$-\nabla_{\mathbf{z}_i} J = \sum_{j \neq i} \alpha_{ij}(\mathbf{z}_i - \mathbf{z}_j), \tag{23}$$

which is a weighted sum of the pairwise repulsive directions established above. Thus, joint overlap minimization aggregates these pairwise repulsive interactions among the latent centers, encouraging their overall separation and increasing the mutual exclusivity of the corresponding latent random variables. In a bounded space such as $[-1, 1]^D$, these repulsive interactions further promote separation toward the boundary of the hypercube, thereby encouraging binarization. Theorem 3.5 is thus proved.

Remark A.2 *This proof shows that, under isotropic Gaussian noise, minimizing overlap between latent distributions induces pairwise repulsive updates that increase the Euclidean separation between their centers. The emergence of mutual exclusivity, however, is not restricted to Gaussian noise. For more general symmetric and unimodal noise distributions, reducing the overlap coefficient similarly promotes separation between the corresponding latent distributions, although the induced notion of separation may not necessarily correspond to the Euclidean distance.*

A.2 Experimental Details

Table 8: Architecture details of MERVar-AE with latent size 8×8, using the CIFAR-10 dataset.

	Type	Weight	Stride	Padding	Data Size
Encoder	Input	–	–	–	$N \times 3 \times 32 \times 32$
	Conv2d	$3 \times 64 \times 4 \times 4$	2	1	$N \times 64 \times 16 \times 16$
	ReLU	–	–	–	$N \times 64 \times 16 \times 16$
	Conv2d	$64 \times 128 \times 4 \times 4$	2	1	$N \times 128 \times 8 \times 8$
	ReLU	–	–	–	$N \times 128 \times 8 \times 8$
	Conv2d	$128 \times 128 \times 3 \times 3$	1	1	$N \times 128 \times 8 \times 8$
	ResConv2d	$128 \times 128 \times 3 \times 3$	1	1	$N \times 128 \times 8 \times 8$
	ResConv2d	$128 \times 128 \times 3 \times 3$	1	1	$N \times 128 \times 8 \times 8$
	Conv2d	$128 \times d \times 3 \times 3$	1	1	$N \times d \times 8 \times 8$
Lat.	Binarize	–	–	–	$N \times d \times 8 \times 8$
Decoder	Conv2d	$d \times 128 \times 3 \times 3$	1	1	$N \times 128 \times 8 \times 8$
	Conv2d	$128 \times 64 \times 3 \times 3$	1	1	$N \times 64 \times 8 \times 8$
	Conv2d	$64 \times 128 \times 3 \times 3$	1	1	$N \times 128 \times 8 \times 8$
	ResConv2d	$128 \times 128 \times 3 \times 3$	1	1	$N \times 128 \times 8 \times 8$
	ResConv2d	$128 \times 128 \times 3 \times 3$	1	1	$N \times 128 \times 8 \times 8$
	ConvT2d	$128 \times 64 \times 4 \times 4$	2	1	$N \times 64 \times 16 \times 16$
	ReLU	–	–	–	$N \times 64 \times 16 \times 16$
	ConvT2d	$64 \times 3 \times 4 \times 4$	2	1	$N \times 3 \times 32 \times 32$
	Output	–	–	–	$N \times 3 \times 32 \times 32$

ResConv2d: A residual block consisting of ReLU, a 3×3 convolution with 32 intermediate channels, ReLU, and a 1×1 convolution restoring the input channel dimension, with an identity skip connection. Both convolutions use stride 1 and no bias. **Lat.** = Latents;

We implement our method in PyTorch and use the Adam optimizer. All experiments are conducted on a server equipped with an NVIDIA RTX 4080 GPU (16 GB). Unless otherwise specified, all compared methods are initialized without external pretrained checkpoints and trained under the same experimental setting to ensure a controlled comparison. All methods share the same encoder-decoder architecture and a learning rate of 3×10^{-4}. The encoder-decoder architecture used for evaluation is directly adopted from the publicly available VQ-VAE implementation on GitHub, with the details in Tab. 8. The corresponding network implementation is also provided in the supplementary material. All methods are first trained for 50,000 optimization updates on CIFAR-10 with a batch size of 512 and then further trained for 10,000 additional updates on each target dataset with a batch size of 32 due to our limited computational resources. For methods involving two-stage training, such as the proposed approach and Gumbel-Softmax, the final 1,000 updates of the corresponding training budget are allocated to the second stage. For datasets with official training/test or training/validation splits, models are further trained on the corresponding training set and evaluated on the test or validation set,

including MNIST, Fashion-MNIST, CIFAR-10, ImageNet, and COCO2017. For datasets without an official training/test split, 80% of the images are randomly selected for training, while the remaining 20% are used for evaluation. For Kodak, the model pretrained on ImageNet is directly used for evaluation without further training, since the dataset contains only a limited number of images. The complete set of comparison experiments takes approximately three weeks to complete, whereas a complete training run of the proposed approach takes less than 24 hours.

All compared methods are implemented based on their official implementations. For a fair comparison, the discrete representation capacity of each method is configured to match the same effective codebook size of 2^{10}, equivalently 10 bits per latent token. For methods without an explicit codebook, the effective codebook size is defined as the total number of distinct discrete states representable by each latent token. **VQ-VAE** uses an embedding dimension of 64, a commitment cost of $\beta = 0.25$, and 1024 embeddings, corresponding to a codebook size of 2^{10}. **VP-VAE** adopts the official `VectorPerturbation` module with 1024 embeddings (2^{10}). **SQ-VAE** follows the official Gaussian SQ-VAE (I) implementation with 1024 codebook embeddings (2^{10}) of dimension 64, retaining its ARELBO objective, latent variational terms, and Gumbel-Softmax temperature annealing schedule. **FSQ** uses $L = 4$ quantization levels and $d = 5$ latent dimensions, yielding $4^5 = 2^{10}$ possible codes. **BSQ** adopts the official `BinarySphericalQuantizer` with $d = 10$ binary dimensions, yielding 2^{10} possible codes. **Gumbel-Softmax** uses $2^{10} = 1024$ categorical states with a temperature of $\tau = 1.0$. **LFQ** performs per-dimension sign binarization with $d = 10$ binary dimensions, yielding 2^{10} possible codes. **TokenBridge** follows its official post-training quantization procedure, where a continuous autoencoder is first trained and subsequently frozen before quantization, using 10 bits per latent position to represent 2^{10} possible codes. **LooC** follows its official implementation, splitting the 64-dimensional latent vector into two segments that share the same codebook of 32 embeddings, yielding $(2^5)^2 = 2^{10}$ possible combined codes, while retaining its default configuration.

A.2.1 Implementation details of Symmetric Noise and Activation Function

The details of the activation functions and noise distributions are shown in Tab. 9 and Tab. 10, respectively. For the activation function ablation, Gaussian noise is fixed while different activation functions and their corresponding binarization thresholds are evaluated. For the noise type ablation, $\tanh$ is fixed while different symmetric noise distributions are evaluated.

Table 9: Activation functions evaluated in the ablation study.

Activation	Formula	Output range	Threshold
Tanh	$\mathrm{Tanh}(\mathbf{z}^{\mathrm{e}})$	$(-1,\ 1)$	0
Clamp	$\mathrm{clamp}(\mathbf{z}^{\mathrm{e}},\ -1,\ 1)$	$[-1,\ 1]$	0
Softsign	$\mathrm{Softsign}(\mathbf{z}^{\mathrm{e}})$	$(-1,\ 1)$	0
ReLU6	$\mathrm{ReLU6}(\mathbf{z}^{\mathrm{e}})$	$[0,\ 6]$	3
Sigmoid	$\mathrm{Sigmoid}(\mathbf{z}^{\mathrm{e}})$	$(0,\ 1)$	0.5

Table 10: Noise distributions evaluated in the ablation study.

Noise type	Distribution formula
Gaussian	$\epsilon \sim \mathcal{N}(0,1)$
Uniform	$\epsilon = 2u - 1, \quad u \sim \mathcal{U}(0,1)$
Triangular	$\epsilon = u_1 + u_2 - 1, \quad u_1, u_2 \sim \mathcal{U}(0,1)$
Logistic	$\epsilon = \log(u/(1-u)), \quad u \sim \mathcal{U}(\varepsilon_0, 1-\varepsilon_0)$
Laplace	$\epsilon = -\,\mathrm{sgn}(u-0.5)\ln(1-2\lvert u-0.5\rvert), \quad u \sim \mathcal{U}(\varepsilon_0, 1-\varepsilon_0)$

where $\mathrm{sgn}(\cdot)$ denotes the sign function, and $\varepsilon_0{=}10^{-6}$ ensures numerical stability in the inverse-CDF sampling of the Logistic and Laplace distributions.

A.3 Additional Experiments

A.3.1 MSE Lower Bound Theorem

To validate the MSE Lower Bound Theorem, we evaluated the MERVar autoencoder using varying Gaussian noise levels, with $\sigma \in \{0.0, 0.2, 0.4, 0.8, 1.0\}$. For each configuration, the decoder is trained for a sufficient number of updates to ensure convergence. Based on Theorem 3.3, as σ increases, the overlap coefficient grows accordingly, which leads to a higher reconstruction error bound, thereby degrading the reconstruction quality. The relationship between σ and the average Peak Signal-to-Noise Ratio (PSNR) is illustrated in Fig. 3. A clear monotonic trend is observed: higher noise intensity σ results in lower reconstruction quality.

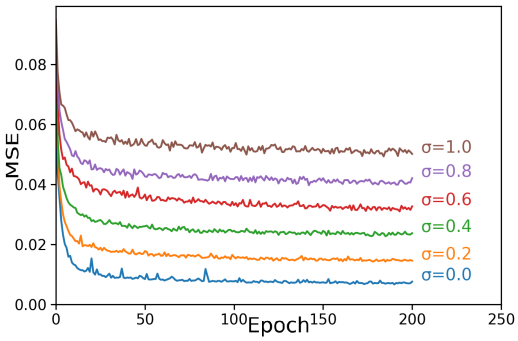

Figure 3: Ablation study of the MSE lower-bound theorem 3.3. Increasing σ elevates the reconstruction lower bound. We evaluate several MERVar autoencoders across varying σ values to empirically validate this theoretical constraint.

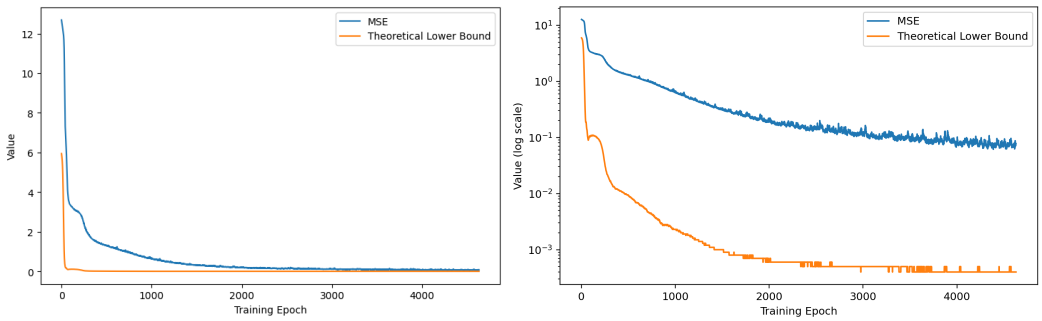

Figure 4: Training dynamics of the reconstruction MSE and the theoretical overlap-dependent lower bound. The lower bound remains below the reconstruction MSE throughout training and decreases rapidly during the early stage of optimization. The logarithmic-scale visualization further shows their overall decreasing trends across the full training trajectory.

To empirically investigate the MSE lower bound established in Theorem 3.3, we track both the reconstruction MSE and its corresponding overlap-dependent lower bound throughout training. As shown in Fig. 4, the theoretical lower bound remains below the empirical reconstruction MSE throughout the entire optimization process, consistent with the inequality established in Theorem 3.3. Notably, the lower bound decreases rapidly during the early stage of training, indicating that reconstruction optimization quickly reduces the overlap-dependent irreducible component by separating the latent distributions. The reconstruction MSE subsequently continues to decrease while the lower bound becomes comparatively small. Overall, the two quantities exhibit consistent decreasing trends, with a Pearson correlation of 0.7333 and a Spearman correlation of 0.9731 over the complete training trajectory. These observations empirically support the overlap-dependent lower-bound mechanism predicted by Theorem 3.3.

A.3.2 Visualization of Mutually Exclusive Random Variables

The visualization of mutually exclusive random variables is shown in Fig. 5. In particular, 16 images are randomly selected from CIFAR-10 for visualization. During training, we plot the observed values of $\tilde{\mathbf{z}}_i$ mapped to image $\mathbf{x}_i$. Moreover, observed values from the same random variable, which are used to reconstruct the same image, are colored the same. For better visualization, the latent dimension is set to 2, with Gaussian noise controlled by $\sigma = 0.1$. At the beginning, the distributions of different latent random variables $\tilde{\mathbf{z}}_i$ are overlapping, and the reconstruction quality is poor. With the increasing number of training updates, these random variables move away from each other, as if their probability distributions are mutually exclusive.

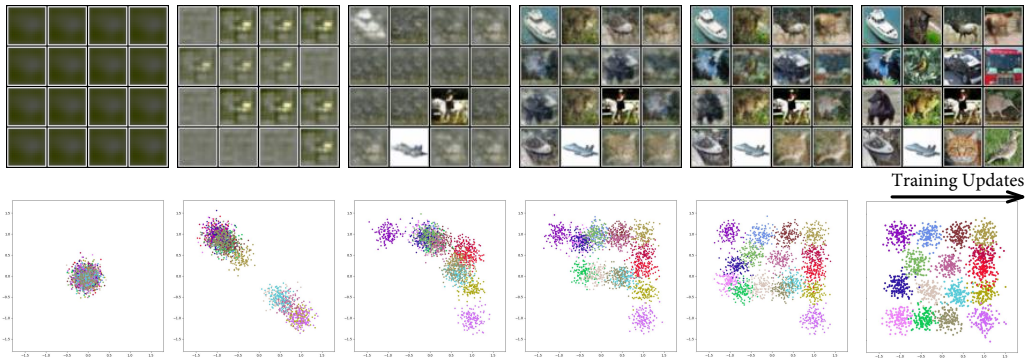

Figure 5: Demonstration of the mutually exclusive random variables. As training updates increase, the distributions of latent random variables move away from each other, eventually becoming nearly non-overlapping. In the second row, points of the same color represent the observed values sampled from the same latent variable $\tilde{\mathbf{z}}_i$.

A.3.3 Symmetric Noise and Activation Function

The proposed MERVar-AE supports various activation functions and noise types. To demonstrate this, we evaluated MERVar-AE with activation functions including Tanh, Clamp, Softsign, ReLU6, and Sigmoid, and noise types including Gaussian, Uniform, Triangular, Logistic, and Laplace. The experimental results are presented in Tab. 11. We use the CIFAR-10 dataset for evaluation with a fixed spatial latent size of 8×8. The autoencoder architecture is taken from the classic VQ-VAE (van den Oord et al., 2017) implementation on GitHub. Our results show that, while various activation functions yield similar reconstruction performance, $\tanh$ achieves the best results. In particular, Gaussian noise with $\sigma = 0.3$ is used for the activation function experiments. Consequently, $\tanh$ is adopted for the subsequent noise-type experiments, as shown in the right part of Tab. 11. For these experiments, σ is reduced to 0.1 for all noise types because Logistic noise did not yield satisfactory reconstruction results with $\sigma = 0.3$. Thus, the two Tanh + Gaussian entries correspond to different noise scales rather than identical experimental configurations. Due to the distinct probability density functions (PDFs) of different noise types, the overlap coefficient varies even at the same noise scale $\sigma = 0.1$. This variation affects the reconstruction lower bound and may help explain the PSNR differences across noise types. Regardless of the noise type, PSNR consistently improves as the number of latent dimensions d increases.

Table 11: Ablation experiments of the proposed MERVar-AE across various activation functions and noise types. Left: activation ablation with Gaussian noise and $\sigma = 0.3$. Right: noise ablation with Tanh activation and $\sigma = 0.1$. 2^d represents the total number of possible discrete code combinations, where d is the number of latent dimensions.

Activation	2^1	2^3	2^6	2^8	2^{11}	Noise	2^1	2^3	2^6	2^8	2^{11}
Tanh	18.39	21.40	25.45	24.95	26.01	Gaussian	20.27	23.12	25.71	26.42	27.69
Clamp	18.40	21.23	23.18	24.32	25.42	Uniform	21.13	24.44	27.26	28.27	29.46
Softsign	18.48	21.30	22.99	24.41	25.58	Triangular	21.66	25.08	27.98	29.27	30.94
ReLU6	17.79	19.55	22.50	23.14	23.09	Logistic	18.71	21.38	23.37	24.35	25.20
Sigmoid	19.05	21.44	23.22	24.12	25.07	Laplace	19.57	22.12	24.38	25.22	26.47

A.3.4 Quantitative Experiments of Mutually Exclusive Random Variables

The quantitative experiments of the mutually exclusive random variables are presented in Fig. 6. In particular, we demonstrate the histogram of latents during training and compute the binary rate of all latent elements. The binary rate is computed based on the rate of latents whose absolute values have a distance of less than 10^{-1} from the boundaries. During training, as the PSNR values increase, the binary rate also rises. The histogram of the latents progressively concentrates toward the boundaries, ultimately becoming highly concentrated around $\{-1, 1\}$, indicating increasingly binary latent representations.

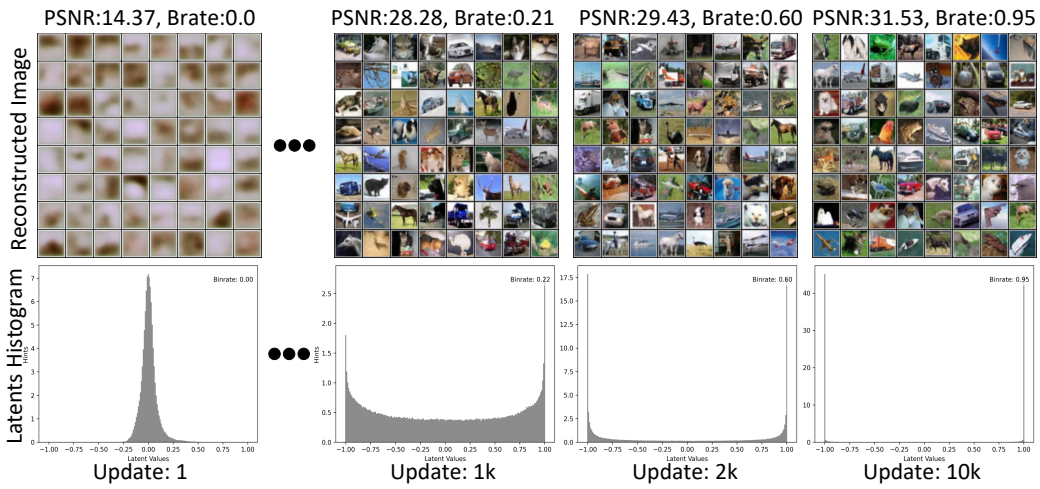

Figure 6: Demonstration of the binary rate of latent values (brate) and reconstruction quality under the Peak Signal-to-Noise Ratio (PSNR) across training updates. The histogram of latent values is illustrated on the second row.

A.3.5 Interpolation in the Binary Latent Space

An interesting advantage of the proposed approach is that it provides a binary latent space with smooth semantic transitions. Although the latent space is binary, meaningful transitions can still be obtained owing to the injected noise. To demonstrate this, we extracted the latent codes from two different images, then applied interpolation between these two latent codes and decoded them back. In particular, different interpolation techniques are utilized according to the representations of different discrete autoencoders. For example, probability-flipping interpolation is used for the proposed approach and BSQ, whereas direct interpolation is adopted for FSQ since its latent codes consist of integer-valued levels. The comparison results are shown in Fig. 7. The proposed approach exhibits the smoothest latent space among all compared methods. FSQ also produces meaningful interpolation results due to its ordered integer-valued latents. A similar situation is observed for TokenBridge and BSQ. However, noticeable artifacts and noise are present in their interpolated results. In contrast, VQ-VAE, VP-VAE, SQ-VAE, and Gumbel-Softmax represent latents using discrete token indices that do not impose an explicit notion of local similarity between neighboring indices. Consequently, their token spaces do not naturally preserve smooth local transitions. This difference may partially explain the advantage observed in our downstream generation experiments (Sec. A.3.7), as the proposed representation provides a smoother latent space.

A.3.6 STE Optimization in Existing Quantizers

To investigate whether the optimization degradation caused by STE is specific to MERVar-AE, we further conduct controlled experiments on BSQ and VQ-VAE, as shown in Tab. 12 and Tab. 13. We directly load the trained checkpoints used for the CIFAR-10 comparison in Tab. 15, and continue optimization from the same checkpoints. Therefore, the models and training configurations are identical to those used in our main comparison. The learning rate and optimizer follow the default settings described in the Experimental Details. For each method, we separately freeze the encoder or decoder while keeping the remaining components trainable. For VQ-VAE, the quantizer and codebook are regarded as part of the decoder and are therefore frozen together with the decoder.

A similar phenomenon is observed in BSQ. When the encoder is fixed, the reconstruction performance remains stable throughout optimization. In contrast, updating the encoder through STE leads to a clear performance degradation, regardless of whether the decoder is fixed or jointly optimized. This result suggests that the degradation observed in MERVar-AE after introducing STE is not specific to our model, but can also occur in an existing STE-based binary quantizer. We further perform the same experiment on VQ-VAE. We use the same VQ-VAE model and checkpoint as in the CIFAR-10 comparison. Without the commitment loss, updating the encoder through STE while



Figure 7: Comparison between the proposed method and state-of-the-art methods regarding latent space smoothness.

Table 12: Ablation study of BSQ optimization on CIFAR-10. Training and test PSNR (dB) are reported, where Δ denotes the change from initialization to 35K updates.

Method	Metric	Init.	5K	10K	15K	20K	25K	30K	35K	Δ
Fix Dec.	Test	24.88	24.51	24.39	24.59	23.75	24.33	24.05	23.89	-0.99
	Train	24.87	24.42	24.00	24.41	23.92	24.26	23.98	23.55	-1.31
Fix Enc.	Test	24.88	24.95	25.02	25.06	24.58	24.53	25.08	25.14	+0.26
	Train	24.87	24.87	24.87	24.87	24.87	24.87	24.87	24.87	+0.00
STE	Test	24.88	23.19	23.42	23.96	23.81	24.06	24.25	23.81	-1.07
	Train	24.87	24.14	23.87	24.32	24.40	24.03	24.23	23.99	-0.87

Fix Dec.: Fixed decoder parameters; Fix Enc.: Fixed encoder parameters;
Init.: Initialization checkpoint;

fixing the decoder causes severe degradation, with PSNR decreasing from 24.63 to approximately 12.44. In contrast, fixing the encoder preserves the reconstruction performance at approximately 24.7. Interestingly, when the commitment loss is retained, this degradation largely disappears and all settings remain stable. This suggests that the commitment loss provides an additional continuous optimization constraint that stabilizes the encoder under surrogate-gradient optimization. Overall, these results indicate that surrogate-gradient mismatch can affect the optimization of different STE-based quantizers, while additional latent constraints, such as the commitment loss in VQ-VAE, can mitigate this effect.

Table 13: Ablation study of STE-based encoder optimization and commitment regularization in VQ-VAE on CIFAR-10. Training and test PSNR (dB) are reported, where Δ denotes the change from initialization to 7K updates.

Method	Metric	Init.	1K	2K	3K	4K	5K	6K	7K	Δ
Fix Dec. w/o CL	Test	24.63	12.39	12.44	12.44	12.44	12.44	12.44	12.44	-12.20
	Train	24.61	12.40	12.45	12.45	12.45	12.45	12.45	12.45	-12.15
Fix Enc. w/o CL	Test	24.63	24.68	24.69	24.69	24.72	24.72	24.73	24.74	+0.10
	Train	24.61	24.65	24.66	24.65	24.68	24.67	24.69	24.69	+0.09
Fix Dec. w/ CL	Test	24.63	24.68	24.72	24.73	24.75	24.77	24.65	24.68	+0.04
	Train	24.61	24.65	24.68	24.70	24.71	24.73	24.65	24.60	-0.01
Fix Enc. w/ CL	Test	24.63	24.68	24.68	24.68	24.68	24.68	24.68	24.65	+0.02
	Train	24.61	24.65	24.65	24.65	24.65	24.65	24.65	24.68	+0.08
STE	Test	24.63	24.68	24.74	24.68	24.65	24.68	24.69	24.68	+0.05
	Train	24.61	24.65	24.71	24.65	24.68	24.65	24.66	24.65	+0.05

Fix Dec.: Fixed decoder; Fix Enc.: Fixed encoder; w/o CL: without commitment loss; w/ CL: with commitment loss; Init.: Initialization checkpoint.

A.3.7 Downstream Evaluation on Image Generation

To evaluate the applicability of the learned discrete representations to downstream generative modeling, we conduct a controlled autoregressive generation experiment on CIFAR-10, as reported in Tab. 14. For each discrete representation learning method, we directly use the three independently trained autoencoder checkpoints obtained from the CIFAR-10 reconstruction experiments, without additional fine-tuning. Each checkpoint is used to encode the CIFAR-10 images into its corresponding discrete latent representations, on which an independent standard autoregressive Transformer is trained. Thus, each method is evaluated using three independent autoencoder–Transformer pipelines, and we report the mean and standard deviation across the three runs. For all methods, the discrete representations are organized as sequences of 64 tokens with the same effective vocabulary size of 2^{10}.

To isolate the effect of the learned discrete representations, we use the same Transformer architecture and training configuration for all methods. We adopt a lightweight causal Transformer consisting of two Transformer blocks with causal self-attention and residual connections, with a model dimension of 256, four attention heads, and a total of 2.21M parameters. Each Transformer is trained for 10,000 optimization updates using Adam with a learning rate of 3×10^{-4} and a batch size of 512. After training, we autoregressively generate 50,000 images for each run using multinomial sampling from the predicted categorical distributions, and compute FID and IS against the CIFAR-10 training set. The same sampling procedure is used for all methods. The reconstruction PSNR and reconstruction FID (rFID) of the corresponding autoencoders are also reported for reference. This experiment examines whether the learned discrete representations can serve as effective tokens for downstream autoregressive modeling, rather than aiming for direct comparison with existing image generation methods. As shown in Tab. 14, MERVar-AE achieves the lowest rFID and generation FID, together with the highest IS. These results suggest that the representations learned by our MERVar-AE remain effective for downstream autoregressive generation. An interesting phenomenon is observed for VQ-VAE and SQ-VAE. Although SQ-VAE achieves better reconstruction quality, its FID is substantially worse. We hypothesize that the stochastic quantization in SQ-VAE improves codebook utilization at the cost of a more dispersed and uncertain token distribution, thereby increasing the difficulty of learning the latent prior with a Transformer. In addition, token indices do not preserve neighborhood similarity, so transitions between nearby indices do not necessarily correspond to nearby latent semantics. These factors may partially explain the weaker downstream generation performance of SQ-VAE.

A.3.8 Codebook Usage Comparison

As shown in Tab. 15, at a codebook size of 2^{10}, our method achieves the best reconstruction quality, with a PSNR of 25.68, an LPIPS of 0.188, and an SSIM of 0.858, while also fully utilizing the codebook (100.00% usage) with a high code usage entropy of 9.681, close to the theoretical

Table 14: Downstream autoregressive generation performance on CIFAR-10 using discrete representations learned by different methods. The same Transformer architecture and training protocol are used for all methods.

Method	Tokens	Reconstruction		Generation	
		PSNR↑	rFID↓	FID↓	IS↑
Transformer + VQ-VAE	64	24.57 ± 0.17	56.52 ± 0.42	80.62 ± 1.85	4.33 ± 0.34
Transformer + VP-VAE	64	22.57 ± 0.15	67.92 ± 2.76	93.96 ± 5.24	3.38 ± 0.31
Transformer + SQ-VAE	64	24.78 ± 0.03	39.67 ± 0.39	172.59 ± 6.92	2.39 ± 0.05
Transformer + FSQ	64	23.45 ± 0.01	71.85 ± 0.67	100.27 ± 3.57	3.84 ± 0.09
Transformer + BSQ	64	24.45 ± 0.31	51.80 ± 0.78	92.04 ± 1.29	4.03 ± 0.02
Transformer + LFQ	64	23.75 ± 0.30	61.37 ± 1.46	90.51 ± 1.47	3.99 ± 0.10
Transformer + Gumbel-Softmax	64	23.38 ± 0.08	70.77 ± 2.47	100.64 ± 2.45	3.33 ± 0.23
Transformer + TokenBridge	64	21.04 ± 0.03	162.75 ± 7.47	189.75 ± 11.40	2.77 ± 0.05
Transformer + LooC	64	25.02 ± 0.00	53.69 ± 0.71	83.34 ± 0.39	4.23 ± 0.03
Transformer + Ours	64	**25.68** ± 0.09	**39.43** ± 0.46	**73.05** ± 0.42	**4.59** ± 0.04

Table 15: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on CIFAR-10.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	30.47% ± 6.03%	8.047 ± 0.265	24.57 ± 0.17	0.241 ± 0.008	0.819 ± 0.007
VP-VAE	2^{10}	87.70% ± 2.25%	8.118 ± 0.074	22.57 ± 0.15	0.294 ± 0.009	0.750 ± 0.007
SQ-VAE	2^{10}	99.93% ± 0.06%	9.727 ± 0.021	24.78 ± 0.03	**0.185** ± 0.001	0.855 ± 0.002
FSQ	2^{10}	23.01% ± 0.55%	7.163 ± 0.061	23.45 ± 0.01	0.268 ± 0.003	0.790 ± 0.002
BSQ	2^{10}	99.48% ± 0.82%	9.215 ± 0.019	24.45 ± 0.31	0.231 ± 0.004	0.819 ± 0.006
LFQ	2^{10}	78.32% ± 13.12%	8.280 ± 0.414	23.75 ± 0.30	0.259 ± 0.008	0.796 ± 0.010
Gumbel.	2^{10}	16.21% ± 0.55%	7.013 ± 0.074	23.38 ± 0.08	0.285 ± 0.007	0.784 ± 0.006
TB.	2^{10}	99.90% ± 0.00%	**9.966** ± 0.029	21.04 ± 0.03	0.435 ± 0.003	0.615 ± 0.001
LooC	2^{10}	79.27% ± 0.74%	8.623 ± 0.016	25.02 ± 0.00	0.220 ± 0.001	0.836 ± 0.000
Ours	2^{10}	**100.00%** ± 0.00%	9.681 ± 0.051	**25.68** ± 0.09	0.188 ± 0.004	**0.858** ± 0.003

Gumbel.: Gumbel-Softmax TB.: TokenBridge

maximum of 10. LooC and VQ-VAE are the closest competitors in PSNR, achieving 25.02 and 24.57, respectively, but exhibit lower reconstruction quality and codebook utilization. In contrast, FSQ and Gumbel-Softmax show relatively low codebook usage of 23.01% and 16.21%, respectively.

BSQ achieves near-full codebook utilization (99.48%) and a high entropy of 9.215, but its reconstruction performance remains lower than ours (PSNR 24.45 vs. 25.68). Similarly, VP-VAE and LFQ achieve relatively high codebook usage (87.70% and 78.32%, respectively), but their PSNR and LPIPS remain clearly worse than ours. TokenBridge attains near-full codebook usage (99.90%) and a high entropy of 9.966, yet exhibits the worst LPIPS (0.435) among all compared methods, indicating that high codebook utilization or entropy alone does not guarantee reconstruction fidelity. SQ-VAE achieves near-full codebook utilization (99.93%) and a high entropy of 9.727, with slightly better LPIPS than ours (0.185 vs. 0.188), but lower PSNR (24.78 vs. 25.68) and SSIM (0.855 vs. 0.858). Previous methods (e.g., VQ-VAE, BSQ, LFQ, and FSQ) rely on a Straight-Through Estimator (STE) to handle their non-differentiable quantization operations, such as nearest-neighbor lookup, sign, or rounding. This mismatch between the forward and backward computations introduces a surrogate gradient. In contrast, our method avoids STE entirely, such that the gradient during continuous training is the exact gradient of the corresponding forward computation rather than a surrogate approximation. SQ-VAE also avoids STE, but uses a Gumbel-Softmax relaxation with a learned codebook. Under the same encoder-decoder architecture and experimental setting, our method achieves full codebook utilization, the highest PSNR and SSIM, and competitive LPIPS among the compared approaches.

Table 16: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on ImageNet.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	39.39% ± 4.67%	8.052 ± 0.009	27.08 ± 0.03	0.294 ± 0.000	0.805 ± 0.001
VP-VAE	2^{10}	99.67% ± 0.20%	8.147 ± 0.143	24.61 ± 0.28	0.383 ± 0.009	0.730 ± 0.013
SQ-VAE	2^{10}	**100.00%** ± 0.00%	9.550 ± 0.032	25.78 ± 0.50	0.294 ± 0.011	0.795 ± 0.020
FSQ	2^{10}	23.47% ± 0.30%	7.019 ± 0.029	25.21 ± 0.45	0.358 ± 0.016	0.759 ± 0.013
BSQ	2^{10}	99.51% ± 0.68%	8.337 ± 0.175	24.65 ± 0.27	0.376 ± 0.011	0.747 ± 0.008
LFQ	2^{10}	64.68% ± 17.02%	7.064 ± 0.915	25.71 ± 0.39	0.341 ± 0.017	0.775 ± 0.011
Gumbel.	2^{10}	19.76% ± 2.15%	6.957 ± 0.123	25.48 ± 0.42	0.359 ± 0.018	0.752 ± 0.019
TB.	2^{10}	99.90% ± 0.00%	**9.898** ± 0.055	23.39 ± 0.21	0.532 ± 0.112	0.659 ± 0.005
LooC	2^{10}	24.61% ± 0.00%	7.349 ± 0.011	26.73 ± 0.02	0.310 ± 0.000	0.796 ± 0.000
Ours	2^{10}	**100.00%** ± 0.00%	9.189 ± 0.064	**27.66** ± 0.05	**0.262** ± 0.004	**0.826** ± 0.002

Gumbel.: Gumbel-Softmax TB.: TokenBridge

Table 17: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on COCO2017.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	35.38% ± 0.30%	8.016 ± 0.016	28.80 ± 0.04	0.241 ± 0.002	0.859 ± 0.001
VP-VAE	2^{10}	97.53% ± 0.06%	7.796 ± 0.000	25.02 ± 0.00	0.372 ± 0.000	0.756 ± 0.000
SQ-VAE	2^{10}	**100.00%** ± 0.00%	9.497 ± 0.009	27.89 ± 0.08	0.242 ± 0.002	0.860 ± 0.001
FSQ	2^{10}	23.11% ± 0.55%	6.770 ± 0.090	25.86 ± 0.41	0.338 ± 0.013	0.796 ± 0.010
BSQ	2^{10}	98.21% ± 1.82%	8.401 ± 0.272	26.35 ± 0.82	0.321 ± 0.029	0.814 ± 0.024
LFQ	2^{10}	53.81% ± 19.86%	6.919 ± 0.871	26.79 ± 0.48	0.305 ± 0.017	0.820 ± 0.010
Gumbel.	2^{10}	25.58% ± 4.45%	7.381 ± 0.176	27.44 ± 0.16	0.293 ± 0.004	0.826 ± 0.005
TB.	2^{10}	99.90% ± 0.00%	**9.905** ± 0.057	24.62 ± 0.21	0.430 ± 0.002	0.716 ± 0.003
LooC	2^{10}	82.68% ± 0.48%	7.612 ± 0.018	27.79 ± 0.02	0.293 ± 0.001	0.829 ± 0.001
Ours	2^{10}	**100.00%** ± 0.00%	9.086 ± 0.066	**29.07** ± 0.05	**0.231** ± 0.004	**0.870** ± 0.002

Gumbel.: Gumbel-Softmax TB.: TokenBridge

A.3.9 Comparison with Strong Pre-Trained Quantization Model

We compare MERVar-AE with the strong pre-trained autoencoder from LDM using the same codebook size of 2^{13}. Specifically, we use the official LDM $f = 4$, 2^{13}-codebook checkpoint without attention, downloaded from the LDM repository. For this experiment, we adopt the VQGAN autoencoder architecture for MERVar-AE and use a separate training protocol from the unified comparison experiments described above. MERVar-AE is trained for 50K updates on CIFAR-10 with a batch size of 512, followed by 100K updates on ImageNet with a batch size of 32. Adam with a learning rate of 3×10^{-4} is used throughout training. The entire training process takes approximately 12 hours on a single NVIDIA RTX 4080 16 GB GPU. As shown in Tab. 24, despite using substantially fewer parameters (12.96M vs. 53.22M) and a modest training budget, MERVar-AE achieves higher PSNR and SSIM on most evaluated datasets while maintaining competitive LPIPS. These results demonstrate that MERVar-AE can learn an effective discrete representation with substantially lower training cost and model size.

A.3.10 Ablation Experiment for Hard Binarization

We conducted experiments on a subset of ImageNet containing 512 images to evaluate the training strategy, as shown in Fig. 8. From the beginning of training, noise was injected into the latent variables, while a $\tanh$ activation was used to constrain their values to the range $[-1, 1]$. As training progressed, the proportion of latent values approaching either -1 or 1, referred to as the binary rate, steadily increased, while the mean squared error (MSE) consistently decreased. When the binary rate

Table 18: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on CelebA-HQ.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	30.34% ± 5.97%	6.448 ± 0.224	30.25 ± 0.07	0.295 ± 0.008	0.862 ± 0.004
VP-VAE	2^{10}	92.68% ± 0.00%	6.899 ± 0.002	27.06 ± 0.01	0.391 ± 0.000	0.796 ± 0.000
SQ-VAE	2^{10}	**100.00%** ± 0.00%	9.105 ± 0.106	30.88 ± 1.88	**0.186** ± 0.031	0.893 ± 0.033
FSQ	2^{10}	22.82% ± 0.79%	6.636 ± 0.175	29.09 ± 0.45	0.266 ± 0.013	0.870 ± 0.005
BSQ	2^{10}	93.20% ± 5.40%	7.709 ± 0.607	28.99 ± 0.95	0.281 ± 0.043	0.861 ± 0.024
LFQ	2^{10}	56.05% ± 18.06%	6.836 ± 1.013	29.95 ± 0.59	0.236 ± 0.017	0.885 ± 0.008
Gumbel.	2^{10}	20.51% ± 3.86%	6.888 ± 0.217	30.55 ± 0.29	0.240 ± 0.013	0.881 ± 0.007
TB.	2^{10}	99.90% ± 0.00%	**9.815** ± 0.074	27.96 ± 0.83	0.353 ± 0.032	0.821 ± 0.016
LooC	2^{10}	75.52% ± 0.46%	6.408 ± 0.011	29.73 ± 0.11	0.297 ± 0.002	0.864 ± 0.002
Ours	2^{10}	**100.00%** ± 0.00%	8.136 ± 0.088	**31.69** ± 0.07	0.217 ± 0.004	**0.900** ± 0.002

Gumbel.: Gumbel-Softmax TB.: TokenBridge

Table 19: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on LSUN Bedroom (10k).

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	31.77% ± 6.75%	7.102 ± 0.200	28.17 ± 0.13	0.314 ± 0.009	0.838 ± 0.006
VP-VAE	2^{10}	98.70% ± 0.25%	7.770 ± 0.090	25.73 ± 0.26	0.378 ± 0.007	0.779 ± 0.007
SQ-VAE	2^{10}	**100.00%** ± 0.00%	9.287 ± 0.007	**29.93** ± 0.04	**0.208** ± 0.001	**0.897** ± 0.001
FSQ	2^{10}	23.18% ± 0.65%	6.731 ± 0.072	26.63 ± 0.45	0.321 ± 0.015	0.827 ± 0.007
BSQ	2^{10}	98.99% ± 0.81%	8.574 ± 0.176	27.41 ± 0.55	0.297 ± 0.015	0.843 ± 0.011
LFQ	2^{10}	62.60% ± 8.19%	6.353 ± 0.203	26.17 ± 0.39	0.342 ± 0.026	0.811 ± 0.013
Gumbel.	2^{10}	26.37% ± 4.87%	7.193 ± 0.147	27.91 ± 0.15	0.302 ± 0.004	0.834 ± 0.004
TB.	2^{10}	99.90% ± 0.00%	**9.959** ± 0.019	25.04 ± 0.26	0.430 ± 0.002	0.743 ± 0.005
LooC	2^{10}	83.04% ± 0.63%	7.225 ± 0.020	28.25 ± 0.03	0.312 ± 0.001	0.847 ± 0.001
Ours	2^{10}	**100.00%** ± 0.00%	8.601 ± 0.058	29.67 ± 0.07	0.240 ± 0.004	0.882 ± 0.002

Gumbel.: Gumbel-Softmax TB.: TokenBridge

reached 0.90, we enabled hard binarization of the latent variables and simultaneously removed the injected noise. This transition led to a substantial reduction in MSE, primarily due to the elimination of noise perturbations. Moreover, introducing hard binarization had little impact on training stability, suggesting that the decoder had already adapted to latent representations concentrated near the discrete values of -1 and 1 during the preceding noise-injected training stage.

A.3.11 Code Stability Under Input Perturbations

We first examine the distribution of the latent pre-activations $\mathbf{z}^{\mathrm{e}}$ before the $\tanh$ activation. As shown in Tab. 25, the mean absolute magnitude is $\mathbb{E}(|\mathbf{z}^{\mathrm{e}}|) = 4.7059$, indicating that a substantial portion of the latent pre-activations have relatively large magnitudes and are therefore mapped toward the saturated regions of $\tanh$. Meanwhile, the minimum absolute value reaches 1.6689×10^{-6}, indicating that a small population of latent variables remains extremely close to the decision boundary at zero. This motivates a more fine-grained stability analysis that separately examines near-boundary and saturated latent variables.

We further investigate the stability of MERVar-AE by stratifying the latent variables according to their distance from the decision boundary. Specifically, we compute the population ratios of saturated and near-boundary latents for MERVar-AE trained on CIFAR-10 with a 10-dimensional binary latent space, corresponding to 2^{10} possible codes. We also evaluate the bit-flip rate of each latent group under different levels of input perturbation. Specifically, Gaussian noise with standard deviation τ ranging from 0.1 to 1.0 is added directly to the input images, and the resulting binary representations

Table 20: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on Kodak.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	30.27% ± 6.03%	6.981 ± 0.212	26.96 ± 0.18	0.376 ± 0.009	0.885 ± 0.008
VP-VAE	2^{10}	82.62% ± 4.56%	6.969 ± 0.122	23.74 ± 0.47	0.455 ± 0.004	0.799 ± 0.002
SQ-VAE	2^{10}	99.93% ± 0.06%	8.459 ± 0.036	26.97 ± 0.10	0.351 ± 0.005	0.882 ± 0.003
FSQ	2^{10}	22.85% ± 0.80%	6.533 ± 0.318	24.85 ± 0.84	0.443 ± 0.036	0.850 ± 0.017
BSQ	2^{10}	62.86% ± 9.46%	8.337 ± 0.471	21.09 ± 1.05	0.383 ± 0.053	0.724 ± 0.052
LFQ	2^{10}	79.75% ± 15.37%	7.941 ± 0.272	27.31 ± 0.74	0.358 ± 0.023	0.890 ± 0.011
Gumbel.	2^{10}	15.17% ± 1.50%	5.680 ± 0.482	24.00 ± 2.16	0.456 ± 0.036	0.820 ± 0.035
TB.	2^{10}	99.90% ± 0.00%	**9.754** ± 0.106	22.53 ± 0.68	0.539 ± 0.014	0.767 ± 0.017
LooC	2^{10}	74.64% ± 0.23%	6.884 ± 0.117	26.91 ± 0.04	0.377 ± 0.001	0.885 ± 0.001
Ours	2^{10}	**100.00%** ± 0.00%	8.731 ± 0.066	**28.67** ± 0.05	**0.300** ± 0.002	**0.919** ± 0.001

Gumbel.: Gumbel-Softmax TB.: TokenBridge

Table 21: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on FFHQ Thumbnail.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	31.97% ± 7.11%	7.266 ± 0.249	27.24 ± 0.12	0.261 ± 0.009	0.836 ± 0.006
VP-VAE	2^{10}	99.09% ± 0.25%	8.039 ± 0.077	24.81 ± 0.09	0.325 ± 0.006	0.771 ± 0.005
SQ-VAE	2^{10}	**100.00%** ± 0.00%	9.689 ± 0.060	24.56 ± 0.63	0.225 ± 0.026	0.835 ± 0.026
FSQ	2^{10}	23.44% ± 0.35%	6.835 ± 0.223	25.83 ± 0.50	0.280 ± 0.021	0.817 ± 0.015
BSQ	2^{10}	98.50% ± 1.29%	8.725 ± 0.196	26.29 ± 0.56	0.253 ± 0.021	0.834 ± 0.014
LFQ	2^{10}	87.24% ± 12.80%	7.801 ± 0.223	26.42 ± 0.57	0.255 ± 0.025	0.836 ± 0.008
Gumbel.	2^{10}	19.86% ± 3.09%	6.767 ± 0.191	26.36 ± 0.19	0.271 ± 0.007	0.819 ± 0.005
TB.	2^{10}	99.90% ± 0.00%	**9.898** ± 0.057	23.94 ± 0.18	0.431 ± 0.010	0.730 ± 0.005
LooC	2^{10}	83.82% ± 0.15%	7.516 ± 0.018	27.42 ± 0.04	0.249 ± 0.000	0.844 ± 0.001
Ours	2^{10}	**100.00%** ± 0.00%	8.950 ± 0.059	**28.57** ± 0.08	**0.190** ± 0.005	**0.875** ± 0.003

Gumbel.: Gumbel-Softmax TB.: TokenBridge

are compared with those obtained from the original inputs. The results are reported in Tab. 26. Note that the latent grouping here is computed on $\mathbf{z}_i$, which is the output of the Tanh activation, while the reported binary representations are obtained after hard binarization. As expected, latents closer to the decision boundary are more susceptible to bit flips. For example, at $\tau = 0.1$, the bit-flip rate reaches 47.4% for latents satisfying $|\mathbf{z}| < 0.1$. However, near-boundary latents constitute only a small fraction of the overall representation: only 1.7%, 3.4%, and 5.2% of the latents satisfy $|\mathbf{z}| < 0.1$, $|\mathbf{z}| < 0.2$, and $|\mathbf{z}| < 0.3$, respectively. Consequently, the overall bit-flip rate remains 8.4% at $\tau = 0.1$. Despite these bit flips, the reconstruction quality is only mildly affected, with PSNR decreasing from 25.68 dB to 25.33 dB and SSIM from 0.858 to 0.849. These results indicate that although near-boundary latents are locally more sensitive to input perturbations, their small population limits their impact on the stability and reconstruction quality of the overall representation.

A.3.12 Code Stability Under Latent Perturbations

We evaluate code stability by directly perturbing the latent representations. After training, Gaussian noise with standard deviation $\tau \in [0, 1]$ is added directly to the tanh outputs $\mathbf{z}_i$, after which hard binarization is applied, using CIFAR-10 test images. We compare the resulting binary codes with those obtained without perturbation and report the average Hamming distance, average maximum Hamming distance, and reconstruction quality in Tab. 27. The average maximum Hamming distance is obtained by first computing the maximum Hamming distance within each image and then averaging it over the test set. We conduct the same experiment for BSQ under an identical experimental

Table 22: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on Fashion-MNIST.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	21.12% ± 3.53%	6.964 ± 0.411	25.10 ± 0.04	0.080 ± 0.004	0.866 ± 0.004
VP-VAE	2^{10}	96.09% ± 0.93%	7.707 ± 0.166	20.52 ± 0.40	0.175 ± 0.017	0.705 ± 0.021
SQ-VAE	2^{10}	**100.00%** ± 0.00%	9.409 ± 0.052	26.22 ± 0.52	**0.047** ± 0.003	**0.910** ± 0.006
FSQ	2^{10}	22.85% ± 0.68%	6.562 ± 0.239	20.55 ± 0.31	0.148 ± 0.004	0.727 ± 0.012
BSQ	2^{10}	**100.00%** ± 0.00%	9.529 ± 0.042	23.69 ± 0.25	0.086 ± 0.005	0.843 ± 0.007
LFQ	2^{10}	72.88% ± 8.83%	7.467 ± 0.571	22.03 ± 0.50	0.109 ± 0.007	0.794 ± 0.019
Gumbel.	2^{10}	18.95% ± 1.19%	6.581 ± 0.053	23.13 ± 0.10	0.092 ± 0.001	0.830 ± 0.002
TB.	2^{10}	99.84% ± 0.11%	9.623 ± 0.024	19.48 ± 0.15	0.222 ± 0.017	0.633 ± 0.020
LooC	2^{10}	85.42% ± 4.59%	6.510 ± 0.239	24.45 ± 0.17	0.074 ± 0.003	0.861 ± 0.005
Ours	2^{10}	**100.00%** ± 0.00%	**9.625** ± 0.011	**26.60** ± 0.08	0.049 ± 0.000	0.909 ± 0.001

Gumbel.: Gumbel-Softmax TB.: TokenBridge

Table 23: Comparison of codebook usage, code usage entropy, and reconstruction performance of different discrete representation learning methods on MNIST.

Method	Codebook Size	Codebook Usage↑	Code Usage Entropy↑	PSNR↑	LPIPS↓	SSIM↑
VQ-VAE	2^{10}	7.00% ± 1.83%	5.036 ± 0.508	27.90 ± 0.47	0.022 ± 0.002	0.958 ± 0.008
VP-VAE	2^{10}	76.66% ± 1.19%	6.030 ± 0.166	21.08 ± 0.06	0.084 ± 0.003	0.841 ± 0.018
SQ-VAE	2^{10}	94.37% ± 0.90%	8.929 ± 0.022	28.38 ± 0.36	0.011 ± 0.001	0.980 ± 0.003
FSQ	2^{10}	21.88% ± 1.20%	5.373 ± 0.586	21.60 ± 0.14	0.041 ± 0.002	0.899 ± 0.006
BSQ	2^{10}	**100.00%** ± 0.00%	**9.606** ± 0.038	25.97 ± 0.21	0.019 ± 0.001	0.966 ± 0.002
LFQ	2^{10}	74.96% ± 14.00%	7.461 ± 0.648	24.08 ± 0.71	0.028 ± 0.004	0.945 ± 0.007
Gumbel.	2^{10}	17.61% ± 0.32%	6.161 ± 0.249	25.45 ± 0.47	0.021 ± 0.002	0.963 ± 0.004
TB.	2^{10}	99.15% ± 0.72%	8.989 ± 0.026	20.43 ± 0.16	0.116 ± 0.010	0.800 ± 0.017
LooC	2^{10}	33.98% ± 3.14%	6.486 ± 0.083	27.99 ± 0.29	0.016 ± 0.000	0.976 ± 0.001
Ours	2^{10}	**100.00%** ± 0.00%	9.084 ± 0.100	**29.75** ± 0.30	**0.010** ± 0.001	**0.987** ± 0.001

Gumbel.: Gumbel-Softmax TB.: TokenBridge

setting. Interestingly, MERVar-AE and BSQ exhibit nearly identical Hamming distances across all perturbation levels, indicating comparable stability of their binary codes against direct latent perturbations. However, MERVar-AE consistently preserves substantially better reconstruction quality as the perturbation strength increases. For example, at $\tau = 0.6$, the average Hamming distances are 0.479 and 0.478 for MERVar-AE and BSQ, respectively, while MERVar-AE achieves a PSNR of 22.97 dB compared with 19.59 dB for BSQ. Similarly, MERVar-AE obtains higher SSIM (0.794 vs. 0.695) and lower LPIPS (0.237 vs. 0.317). These results suggest that, despite comparable bit-flip behavior, the binary representations learned by MERVar-AE are more robust in terms of reconstruction quality when latent perturbations induce code changes.

A.3.13 Qualitative Evaluation

The qualitative evaluation of the proposed approach is shown in Fig. 9. The proposed approach produces high-quality binary latents for images with high or low resolutions. For visualization, we set the latent dimension to $d = 1$ while keeping the spatial latent size unchanged; these additional experiments are separate from the quantitative results reported above. The proposed approach produces high-quality binary latents for images with high or low resolutions. In addition, more demonstrations of the smoothness of the latent space learned by the proposed approach are shown in Fig. 10, Fig. 11, and Fig. 12.

Table 24: Comparison between the proposed MERVar-AE and LDM across nine datasets.

Method	Codebook Size	Params (M)	CIFAR-10			ImageNet		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LDM	2^{13}	53.22M	17.074	0.6120	0.1974	25.844	0.7779	0.1068
Ours	2^{13}	12.96M	**25.281**	**0.8778**	**0.1178**	**28.578**	**0.8805**	**0.1004**

Method	Codebook Size	Params (M)	COCO2017			CelebA-HQ		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LDM	2^{13}	53.22M	27.636	0.8424	**0.0887**	31.916	0.9002	**0.0594**
Ours	2^{13}	12.96M	**29.906**	**0.9141**	0.0991	**33.200**	**0.9468**	0.0655

Method	Codebook Size	Params (M)	LSUN Bedroom (10k)			FFHQ Thumbnail		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LDM	2^{13}	53.22M	28.528	0.8712	**0.0767**	26.423	0.8385	0.0761
Ours	2^{13}	12.96M	**30.528**	**0.9264**	0.0878	**29.182**	**0.9183**	**0.0710**

Method	Codebook Size	Params (M)	MNIST			Fashion-MNIST		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LDM	2^{13}	53.22M	22.204	**0.9408**	**0.0259**	18.721	0.7600	0.0865
Ours	2^{13}	12.96M	**22.458**	0.8655	0.0378	**23.344**	**0.8653**	**0.0582**

Method	Codebook Size	Params (M)	Kodak		
			PSNR↑	SSIM↑	LPIPS↓
LDM	2^{13}	53.22M	27.396	0.9191	0.1160
Ours	2^{13}	12.96M	**29.533**	**0.9516**	**0.1159**

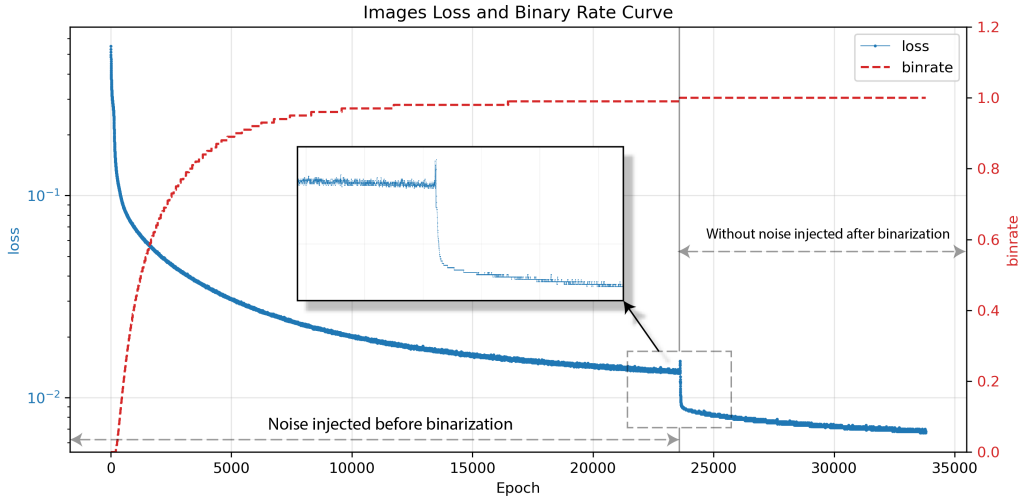

Figure 8: Ablation experiment for hard binarization in the proposed MERVar Autoencoder. During training, the binary rate of latents keeps increasing. When the binarization rate exceeds a threshold (e.g., 90%), hard binarization is applied and noise injection is deactivated, which accounts for the sudden drop in loss.

Table 25: Statistics of the latent pre-activations $\mathbf{z}^{\mathrm{e}}$ before the $\tanh$ activation on CIFAR-10.

$\mathbb{E}(\lvert\mathbf{z}^{\mathrm{e}}\rvert)$	$\mathrm{std}(\lvert\mathbf{z}^{\mathrm{e}}\rvert)$	$\min(\lvert\mathbf{z}^{\mathrm{e}}\rvert)$	$\max(\lvert\mathbf{z}^{\mathrm{e}}\rvert)$	$\mathbb{E}(\mathbf{z}^{\mathrm{e}})$	$\mathrm{std}(\mathbf{z}^{\mathrm{e}})$	$\min(\mathbf{z}^{\mathrm{e}})$	$\max(\mathbf{z}^{\mathrm{e}})$
4.7059	4.4440	1.6689×10^{-6}	78.0336	0.6757	6.4373	-54.3867	78.0336

Table 26: Robustness analysis of latent representations under Gaussian input perturbations on CIFAR-10. Population Ratio denotes the proportion of latent variables belonging to each latent group, while τ denotes the standard deviation of the Gaussian noise added to the input images. Each τ column reports the bit-flip rate of the corresponding latent group.

Latent group	Population Ratio	τ 0.1	τ 0.2	τ 0.3	τ 0.4	τ 0.5	τ 0.6	τ 0.7	τ 0.8	τ 0.9	τ 1.0
$P(\lvert\mathbf{z}\rvert < 0.1)$	0.017	0.474	0.486	0.492	0.495	0.495	0.495	0.497	0.498	0.498	0.499
$P(\lvert\mathbf{z}\rvert < 0.2)$	0.034	0.452	0.475	0.484	0.489	0.491	0.492	0.494	0.496	0.497	0.498
$P(\lvert\mathbf{z}\rvert < 0.3)$	0.052	0.430	0.463	0.475	0.482	0.486	0.489	0.491	0.493	0.495	0.496
$P(\lvert\mathbf{z}\rvert < 0.4)$	0.071	0.408	0.451	0.467	0.476	0.481	0.485	0.488	0.491	0.493	0.494
$P(\lvert\mathbf{z}\rvert < 0.5)$	0.092	0.385	0.439	0.459	0.470	0.477	0.481	0.485	0.489	0.491	0.493
$P(\lvert\mathbf{z}\rvert < 0.6)$	0.116	0.361	0.424	0.449	0.463	0.471	0.477	0.482	0.486	0.489	0.491
$P(\lvert\mathbf{z}\rvert < 0.7)$	0.143	0.336	0.408	0.438	0.454	0.465	0.472	0.478	0.482	0.486	0.489
$P(\lvert\mathbf{z}\rvert < 0.8)$	0.179	0.309	0.391	0.425	0.445	0.458	0.467	0.474	0.479	0.484	0.487
$P(\lvert\mathbf{z}\rvert < 0.9)$	0.232	0.274	0.367	0.409	0.432	0.448	0.459	0.468	0.475	0.480	0.485
All Latents	1.000	0.084	0.156	0.214	0.262	0.301	0.335	0.362	0.386	0.405	0.422
PSNR		25.33	24.01	22.16	20.27	18.54	17.06	15.80	14.72	13.80	13.02
SSIM		0.849	0.803	0.732	0.653	0.578	0.511	0.454	0.403	0.358	0.320

Table 27: Code stability and reconstruction robustness of MERVar-AE and BSQ under Gaussian perturbations applied directly to the latent representations on CIFAR-10. τ denotes the standard deviation of the Gaussian perturbation.

Method	Metric	τ 0	τ 0.1	τ 0.2	τ 0.3	τ 0.4	τ 0.5	τ 0.6	τ 0.7	τ 0.8	τ 0.9	τ 1.0
Ours	$\mathbb{E}$(Ham.)	0	0	0	0.004	0.062	0.228	0.479	0.767	1.058	1.333	1.587
	$\mathbb{E}(\max(\text{Ham.}))$	0	0	0	0.239	1.082	1.825	2.531	3.208	3.764	4.243	4.646
	PSNR	25.68	25.68	25.68	25.72	25.42	24.50	22.97	21.10	19.37	17.94	16.81
	SSIM	0.858	0.858	0.858	0.857	0.854	0.833	0.794	0.738	0.673	0.610	0.552
	LPIPS	0.188	0.188	0.188	0.188	0.190	0.207	0.237	0.277	0.319	0.358	0.391
BSQ	$\mathbb{E}$(Ham.)	0	0	0	0.004	0.062	0.227	0.478	0.766	1.056	1.331	1.585
	$\mathbb{E}(\max(\text{Ham.}))$	0	0	0	0.241	1.083	1.822	2.539	3.208	3.769	4.260	4.664
	PSNR	24.87	24.87	24.87	24.79	23.84	21.77	19.59	17.85	16.52	15.50	14.70
	SSIM	0.833	0.833	0.833	0.832	0.815	0.766	0.695	0.620	0.551	0.490	0.440
	LPIPS	0.220	0.220	0.220	0.221	0.234	0.269	0.317	0.365	0.406	0.439	0.466



Figure 9: The qualitative evaluation of the proposed method on CelebA-HQ, ImageNet and CIFAR-10 datasets.

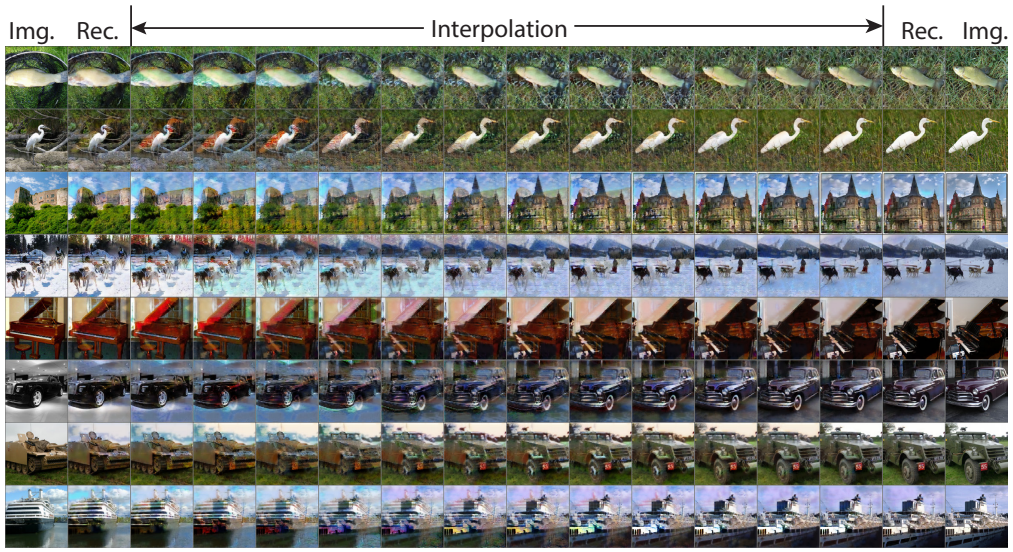

Figure 10: Demonstration of the smoothness of the latent space learned by the proposed approach on the ImageNet dataset.

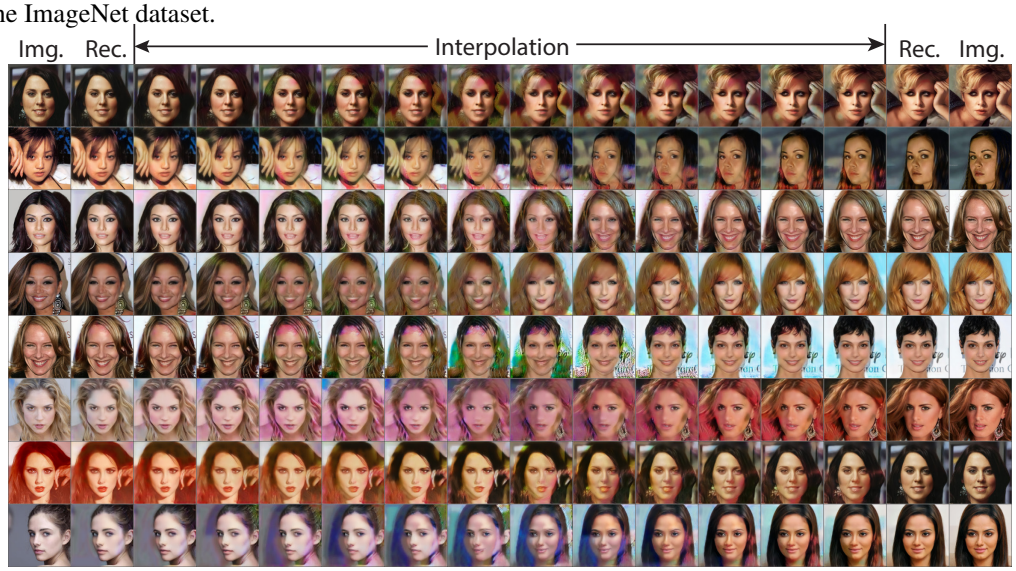

Figure 11: Demonstration of the smoothness of the latent space learned by the proposed approach on the CelebA-HQ dataset.

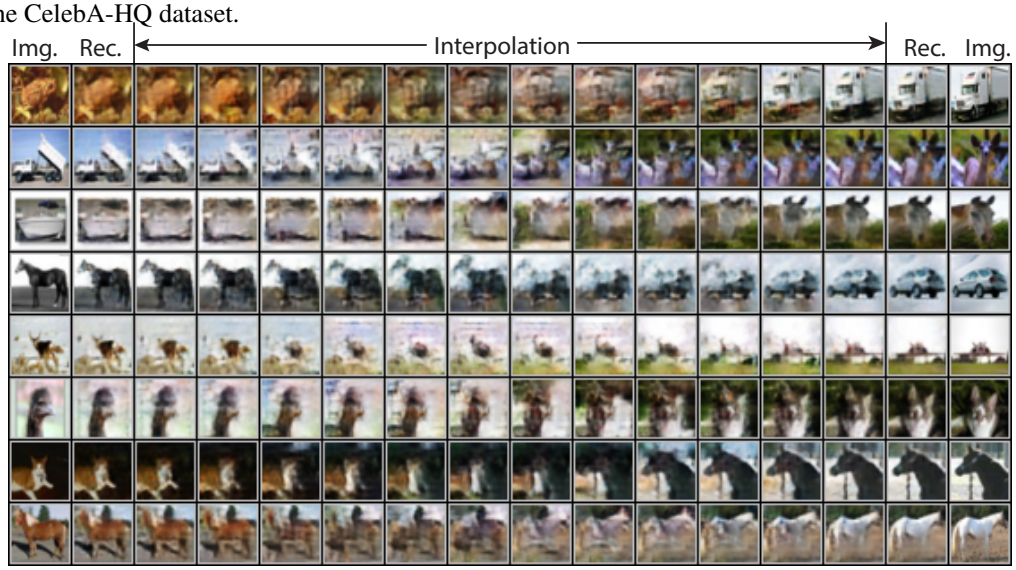

Figure 12: Demonstration of the smoothness of the latent space learned by the proposed approach on the CIFAR-10 dataset.