FLAT-SID: RESAMPLING IMAGE AND TEXT INTO FLEXIBLE-LENGTH ALIGNED SEMANTIC IDS VIA LEARNED STEP QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Representing visual content as discrete 1D tokens enables efficient storage, indexing, and seamless integration with generative models. However, existing 1D visual tokenizers such as TiTok and FlexTok prioritize pixel reconstruction over language alignment and rely on training a downstream model for cross-modal generation tasks. In this work, we introduce FLAT-SID, a discrete representation pre-training framework that jointly learns **F**lexible-**L**ength, **A**ligned, **T**ransmodal semantic ID tokens and their decoders, unifying cross-modal retrieval and generation (T2I and I2T) within a single architecture. A core challenge in end-to-end discrete training is to disentangle representation learning and quantization. We address this by applying quantization-aware training (QAT) to continuous latents. Specifically, we use Learned Step Quantization (LSQ) from the QAT literature, learning adaptive dimension-wise step sizes without the rigid grid constraints of lookup-free vector quantizers like FSQ or LFQ. Building on LSQ, FLAT-SID supports flexible-length semantic ID outputs for sparse retrieval and cross-modal generation, allowing for a tunable trade-off between performance and storage efficiency. The pre-trained FLAT-SID achieves 75.4 GenEval scores and, with lightweight fine-tuning, delivers competitive performance among state-of-the-art sparse methods: T2I generation with 81.2 GenEval score, I2T generation on MSCOCO with 139.4 CIDEr score, and T2I retrieval with Recall@5 70.9 and I2T 83.9 both on MSCOCO. FLAT-SID demonstrates the effectiveness of QAT for discrete semantic representations with a unified contrastive-generative framework.

1 INTRODUCTION

Representing visual content as a compact sequence of tokens has emerged as a core primitive for scalable multimodal systems. Instead of processing images as spatial grids of patches, recent work shows that images can be compressed into a discrete 1D token sequence that is cheap to store, supports exact prefix-based matching and sparse retrieval, and can be integrated into autoregressive models as a foreign vocabulary: TiTok (Yu et al., 2024b) demonstrated that as few as 32 tokens suffice for high-fidelity image reconstruction, and FlexTok (Bachmann et al., 2025) extended this idea to flexible-length representations in which the number of tokens K can be chosen at inference time. This yields a coarse-to-fine ordering: a single token captures the gist, and additional tokens progressively add detail. Such a Matryoshka flexibility enables a controllable rate–quality trade-off and has proven effective for image reconstruction (Bachmann et al., 2025).

A limitation of the above discrete 1D visual tokens is that they are built on a single modality via pixel reconstruction and lack semantic grounding. Moreover, to leverage these pre-trained tokens in cross-modal tasks, a downstream model—such as an autoregressive transformer—must be trained separately, to either translating visual tokens into text or generating them from text. On the other hand, a recent work FLAT (Sun et al., 2026) shows that image-text-aligned continuous tokens can be learned by training a shared representation encoder and decoders jointly with contrastive and generative objectives, enabling transmodal retrieval and generation from the same latent. However, these latents remain continuous and cannot function as compact integer IDs for indexing, de-duplication, LLM vocabulary extension and other sparse applications. This paper addresses that gap by making the representations discrete. We introduce FLAT-SID, a framework that jointly pre-trains **F**lexible-**L**ength, **A**ligned, **T**ransmodal **S**emantic **IDs** and the decoders that consume them (for T2I and I2T). FLAT-SID is trained end-to-end, in a similar architecture as the FLAT continuous tokenizer, but with a quantization bottleneck inserted into the latent space: a low-dimensional quantized code acts as both the semantic ID and the contrastive space, while feeding both decoders for I2T and T2I generation.

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

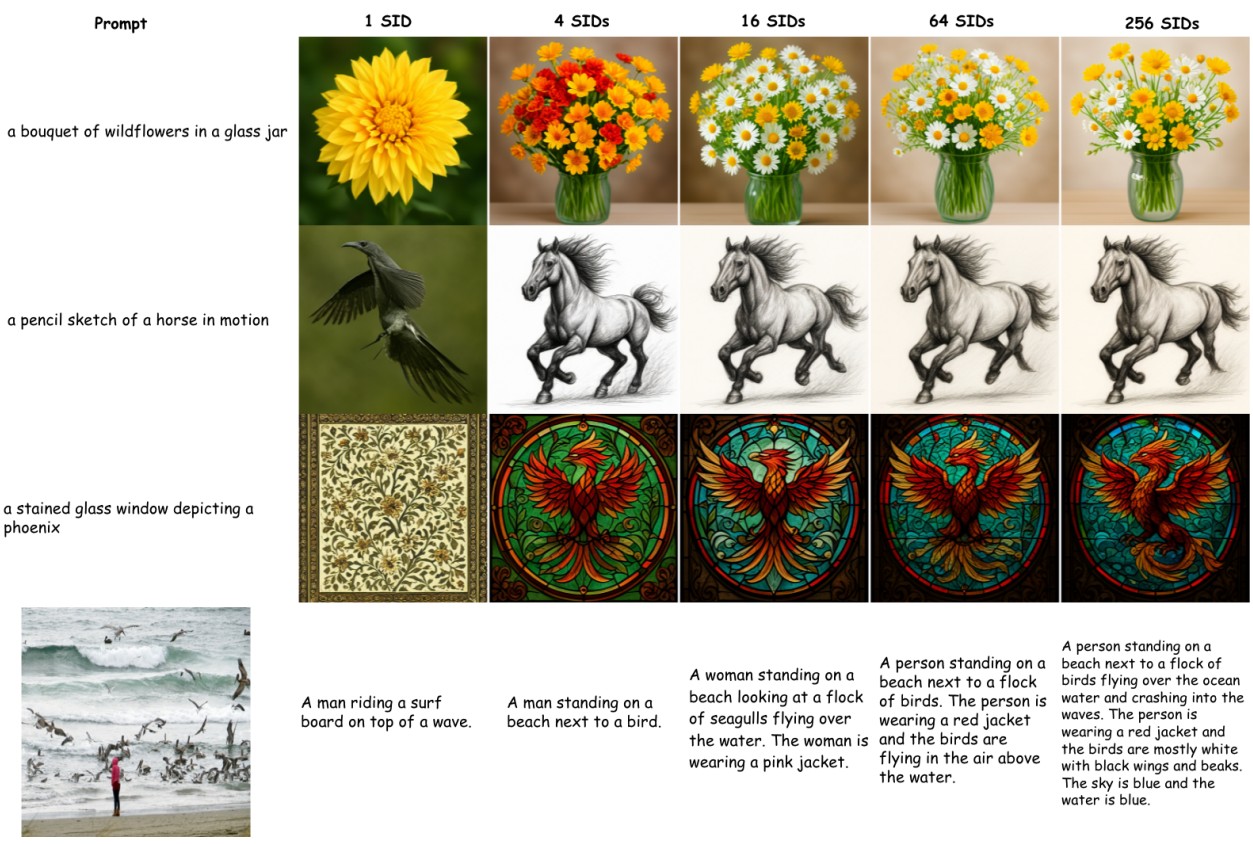


Figure 1: Flexible-length Semantic ID for T2I and I2T generation.

The central challenge in end-to-end discrete tokenizer training is to disentangle representation learning from quantization, ensuring that discretization merely introduces lower-precision representations without distorting latent geometry. Inspired by quantization-aware training (QAT) for model weights, we aim to enable the encoder and decoders to optimize their parameters with direct awareness of the representation quantization. From this perspective, existing lookup-free quantizers operate as QAT variants with overly restrictive grids: LFQ (Yu et al., 2024a) restricts each dimensional value to binary, while FSQ (Mentzer et al., 2023) enforces a bounding non-linearity (e.g., $\tanh$) prior to rounding. Instead, we adapt Learned Step-size Quantization (LSQ)—originally developed for weight QAT—as the discretization mechanism for FLAT-SID. Unlike FSQ, LSQ eliminates bounding non-linearities and optimizes a learned step size per dimension, enabling grid spacing to naturally adapt to the data distribution. We conduct extensive ablations comparing LSQ against established lookup-free quantizers on ImageNet reconstruction. Across all evaluated codebook configurations—varying code dimension and total bits per token—LSQ consistently achieves lower rFID than LFQ and FSQ at equivalent discrete bitrates. Furthermore, by analyzing the trade-off between code dimension and quantization levels per dimension under a fixed codebook size, we observe a U-shaped loss curve balancing compression and quantization error, establishing a practical guideline for designing the quantized latent. Building on LSQ, we construct FLAT-SID, which flexibly resamples text or image inputs into 1 to 256 semantic IDs, enabling a tunable trade-off between performance and storage efficiency. We demonstrate that joint pre-training with contrastive and generative objectives yields semantic IDs capable of addressing both T2I and I2T retrieval and generation. The pre-trained checkpoint achieves 75.4 GenEval score, and lightweight task-specific fine-tuning pushes performance across all tasks into state-of-the-art range: for T2I generation, FLAT-SID achieves 81.2 GenEval score, outperforming prior 1D discrete tokenizers including those with text alignment; for I2T generation, it achieves a 139.4 CIDEr score on MSCOCO, matching or surpassing multiple continuous baselines; and for retrieval, FLAT-SID achieves Recall@5 70.9 (T2I) and 83.9 (I2T) on MS-COCO, and 88.5 (T2I) and 95.6 (I2T) on Flickr30K, using semantic IDs, outperforming previous SOTA sparse retrieval methods. These results demonstrate the effectiveness of pre-training a unified discrete semantic representations with QAT and joint contrastive-generative objectives.

2 Related Work

1D Tokenizers. 1D visual tokenizers replace 2D grids with compact sequences usable by autoregressive or diffusion transformers. TiTok (Yu et al., 2024b) shows 32 **discrete** tokens suffice for reconstruction. FlexTok (Bachmann et al., 2025) makes length flexible via nested dropout, giving coarse-to-fine prefixes $K \in 1..256$. One-D-Piece (Miwa et al., 2025) formulates it as quality-controllable compression with tail-drop and causal ordering. ElasticTok (Yan et al., 2024) extends adaptivity to images and videos. AdaTok (Zhang et al., 2025) allocates adaptively by learned budget. Regarding **text guidance** and semantic alignment, TexTok (Zha et al., 2025) and TA-TiTok (Kim et al., 2025) both make image reconstruction text-aware, conditioning reconstruction on captions. SEED (Ge et al., 2023) maps images to 1D discrete LLM-compatible tokens via Q-Former. Regarding **continuous** variants, ViLex (Wang et al., 2025) learns a continuous visual lexicon in LLM embedding space. MetaQueries (Pan et al., 2025) resamples frozen text encoders into continuous 1D tokens for image generation, and RepTok (Gui et al., 2025) resamples DINOv2 (Oquab et al., 2023) features to highly compressed latent for perception-preserving generation. FLAT-SID sits at the intersection of flexible-length, text-aligned, and discrete semantic representations. Unlike all prior 1D tokenizers that optimize for image reconstruction, FLAT-SID learns transmodal tokens directly, with no reconstruction loss, but via sparse contrastive and cross-modal generation losses. App. A shows a table comparing all 1D tokenizers.

Vector Quantization. Vector quantization was initially popularized by VQ-VAE (van den Oord et al., 2017) which maps encoder features to nearest entries in a **learned codebook**, and extended by VQGAN (Esser et al., 2021) with perceptual and adversarial losses. To increase capacity, residual quantization (Zeghidour et al., 2022; Lee et al., 2022) stack multiple hierarchical codebooks. However, a persistent issue in vector quantization is codebook collapse and low utilization. **Lookup-free** quantizers remove explicit codebook learning. LFQ from MAGVIT-v2 (Yu et al., 2024a) quantizes each dimension to binary values without lookup table. BSQ (Zhao et al., 2024) further normalizes onto the hypersphere before binarization for better utilization. FSQ (Mentzer et al., 2023) replaces vector lookup with scalar quantization: a bounding non-linearity ($\tanh$) followed by rounding to fixed levels per dimension. All above are trained with Straight-Through Estimators (Bengio et al., 2013) and can be viewed as Quantization-Aware Training (QAT) variants on continuous latents but with restricted latent space. On the other hand, QAT was initially developed for model compression to simulate low-precision weights during training. Early methods like PACT (Choi et al., 2018) learn clipping bounds, while LSQ (Esser et al., 2019) learns adaptive scale per dimension, allowing the grid to adapt to the data distribution without a bounding non-linearity. FLAT-SID is built with LSQ applied in vector quantization.

3 Method

FLAT-SID is an image–text aligned semantic tokenizer. Given an image or a text as input, it emits a flexible ordered sequence of 1 to K discrete semantic IDs. Any prefix K is a valid input for transmodal retrieval or generation. Two ingredients make this possible: a quantization bottleneck that turns continuous latent into semantic code (Sec. 3.1), and the FLAT encoder-decoder joint training stack (Sec. 3.2).

3.1 Quantization

3.1.1 Preliminaries: Lookup-Free Vector Quantization

We first review existing work on lookup-free vector quantization. This family avoids an explicit codebook by quantizing each latent dimension independently, so that the implicit codebook is the Cartesian product of the per-dimension level sets.

Notation. Throughout, $z \in \mathbb{R}^d$ denotes a continuous latent produced by a representation encoder at one token position, and $u \in \mathbb{R}^c$ its projection into the low-dimensional code space, with $c < d$. $\hat{u}$ is the quantized version of u. L_i is the number of quantization levels of code dimension i. The compression $z \to u$ keeps the implicit codebook size $|\mathcal{C}| = \prod_{i=1}^{c} L_i$ manageable.

Shared Bottleneck Scaffolding. All quantizers we compare are inserted at the same place and share the same scaffolding. We project z down to u, quantize it to $\hat{u}$, and project back:

$$u = W_{\text{in}} z + b_{\text{in}}, \quad \hat{u} = \mathcal{Q}(u), \quad \hat{z} = W_{\text{out}} \hat{u} + b_{\text{out}}, \tag{1}$$

with $W_{\text{in}} \in \mathbb{R}^{c \times d}$ and $W_{\text{out}} \in \mathbb{R}^{d \times c}$, and $\mathcal{Q}$ the quantizer. Next we compare MAGVIT-v2 LFQ (Yu et al., 2024a) and FSQ (Mentzer et al., 2023), which differ only in the quantized latent computation.

LFQ. MAGVIT-v2 LFQ binarizes each dimensional value u_i. Define $\text{sign}(x) = +1$ for $x \geq 0$ and -1 otherwise. Then

$$\hat{u}_i = \text{sign}(u_i) \in \{-1, +1\} \tag{2}$$

so the quantization grid is fixed to $\{-1, +1\}^c$ and codebook size $|\mathcal{C}| = 2^c$.

FSQ. FSQ assigns L_i uniformly spaced levels to dimension i. Writing the integer bounds and the half-step offset as

$$Q_i^- = -\left\lfloor \frac{L_i}{2} \right\rfloor, \qquad Q_i^+ = \left\lceil \frac{L_i}{2} \right\rceil - 1, \qquad o_i = \begin{cases} 0.5 & L_i \text{ even}, \\ 0 & L_i \text{ odd}, \end{cases} \tag{3}$$

FSQ first bounds and shifts each dimensional value u_i, then rounds it:

$$v_i^{\text{FSQ}} = \text{clamp}\big(\lfloor L_i/2 \rfloor \tanh(u_i) - o_i, Q_i^-, Q_i^+\big), \quad \hat{u}_i = \text{round}\Big(v_i^{\text{FSQ}}\Big) + o_i \tag{4}$$

So the quantization grid and spacing in FSQ are fixed, and $\tanh$ bounds u_i before rounding.

Connection to QAT. Quantization-aware training (QAT) simulates discrete rounding in the forward pass and propagates gradients with a straight-through estimator, so that the surrounding weights learn to compensate for the quantization error. LFQ and FSQ apply exactly this procedure to latents rather than to weights, but both commit to a restrictive, data-independent grid: fixed binary levels in LFQ, and fixed uniform levels behind a saturating nonlinearity in FSQ. This motivates importing the QAT technique that makes the grid itself adaptive and more flexible, i.e. LSQ.

3.1.2 Learned Step Quantization

LSQ keeps the bottleneck of Eq. 1 and replaces the fixed spacing by a learned effective step $s_i > 0$ per code dimension, dropping the bounding nonlinearity altogether. For each coordinate, define

$$v_i = \text{clamp}\big(u_i/s_i - o_i, Q_i^-, Q_i^+\big), \quad \hat{u}_i = (\text{round}(v_i) + o_i)\, s_i \tag{5}$$

where Q_i^-, Q_i^+ and o_i share definition as Eq. 3. So LSQ learns s_i that scales the integer grid to $\hat{u}_i$, thus adaptively setting the grid spacing while keeping it uniform. The following table compares LSQ with LFQ and FSQ. We provide more details and implementation of LSQ in App. B.

Method	Quantized Value ($\hat{u}_i$)	Levels per Dim (L_i)	Implicit Codebook Size	Learnable Grid	Nonlinearity
LFQ	$\text{sign}(u_i)$	2 (Fixed)	2^d	×	×
FSQ	$\text{round}(v_i^{\text{FSQ}}) + o_i$	$L_i \in \mathbb{Z}^+$	$\prod_{i=1}^d L_i$	×	✓ (tanh)
LSQ	$(\text{round}(v_i) + o_i)s_i$	$L_i \in \mathbb{Z}^+$	$\prod_{i=1}^d L_i$	✓ (Step size s_i)	×

3.2 FLAT-SID

3.2.1 Overview of FLAT-SID

FLAT-SID is built upon LSQ and FLAT (Fig. 2). Let x denote an input of modality $m \in \{\text{img}, \text{txt}\}$. FLAT-SID, same as the original FLAT work, appends N learnable register tokens $R = \{r_n\}_{n=1}^N$ to each input and encode the resulting sequence with a shared multimodal encoder f_{enc}. The hidden states at the register positions are projected to obtain the continuous latents z:

$$z = W_{\text{lat}}\, f_{\text{enc}}(x, R) \in \mathbb{R}^{N \times d}, \tag{6}$$

where d is the dimension of the continuous latent and W_{lat} the latent projection.

Different from the original FLAT work, FLAT-SID applies the quantization bottleneck defined in Eq. 1 upon the continuous latent z, producing the quantized latent $\hat{u}$ in lower dimension and the reconstructed latent $\hat{z}$ in the original dimension. Each quantized latent $\hat{u}$ is deterministically mappable to an integer

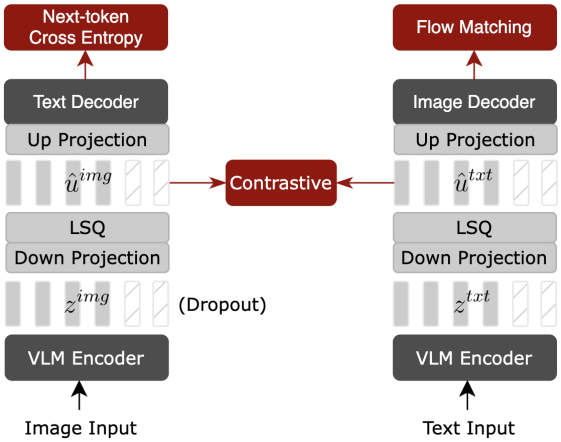


Figure 2: An overview of Flat-SID end-to-end pre-training architecture.

ID and $\hat{z}$ is treated as the semantic embedding of that ID obtained in a look-up free manner during training. The semantic embedding $\hat{z}$ is then used as soft tokens in two cross-modality decoders, for T2I and I2T respectively. Before presenting training details, we first introduce how a semantic integer ID can be deterministically obtained from a quantized latent $\hat{u}$ after training.

3.2.2 How Semantic IDs are Obtained

The demand for converting quantized latent to integer ID is for post-training serialization, indexing and downstream use. In order to do that, for each register and each dimension i, the quantized latent $\hat{u}_i$ can be mapped deterministically to an integer semantic ID a_i following the reverse of Eq. 5:

$$a_i = \frac{\hat{u}_i}{s_i} - o_i - Q_i^- \in \{0, \dots, L_i - 1\}. \tag{7}$$

The semantic ID associated with the quantized latent $\hat{u}$ is then obtained by mixed-radix encoding: $\text{SID} = \sum_{i=1}^{c} a_i \prod_{j=1}^{i-1} L_j$ where L_j is the total number of LSQ levels in dim j. The value of each SID a_i is between $\{0, \dots, L_i - 1\}$, and $\hat{z}_i$ represents its learned embedding entry. Next we present the FLAT-SID training process, which directly uses the look-up free representation $\hat{z}$ as soft tokens in decoders.

3.2.3 Unified Training

As mentioned earlier, FLAT-SID jointly optimizes the representation encoder, quantizer, and two cross-modal decoders. The framework incorporates a contrastive loss to align representations across modalities alongside two generative losses for cross-modal synthesis.

Contrastive Loss. To align image and text representations, we contrast corresponding register positions in the quantized latent $\hat{u}$. For an image–text pair (i, j) we define the similarity computed over N register positions as a late-interaction score between registers at matching positions:

$$s_{ij}^{(N)} = \frac{1}{N} \sum_{n=1}^{N} \frac{(\hat{u}_{i,n}^{\text{img}})^\top \hat{u}_{j,n}^{\text{txt}}}{\|\hat{u}_{i,n}^{\text{img}}\|_2 \, \|\hat{u}_{j,n}^{\text{txt}}\|_2}, \tag{8}$$

and the contrastive loss $\mathcal{L}_{\text{align}}^{(N)}$ as

$$\mathcal{L}_{\text{i2t}}^{(N)} = -\log \frac{\exp\left(s_{i,i}^{(N)}\right)}{\sum_{j=1}^{B_{\text{glob}}} \exp\left(s_{i,j}^{(N)}\right)} \quad \mathcal{L}_{\text{t2i}}^{(N)} = -\log \frac{\exp\left(s_{i,i}^{(N)}\right)}{\sum_{j=1}^{B_{\text{glob}}} \exp\left(s_{j,i}^{(N)}\right)} \quad \mathcal{L}_{\text{align}}^{(N)} = \frac{1}{2}\left(\mathcal{L}_{\text{i2t}}^{(N)} + \mathcal{L}_{\text{t2i}}^{(N)}\right) \tag{9}$$

where B_{glob} denotes the gathered global candidates from all ranks.

Generation Loss. To make image and text representations useful for generations, we apply two generation losses for I2T and T2I respectively, using $\hat{z}^{img}$ and $\hat{z}^{txt}$ as semantic conditions.

For I2T generation and an image input, the embedding $\hat{z}^{img}$ at N register positions are feed to a language model to predict caption y (length t). Each $\hat{z}^{img}$ is projected to the language model input space with a projector followed by RMSNorm. Loss is $\mathcal{L}_{\text{txt}}^{(N)} = -\sum_t \log p_\theta\Big(y_t \,\Big|\, y_{<t}, \hat{z}_{1:N}^{\text{img}}\Big)$.

For T2I generation and a text input, the embedding $\hat{z}^{txt}$ at N register positions are used as semantic conditions for a flow-matching decoder. Let x be the clean VAE image latent, noise $\epsilon \sim \mathcal{N}(0, I)$, and time $t \in [0, 1]$. The noised VAE latent is defined as $x_t = (1 - t)x + t\epsilon$, and the conditional flow-matching objective is $\mathcal{L}_{\text{img}}^{(N)} = \mathbb{E}_{x,\epsilon,t}\Big[\|v_\phi(x_t, t, \hat{z}_{1:N}^{\text{txt}}) - (\epsilon - x)\|_2^2\Big]$ where v is the velocity. Together, the two generative objectives make the FLAT-SID transmodal: image latent supports text generation, while text latent supports image generation.

Optimization with Nested Dropout. To summarize, we optimize the representation encoder, quantization bottleneck and two decoders jointly with the following loss:

$$\mathcal{L} = \lambda_{\text{align}} \mathcal{L}_{\text{align}}^{(N)} + \lambda_{\text{txt}} \mathcal{L}_{\text{txt}}^{(N)} + \lambda_{\text{img}} \mathcal{L}_{\text{img}}^{(N)} \tag{10}$$

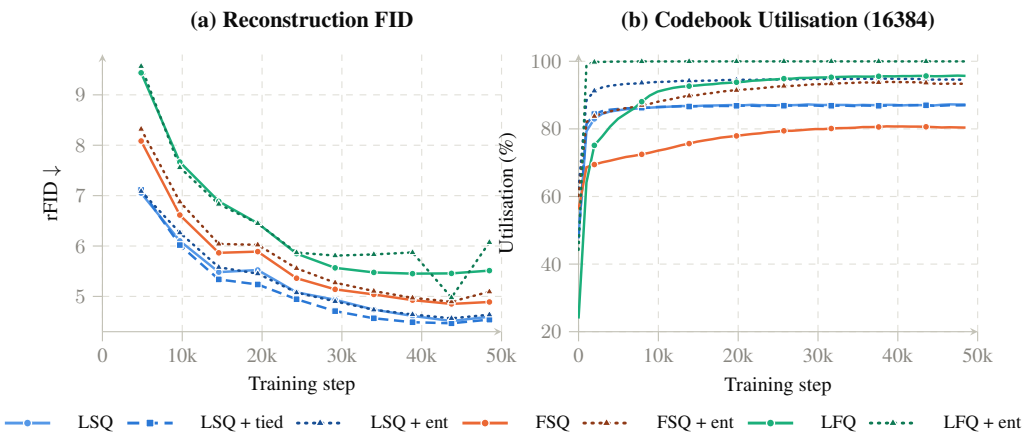


Figure 3: Comparison of various look-up free quantizers at codebook size 16,384 on ImageNet-1K training process.

To enable Matryoshka property where any prefix K instead of a fixed N tokens can be used for retrieval and generation, FLAT-SID adops nested dropout (Bachmann et al., 2025). For each training batch we sample a prefix length K uniformly from a predefined list of training $K \in [1, 4, 16, 64, 256]$, and use the same prefix for all three task objectives. The same sampled K is applied across all ranks. This nested-dropout strategy forces earlier tokens to carry more globally important semantic information, while later tokens progressively refine the details.

4 Experiments

4.1 Quantization Study

Before building the FLAT-SID model, we first perform a comprehensive study to compare LFQ, FSQ, and LSQ variants on the simpler ImageNet-1K reconstruction task, using the standard 1.28M training data, 50K validation, and 256x256 resolution. For all variants, we initialize with the pretrained VAE (Black Forest Labs et al., 2025), which accelerates convergence and isolates the effect of the quantization bottleneck. We provide a comprehensive training config in App. D.1 for reproducibility.

Model Variants. Besides basic LFQ, FSQ and LSQ, we compare two variants on top of them. The first variant is to an entropy loss adopted by MAGVIT-v2 LFQ (Yu et al., 2024a), which is intended to prevent representational collapse and improve utilization of the implicit code space: $\mathcal{L}_{\text{ent}} = \mathbb{E}_i[H(p(\hat{u}_i))] - H(\mathbb{E}_i[p(\hat{u}_i)])$, where $p(\hat{u}_i)$ denotes the code-assignment distribution. The variant is applied to all 3 quantizers, denoted by LFQ+ent, FSQ+ent and LSQ+ent.

The second variant is for LSQ only. We experiment with tied weights for the down-projection $z \to u$ and up projection $\hat{u} \to \hat{z}$ (W_{in} and W_{out} in Eq. 1). This design choice halves the bottleneck parameter, and cleanly separates out compression loss and quantization loss, as explained in App. C. This variant is denoted by LSQ+tied.

Comparisons. We compare all 7 variants over 3 different codebook size 8192, 16384, 65536, corresponding to 13, 14, and 16 bits per token. This leads to 21 training runs. This design isolates the effect of the quantization operator from that of codebook capacity. For LFQ, it is straightforward to get the codebook under total size budget since each dimension has strictly 2 levels; for FSQ and LSQ, we use a fixed grid shape: [8,8,8,4,4] for 8192, [8,8,8,8,4] for 16384, [16,8,8,8,8] for 65536. For the choice of different grid shape within the same total size budget, we leave it to Sec. 4.2.

Conclusions. Fig. 3 shows the validation rFID and codebook utilization during training. The validation rFID is measured on 10k subset for the purpose of choosing the best checkpoint for final evaluation. The codebook utilization is measured on training batches using 200-step EMA. The plots for lower bits (8192 codebook) and higher bits (65536 codebook) experiments are shown in App. D.2. Fig. 4 shows the rFID for all variants and codebook configurations on the full 50K validation sets. We draw the following conclusions: 1) LSQ variants consistently exhibit lower rFID than other quantizers. 2) Entropy loss improves the implicit codebook utilization; however, high codebook utilization does not lead to lower rFID. 3) Tying the up and down projection layers of LSQ does not impact the rFID. In fact, it achieves slightly better rFID that untied LSQ.

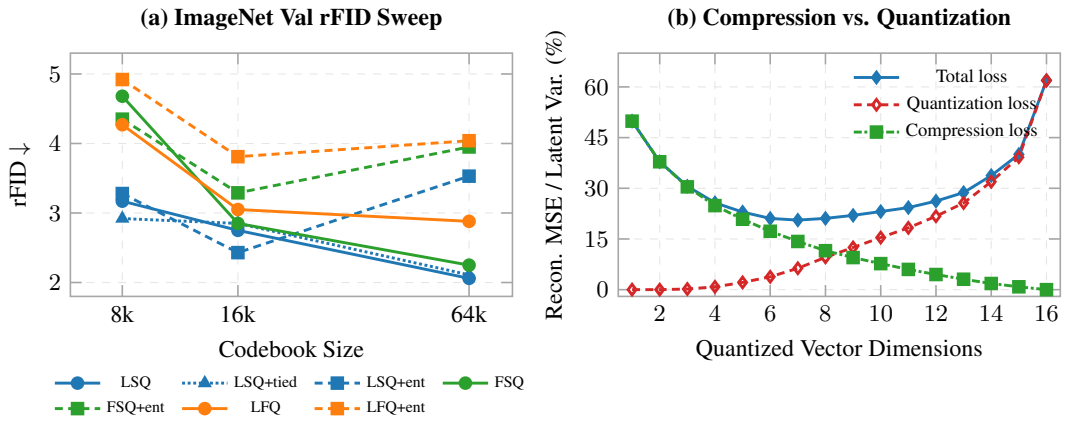


Figure 4: Quantizer performance and error allocation analysis. **(a)** ImageNet 50K validation rFID across codebook sizes (8k–64k). LSQ-based variants achieve superior reconstruction fidelity. **(b)** Reconstruction error breakdown across quantized dimensions c, demonstrating the U-shaped total error curve resulting from the trade-off between compression and quantization loss.

4.2 Uncovering a U-Shape Tradeoff between Compression and Quantization.

Even for a fixed codebook size, FSQ and LSQ support multiple quantization grid shapes. For example, at 65,536 codes one may keep 5 dimensions with levels $[16, 8, 8, 8, 8]$, or 7 with $[16, 8, 4, 4, 4, 4, 2]$, or 16 binary ones — all cost exactly 16 bits per token. This section asks how to pick the best shape.

We note that quantization bottleneck introduces two sources of information loss: compression ($z \to u$) and quantization ($u \to \hat{u}$). A higher-dimensional u minimizes compression loss, but given a fixed codebook budget, it constrains the number of levels per dimension and increases rounding error. Conversely, a lower-dimensional u trades higher compression loss for more levels per dimension, reducing quantization error. We perform an exhaustive search over all valid compression dims and quantization grid shapes and measure both compression and quantization errors. To do so, we freeze the VAE encoder and add simply the quantization bottleneck to minimize the reconstruction error between $\hat{z}$ and z. For codebook size 65,536, there are 231 different configurations, which we evaluate and rank.

Fig. 4b plots the compression and quantization errors across a varying number of quantized vector dimensions on the ImageNet-1K sweep. For each dimension, the plot reports the quantization error under the grid shape that minimizes error. Notably, the total reconstruction error exhibits a U-shaped curve, serving as a guideline for selecting both the number of quantized vector dimensions c and their grid shape, given an uncompressed vector with dimension d. We use this strategy to determine the complete codebook configuration for FLAT-SID, by conducting a similar sweep over our training data.

4.3 FLAT-SID

We adopt Qwen3.5-2B (Qwen Team, 2026) as the backbone for both the representation encoder and the I2T decoder, and SANA (Xie et al., 2024) as the backbone for the T2I decoder. We apply LoRA fine-tuning exclusively to the Qwen3.5-2B components, whereas for the SANA decoder, we fine-tune the full flow model alongside its cross-attention layers. Because neither decoder natively recognizes the semantic IDs, both are trained to process them as a novel semantic condition. Given the backbone, we apply the sweep strategy in Section 4.2 to find the optimal number of compressed dimension to be $c = 14$ and the level configuration to be $[8, 2, \ldots, 2]$. The same compression dim and grid configurations are applied to all 256 registers. The detailed FLAT-SID pre-training configurations are shown in Tab. 5. For each evaluation task, we further fine-tune the pre-trained FLAT-SID checkpoint on one of the three tasks: retrieval, T2I generation and I2T generation. The detailed fine-tuning configurations are shown in Tab. 6. For each task, we report both pre-training and fine-tuning results. Main results in Tab. 1 are computed with $K = 256$ registers, while Fig.1 compares results of different prefix K.

Sparse Retrieval For sparse retrieval, we default use the quantized latent $\hat{u}$ to represent images and text for similarity comparison, maintaining alignment with our training setup. Because each $\hat{u}$ vector maps bijectively to a semantic ID, this approach simulates a production system of storing items as integer IDs, but converting them back to the $\hat{u}$ grid for retrieval. We evaluate performance on the MSCOCO 5K and Flickr30K Karpathy splits (Karpathy & Fei-Fei,

Model	Type	MSCOCO I2T	MSCOCO T2I	Flickr I2T	Flickr T2I
CLIP [29]	Cont.	81.5	62.4	98.7	90.6
ALIGN [14]	Cont.	83.0	69.8	98.7	93.8
FLAVA [30]	Cont.	76.8	67.5	94.0	89.4
FILIP [38]	Cont.	84.3	70.6	99.2	93.4
Florence [42]	Cont.	85.9	71.4	99.1	93.6
COCA [39]	Cont.	86.2	74.2	99.5	95.7
CSR [34]	Sparse	83.5	71.4	96.8	88.0
FLAT [31]	Cont.	88.4	77.9	98.3	93.6
FLAT-SID	**Sparse**	**83.9**	**70.9**	**95.6**	**88.5**
(Pretrain)	Sparse	40.9	53.7	72.2	78.6

(a) I2T and T2I retrieval Recall@5.

Model	BLEU@4	METEOR	ROUGE-L	CIDEr	SPICE
MNIC [8]	30.9	27.5	55.6	108.1	21.0
MIR [18]	32.5	27.2	–	109.5	20.6
NAIC-CMAL [12]	35.3	27.3	56.9	115.5	20.8
SATIC [47]	32.9	27.0	–	111.0	20.5
SCD-Net [22]	37.3	28.1	58.0	118.0	21.6
COCA [39]	40.9	33.9	-	143.6	24.7
CrossFlow [20]	36.4	27.8	57.1	116.2	20.4
FLAT [31]	40.5	31.2	60.8	138.6	24.4
FLAT-SID	**41.2**	**31.6**	**60.8**	**139.4**	**24.5**
(Pretrain)	34.5	28.6	56.9	116.2	21.2

(b) I2T generation.

Model	Type	Score
TA-TiTok [16]	Disc.	0.53
TokenFlow [27]	Disc.	0.55
DC-AR [35]	Disc.	0.69
MetaQuery [26]	Cont.	0.61
MetaQuery(full FT)	Cont.	0.61
Janus-Pro-7B [4]	Cont.	0.80
SANA-1.6B [36]	Cont.	0.66
FLAT [31]	Cont.	0.83
FLAT-SID	**Disc.**	**0.81**
(Pretrain)	Disc.	0.75

(c) T2I generation.

Table 1: FLAT-SID performance on I2T and T2I retrieval and generation, comparing with baselines.

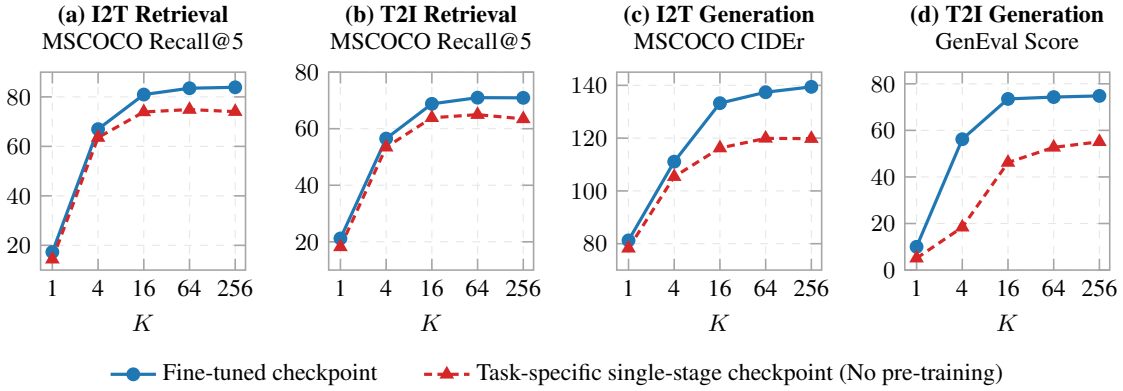


Figure 5: Retrieval and generation performance across various values of K. Fine-tuning consistently outperforms task-specific single-stage training, with larger K yielding steady gains.

2015). Tab. 1a presents the T2I and I2T retrieval results. FLAT-SID matches CSR (Wen et al., 2025), a state-of-the-art sparse retriever that represents items using top-K sparse activations. In App. E.2, we explore as an A/B test to use the up-projected latent $\hat{z}$, for contrastive learning and retrieval, where we found contrasting the sparse $\hat{u}$ space is more effective. In App. E.3, we also conduct a test-time retrieval ablation using semantic IDs directly without mapping them back to either $\hat{u}$ or $\hat{z}$, which also achieves competitive results and demonstrating that the integer ID space itself carries useful cross-modal information. App. F.6 shows examples of retrieval success and failures.

I2T Generation I2T generation is performed by passing the image latent $\hat{z}^{\text{img}}$ to the text decoder as prompt tokens. Since each latent $\hat{z}$ acts as the embedding entry for a unique semantic ID learned via the $\hat{u} \rightarrow \hat{z}$ up-projection, the setup mirrors a scenario where images are stored as semantic IDs, with embedding table parameterized by columns of $\hat{z}$. Following prior work cited in Tab. 1b, performance is evaluated on the MSCOCO 5K Karpathy split (Karpathy & Fei-Fei, 2015) using BLEU-4, METEOR, ROUGE-L, CIDEr, and SPICE. As shown in Tab. 1b, FLAT-SID achieves comparable performance to SOTA continuous baselines such as COCA (Yu et al., 2022) and FLAT (Sun et al., 2026). In App. F.4 we show I2T examples generated by different prefix length K.

T2I Generation Similar to I2T, text-to-image (T2I) generation is achieved by passing the text latent $\hat{z}^{\text{txt}}$ to the image decoder as semantic conditioning. Performance is evaluated using GenEval (Ghosh et al., 2023), with results presented in Tab. 1c. We compare against several key baselines: TA-TiTok (Kim et al., 2025), a text-informed image tokenizer trained on image reconstruction; MetaQuery (Pan et al., 2025), which resamples inputs into continuous 1D tokens using an image generation loss (evaluated in both frozen and trained encoder variants); and public SANA 1.6B, from which the FLAT image decoder is initialized. The SANA baseline uses frozen T5 embeddings as semantic conditions, while FLAT-SID trains it to understand a new representation. Overall, FLAT-SID outperforms the above baselines. The detailed GenEval scores by category is presented in App. E.5 and Tab. 9, where we also compare the pre-trained and fine-tuned checkpoints.

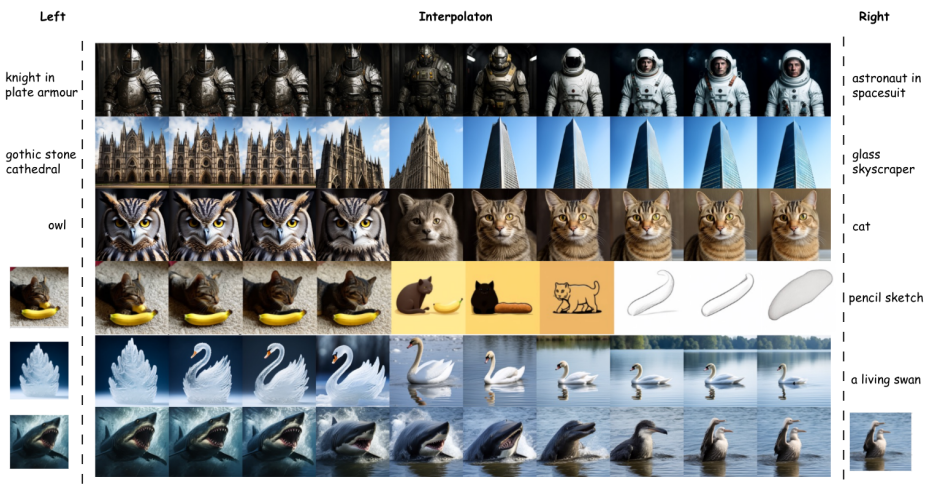


Figure 6: Linear interpolation between two inputs (left and right), which can be two texts, a text-image pair, or two images.

Flexible-Length Because FLAT-SID is trained with nested dropout, it exhibits Matryoshka-style behavior, allowing prefix-K tokens to be used for both retrieval and generation. As shown in Fig. 5, performance steadily improves across all tasks as K increases. Furthermore, the figure compares checkpoints with and without pre-training, demonstrating that pre-training then fine-tuning consistently achieves superior results across all values of K. Fig. 1 shows flexible-K T2I and I2G generation examples. More examples can be found in App. F. In addition, we demonstrate that FLAT-SID can generalize to unseen prefix-K in App. F.3.

Linear Interpolation Given two inputs—each being either text or an image—we perform linear interpolation on their semantic representations and visualize the decoded image content. Interpolation is performed in the lookup embedding space $\hat{z}$ as $\lambda\hat{z}^a + (1-\lambda)\hat{z}^b$, where a and b represent the two inputs. The resulting interpolated embedding is then passed to the image decoder as semantic conditioning. Fig. 6 demonstrates this interpolation by varying λ from 0 to 1 in steps of 0.1 across text–text, image–text, and image–image input pairs.

Storage-Efficient Linear Probing Fig. 7 and Tab. 8 compare frozen ImageNet-1K linear-probing accuracy against representation storage (bytes). As baselines, we consider the continuous FLAT and the backbone Qwen 3.5's last hidden state. For FLAT-SID, the linear probing is conducted with two variants, one using the quantized code $\hat{u}$, the other using the simulated look-up $\hat{z}$; both can be deterministically obtained from stored semantic integer IDs. As results, we found both variants perform similarly. At $K{=}256$, FLAT-SID approaches continuous FLAT performance (80.17% versus 81.76%) with $64\times$ less storage. Under matched 128- and 512-byte budgets, it improves over continuous FLAT by 4.75 and 3.18 points, and also outperforms the Qwen representation with $8\times$ less storage.

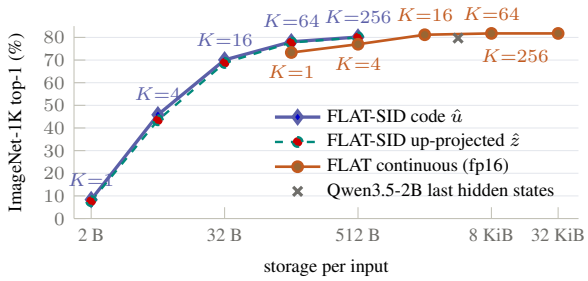


Figure 7: **ImageNet-1K linear-probe accuracy versus storage.** FLAT-SID uses 16-bit integers while FLAT uses fp16.

5 Conclusion

We introduce FLAT-SID, a discrete representation pre-training framework that resamples images and texts into a shared 1D space while jointly training the representation encoder, quantizer, and cross-modal decoders. To bridge the gap between discrete and continuous representation learning, we adopt LSQ as our lookup-free quantization strategy, enabling end-to-end quantization-aware training. Through systematic ablations, we evaluate key architectural choices, including the selection of lookup-free quantizers, latent compression dimensionality, and per-dim grid levels. FLAT-SID achieves strong performance across both T2I and I2T retrieval and generation within a single unified architecture, all while flexibly supporting a variable number of semantic IDs during inference. Overall, FLAT-SID demonstrates the effectiveness of applying quantization-aware training for end-to-end discrete representation learning within unified multimodal generative models.

AI USE STATEMENT

We used generative AI tools to assist with experiment planning and management, edit training and evaluation scripts, organize experiment reports, and summarize and interpret experimental results. We also used these tools to improve the readability of the manuscript, format references, and prepare figures and other presentation artifacts. All AI-assisted code, analyses, figures, and text were reviewed and verified by the authors. The authors take responsibility for the final content of this work.

REFERENCES

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Ouguzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. *ArXiv*, abs/2502.13967, 2025. URL https://api.semanticscholar.org/CorpusID:276450068.

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

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. FLUX.1 Kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and K. Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *ArXiv*, abs/1805.06085, 2018. URL https://api.semanticscholar.org/CorpusID:21721698.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12868–12878, 2021. URL https://api.semanticscholar.org/CorpusID:229297973.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. *ArXiv*, abs/1902.08153, 2019. URL https://api.semanticscholar.org/CorpusID:67788003.

Junlong Gao, Xi Meng, Shiqi Wang, Xia Li, Shanshe Wang, Siwei Ma, and Wen Gao. Masked non-autoregressive image captioning. *arXiv preprint arXiv:1906.00717*, 2019.

Yuying Ge, Yixiao Ge, Ziyun Zeng, Xintao Wang, and Ying Shan. Planting a seed of vision in large language model. *ArXiv*, abs/2307.08041, 2023. URL https://api.semanticscholar.org/CorpusID:259937351.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023.

Ming Gui, Johannes Schusterbauer, Timy Phan, Felix Krause, Joshua M. Susskind, Miguel Angel Bautista, and Bjorn Ommer. Adapting self-supervised representations as a latent space for efficient generation. *ArXiv*, abs/2510.14630, 2025.

Longteng Guo, Jing Liu, Xinxin Zhu, Xingjian He, Jie Jiang, and Hanqing Lu. Non-autoregressive image captioning with counterfactuals-critical multi-agent learning. *arXiv preprint arXiv:2005.04690*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, 2022. arXiv:2111.06377.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pp. 4904–4916. PMLR, 2021.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

Dongwon Kim, Ju He, Qihang Yu, Chenglin Yang, Xiaohui Shen, Suha Kwak, and Liang-Chieh Chen. Democratizing text-to-image masked generative models with compact text-aware one-dimensional tokens. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18442–18452, 2025. URL https://api.semanticscholar.org/CorpusID:275515598.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11513–11522, 2022. URL https://api.semanticscholar.org/CorpusID:247244535.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, 2018.

Chao Li, Tianhong Li, Sai Vidyaranya Nuthalapati, Hong-You Chen, Satya Narayan Shukla, Jianpeng Cheng, Yonghuan Yang, Jun Xiao, Xiangjun Fan, Aashu Singh, Dina Katabi, and Shlok Kumar Mishra. Unifying contrastive and generative objectives for visual understanding and text-to-image generation. *arXiv preprint arXiv:2603.02667*, 2026.

Qihao Liu, Xi Yin, Alan Yuille, Andrew Brown, and Mannat Singh. Flowing from words to pixels: A noise-free framework for cross-modality evolution. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2755–2765. IEEE, 2025.

Jiasen Lu, Liangchen Song, Mingze Xu, Byeongjoo Ahn, Yanjun Wang, Chen Chen, Afshin Dehghan, and Yinfei Yang. Atoken: A unified tokenizer for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28701–28711, 2026.

Jianjie Luo, Yehao Li, Yingwei Pan, Ting Yao, Jianlin Feng, Hongyang Chao, and Tao Mei. Semantic-conditional diffusion networks for image captioning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23359–23368. IEEE, 2023.

Fabian Mentzer, David C. Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *ArXiv*, abs/2309.15505, 2023.

Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-d-piece: Image tokenizer meets quality-controllable compression. *ArXiv*, abs/2501.10064, 2025. URL https://api.semanticscholar.org/CorpusID:275606373.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russ Howes, Po-Yao (Bernie) Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *ArXiv*, abs/2304.07193, 2023. URL https://api.semanticscholar.org/CorpusID:258170077.

Xichen Pan, Satya Narayan Shukla, Aashu Singh, Zhuokai Zhao, Shlok Kumar Mishra, Jialiang Wang, Zhiyang Xu, Jiuhai Chen, Kunpeng Li, Felix Juefei-Xu, Ji Hou, and Saining Xie. Transfer between modalities with metaqueries. *ArXiv*, abs/2504.06256, 2025. URL https://api.semanticscholar.org/CorpusID:277626959.

Liao Qu, Huichao Zhang, Yiheng Liu, Xu Wang, Yi Jiang, Yiming Gao, Hu Ye, Daniel K Du, Zehuan Yuan, and Xinglong Wu. Tokenflow: Unified image tokenizer for multimodal understanding and generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2545–2555. IEEE, 2025.

Qwen Team. Qwen3.5-omni technical report. *ArXiv*, abs/2604.15804, 2026. URL https://api.semanticscholar.org/CorpusID:287607592.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. Flava: A foundational language and vision alignment model. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15617–15629. IEEE, 2022.

Guangyu Sun, Shlok Kumar Mishra, Wentao Bao, Robert Zhenheng Yang, Xiao Wang, Xiyuan Wang, Yujunrong Ma, Chen Yuan, Max Xiangjun Fan, Jun Xiao, and Jianpeng Cheng. Flat: Resampling image and text into 1d flexible-length aligned transmodal tokens for retrieval and generation, 2026. URL https://arxiv.org/abs/2609.16591.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:20282961.

Xudong Wang, Xingyi Zhou, Alireza Fathi, Trevor Darrell, and Cordelia Schmid. Visual lexicon: Rich image features in language space. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19736–19747, 2025. URL https://api.semanticscholar.org/CorpusID:274610575.

Tiansheng Wen, Yifei Wang, Zequn Zeng, Zhong Peng, Yudi Su, Xinyang Liu, Bo Chen, Hongwei Liu, Stefanie Jegelka, and Chenyu You. Beyond matryoshka: Revisiting sparse coding for adaptive representation. *arXiv preprint arXiv:2503.01776*, 2025.

Yecheng Wu, Han Cai, Junyu Chen, Zhuoyang Zhang, Enze Xie, Jincheng Yu, Junsong Chen, Jinyi Hu, Yao Lu, and Song Han. Dc-ar: Efficient masked autoregressive image generation with deep compression hybrid tokenizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18034–18045, 2025.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *ArXiv*, abs/2410.10629, 2024. URL https://api.semanticscholar.org/CorpusID:273346094.

Wilson Yan, Matei Zaharia, Volodymyr Mnih, Pieter Abbeel, Aleksandra Faust, and Hao Liu. Elastictok: Adaptive tokenization for image and video. *ArXiv*, abs/2410.08368, 2024. URL https://api.semanticscholar.org/CorpusID:273323724.

Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. Filip: Fine-grained interactive language-image pre-training. *arXiv preprint arXiv:2111.07783*, 2021.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

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

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *ArXiv*, abs/2406.07550, 2024b.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2022. URL https://api.semanticscholar.org/CorpusID:236149944.

Kaiwen Zha, Lijun Yu, Alireza Fathi, David A. Ross, Cordelia Schmid, Dina Katabi, and Xiuye Gu. Language-guided image tokenization for generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15713–15722, 2025. URL https://api.semanticscholar.org/CorpusID:274597720.

Xinliang Zhang, Lei Zhu, Hangzhou He, Shuang Zeng, Ourui Fu, Jiakui Hu, Zhengjian Yao, and Yanye Lu. Adatok: Adaptive token compression with object-aware representations for efficient multimodal llms. *ArXiv*, abs/2511.14169, 2025. URL https://api.semanticscholar.org/CorpusID:283080727.

Yue Zhao, Yuanjun Xiong, and Philipp Krahenbuhl. Image and video tokenization with binary spherical quantization. *ArXiv*, abs/2406.07548, 2024. URL https://api.semanticscholar.org/CorpusID:270380237.

Yuanen Zhou, Yong Zhang, Zhenzhen Hu, and Meng Wang. Semi-autoregressive transformer for image captioning. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 3132–3136. IEEE, 2021.

LIST OF APPENDICES

A Comparison of 1D Representations **14**

B Learned Step Quantization **15**

B.1 Overview of Learned Step Quantization 15

B.2 Training Procedure and Gradient Flow 15

B.3 Implementation 15

C LSQ with a Tied Orthonormal Parameterization **16**

D ImageNet Experiments Additions **18**

D.1 Training and Evaluation Configuration 18

D.2 Training Plots for Different Codebook Size 18

D.3 Reconstruction Examples 19

D.4 Eval Codebook Utilization 19

E FLAT-SID Experiments Additions **21**

E.1 Training Configurations 21

E.2 Ablation on Contrastive space: $\hat{u}$ vs $\hat{z}$ 21

E.3 Test-time Retrieval with Integer IDs? 22

E.4 Linear-probing results 22

E.5 GenEval Detailed Scores 22

F Qualitative Galleries **23**

F.1 Image Generation with Different Prefix Lengths 23

F.2 Generation after Bit Flipping 23

F.3 Generalization to Unseen Prefix Lengths 26

F.4 Captioning across Prefix Lengths 26

F.5 Semantic Structure of Individual IDs 26

F.6 Retrieval Successes and Failures 26

A Comparison of 1D Representations

The following table directly compares various 1D representations across multiple dimensions.

Method	Flex.	Disc.	Text-Guided	Transmodal	Objective
TiTok (Yu et al., 2024b)	×	✓	×	×	I Recon
FlexTok (Bachmann et al., 2025)	✓	✓	×	×	I Recon
One-D-Piece (Miwa et al., 2025)	✓	✓	×	×	I Recon
ElasticTok (Yan et al., 2024)	✓	✓	×	×	I+V Recon
AdaTok (Zhang et al., 2025)	✓	✓	×	×	I Recon
TexTok (Zha et al., 2025)	×	✓	✓	×	I Recon (condition on T)
TA-TiTok (Kim et al., 2025)	×	✓	✓	×	I Recon (condition on T)
SEED Ge et al. (2023)	×	✓	✓	×	I→T Align + I Recon
ViLex (Wang et al., 2025)	✓	×	✓	×	I Recon
MetaQueries (Pan et al., 2025)	×	×	✓	✓	T→I Gen
RepTok (Gui et al., 2025)	×	✓	×	×	I Recon
AToken (Lu et al., 2026)	×	✓	✓	×	I→T Align + I Recon
FLAT-SID (ours)	✓	✓	✓	✓	**I↔T Align + Gen**

Table 2: **Comparison of 1D representations.** `Flex.`=flexible K via nested/tail-drop where any prefix can be chosen (coarse-to-fine representations), `Disc.`=discrete IDs, `Transmodal` means the tokenizer naitively supports T→I / I→T generation. `I`=Image, `V`=Video, `T`=Text. Prior 1D tokenizers mainly optimize image reconstruction; FLAT-SID has no reconstruction loss and learns transmodal tokens via T→I and I→T alignment + generation directly.

B LEARNED STEP QUANTIZATION

B.1 OVERVIEW OF LEARNED STEP QUANTIZATION

Learned Step Quantization (LSQ) optimizes quantization step sizes jointly with network parameters. Rather than fitting scale factors post-hoc via min-max ranges or static heuristics, LSQ parameterizes the step size s as a continuous, trainable parameter.

For a continuous parameter tensor u (representing weights or activations), standard uniform quantization with step size s maps values to discrete integer levels within bounded range $[Q^-, Q^+]$:

$$\hat{u} = \text{clip}\left(\left\lfloor \frac{u}{s} \right\rceil, Q^-, Q^+\right) \cdot s$$

where $\lfloor \cdot \rceil$ denotes element-wise rounding to the nearest integer, and the clipping operator restricts outputs to target bit-width constraints (e.g., $Q^- = -8$ and $Q^+ = 7$ for 4-bit signed representation).

B.2 TRAINING PROCEDURE AND GRADIENT FLOW

Training LSQ requires propagating gradients through non-differentiable rounding operators and balancing optimization scales across parameters of varying dimensions.

Gradient Formulation The derivative of the rounding function is zero almost everywhere. LSQ employs the Straight-Through Estimator (STE) to pass gradients through rounding while computing analytical partial derivatives for the step size parameter s:

$$\frac{\partial \hat{u}}{\partial u} = \begin{cases} 1 & \text{if } Q^- < \frac{u}{s} < Q^+ \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial \hat{u}}{\partial s} = \begin{cases} -\frac{u}{s} + \left\lfloor \frac{u}{s} \right\rceil & \text{if } Q^- < \frac{u}{s} < Q^+ \\ Q^- & \text{if } \frac{u}{s} \leq Q^- \\ Q^+ & \text{if } \frac{u}{s} \geq Q^+ \end{cases}$$

Gradient Scale Adjustment Because scale factors impact all elements in a tensor simultaneously, gradients accumulated for s can explode or vanish relative to parameter updates. To stabilize SGD step updates, the gradient of s is scaled by factor g:

$$g = \frac{1}{\sqrt{N \cdot Q^+}}$$

where N is the number of elements sharing step size s. The adjusted gradient $\nabla_s \mathcal{L}_{\text{scaled}} = g \cdot \nabla_s \mathcal{L}$ is then used in optimizer updates.

Initialization Step sizes are initialized using tensor statistics prior to quantization training:

$$s_0 = \frac{2\mathbb{E}\left[|u|\right]}{\sqrt{Q^+}}$$

B.3 IMPLEMENTATION

```python
def grad_scale(x: Tensor, scale: float) -> Tensor:
    """Value passes through unchanged; the gradient is multiplied by scale."""
    return x * scale + (x - x * scale).detach()


def round_pass(x: Tensor) -> Tensor:
    """Round with a straight-through gradient."""
    return (x.round() - x).detach() + x
```

```python

class LSQQuantizer(nn.Module):
    """Project latent_dim -> code_dim, scalar-quantize per dimension with a learned
    step, project back."""

    def __init__(
        self,
        latent_dim: int,
        levels: Sequence[int],
        grad_scale_on: bool = True,
    ):
        super().__init__()
        levels = [int(x) for x in levels]
        self.latent_dim = latent_dim
        self.code_dim = len(levels)
        self.grad_scale_on = grad_scale_on

        lv = torch.tensor(levels, dtype=torch.float32)
        self.register_buffer("_levels", lv, persistent=False)
        self.register_buffer("Qn", -(lv // 2), persistent=False)
        self.register_buffer("Qp", (lv - 1) // 2, persistent=False)
        # Even levels quantize u/s - 0.5 and reconstruct (q + 0.5)*s, so their reachable
        # set is symmetric about zero (L=4 -> {-1.5,-0.5,0.5,1.5}*s instead of {-2,-1,0,1}).
        self.register_buffer("offset", torch.where(lv % 2 == 0, 0.5, 0.0), persistent=False)
        self.max_bits = float(lv.log2().sum())

        self.in_proj = nn.Linear(latent_dim, self.code_dim)
        self.out_proj = nn.Linear(self.code_dim, latent_dim)
        self.step = nn.Parameter(torch.ones(self.code_dim))


    def forward(self, z: Tensor, latent_recon: bool = False):
        """z: (B, N, latent_dim) continuous latent, post-LatentNorm."""
        # Quantization must not run in bf16: the round/clamp grid and the learned step
        # need more mantissa than bf16's 8 bits, or neighbouring levels alias together.
        with torch.autocast(device_type=z.device.type, enabled=False):
            zf = z.float()
            u = self.in_proj(zf)                          # (B,N,code_dim)
            s = self.step.abs().clamp_min(1e-4)
            if self.grad_scale_on:
                half_range = float((self.Qp + self.offset).max())
                g = 1.0 / math.sqrt(max(1.0, u[..., 0].numel() * half_range))
                s = grad_scale(s, g)

            v = torch.clamp(u / s - self.offset, self.Qn, self.Qp)
            q = round_pass(v)
            code = (q + self.offset) * s                  # (B,N,code_dim) -- the contrastive space
            z_hat = self.out_proj(code)                   # (B,N,latent_dim) -- decoders consume this

            zero = z_hat.new_zeros(())
            recon = torch.nn.functional.mse_loss(z_hat, zf.detach()) if latent_recon else zero

        return code, z_hat.to(z.dtype), recon
```

Listing 1: Learned Step Size Quantization of the register latent.

C LSQ WITH A TIED ORTHONORMAL PARAMETERIZATION

Since LSQ does not require a fixed saturating nonlinearity such as $\tanh$, we hope this property encourages the quantized latent to preserve, as closely as its finite precision allows, the geometric structure of the continuous latent. In QAT terms, we hope to minimize the initial forward mismatch between continuous and discrete representation training, by training the model to be quantization-aware.

To reduce the gap, we experiment with a variant (i.e., LSQ-tied) where we tie the weights of the down-projection $z \to u$ and the up-projection $\hat{u} \to \hat{z}$. Let $W \in \mathbb{R}^{c \times d}$ have orthonormal rows, so that $WW^\top = I_c$. This parameterization provides a useful separation between continuous compression and quantization, since the reconstruction error decomposes as

$$\|z - \hat{z}\|_2^2 = \underbrace{\left\|(I - W^\top W)z\right\|_2^2}_{\text{compression error}} + \underbrace{\|u - \hat{u}\|_2^2}_{\text{quantization error}} . \tag{11}$$

The first term is the information discarded by dimensionality reduction $z \to u$, whereas the second is the error introduced by LSQ. Initializing W with the leading PCA directions minimizes the compression term among linear rank-c

orthogonal projections. Consequently, relative to the corresponding continuous representation, the only additional error introduced at initialization is the LSQ quantization error. If the projection discards no information, the full bottleneck reconstruction error reduces exactly to the quantization error, recovering the standard QAT setting.

D IMAGENET EXPERIMENTS ADDITIONS

D.1 TRAINING AND EVALUATION CONFIGURATION

Table 3 shows the training and evaluation configurations for the ImageNet-1K tokenization experiment, where the goal is to compare various lookup-free quantizers.

	Setting
Resolution	256×256
VAE latent	16ch, 32×32 grid = 1024 tokens / image
Epochs	10
Precision	FP32
GPUs	8
Local batch	32 (global 256)
Optimizer	AdamW
LR	5e-4
Betas	(0.5, 0.9)
Schedule	cosine
Loss ℓ_1	0.2
Loss ℓ_2	1.0
Loss LPIPS	1.0
Adversarial	DINO-based hinge, adaptive weight
Evaluation	encode $\rightarrow$ quantize $\rightarrow$ decode only no class labels, no CFG

Table 3: **Clean tokenizer comparison setup.** Fixed image tokenizer training for rFID (no CFG, no labels).

D.2 TRAINING PLOTS FOR DIFFERENT CODEBOOK SIZE

Figure 8 compares various look-up free quantizers at codebook size 8192 on ImageNet-1K during the training process. The rFID is measured on 1K validation subset for every checkpointing and the codebook utilization is measured on training data using 200 steps EMA. Figure 9 plots a similar comparison with codebook size 65,536. In general, both figures exhibit similar trends where LSQ achieves lower rFID compared to FSQ and LFQ. Meanwhile, higher codebook utilization does not guarantee lower rFID.

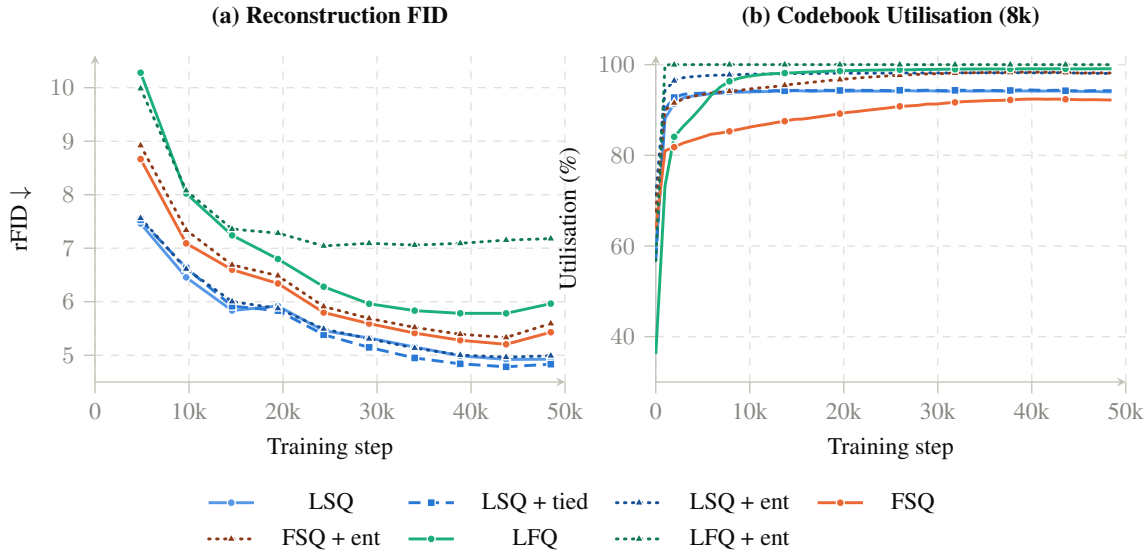


Figure 8: Comparison of various look-up free quantizers at codebook size 8192 on ImageNet-1K training process.

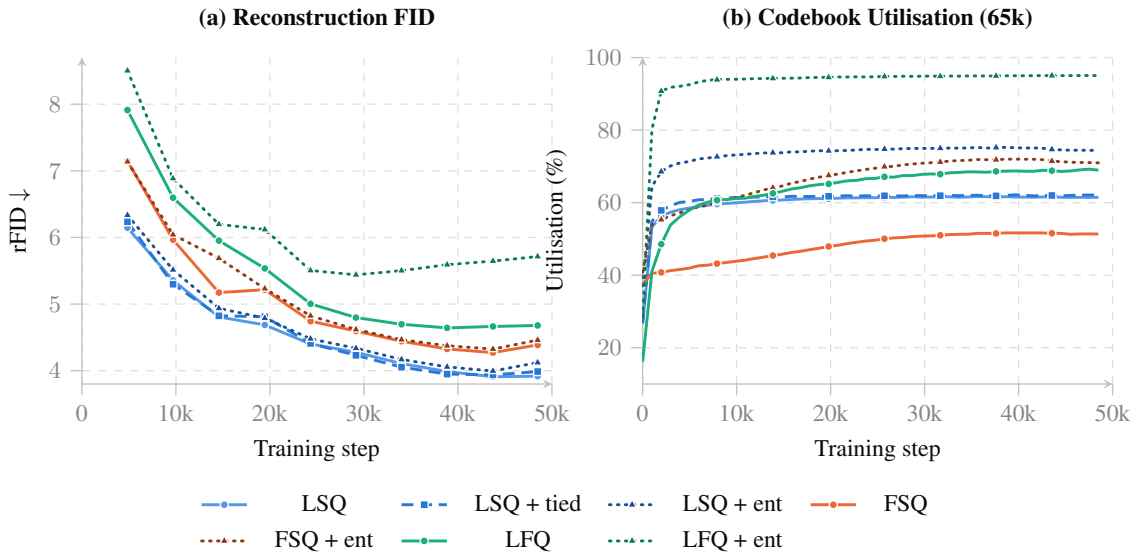


Figure 9: Comparison of various look-up free quantizers at codebook size 65,536 on ImageNet-1K training process.

D.3 Reconstruction Examples

Figure 10 shows the reconstructed images under codebook size 16,384 for various quantizer variants on resolution 256x256.



Figure 10: Reconstructed images using various look-up free quantizers with codebook 16384.

D.4 Eval Codebook Utilization

Table 4 shows the evaluation codebook activation. The evaluation protocol is as follows: since LSQ quantizes each vector to a product grid for every dimension, we measure how much of that codebook is actually reached, both per dimension and jointly, on the full ImageNet-1k validation split (50,000 images at 256×256, 51,200,000 tokens), using the EMA weights of the final checkpoint. As the results show, every dimension reaches all L_j of its levels at all three settings, so no dimension partially collapses. For all levels jointly, all 8,192 codes are used at the smaller codebook setting and 99.23% at 65,536; the tokenizer therefore occupies effectively its entire codebook.

Codebook	Levels	Distinct levels used per dimension					Distinct codes
		j=0	j=1	j=2	j=3	j=4	
8,192	$[8, 8, 8, 4, 4]$	8/8	8/8	8/8	4/4	4/4	$8{,}192/8{,}192$ (100.00%)
16,384	$[8, 8, 8, 8, 4]$	8/8	8/8	8/8	8/8	4/4	$16{,}383/16{,}384$ (99.99%)
65,536	$[16, 8, 8, 8, 8]$	16/16	8/8	8/8	8/8	8/8	$65{,}030/65{,}536$ (99.23%)

Table 4: Codebook utilization of LSQ at 3 codebook sizes, over 51.2M validation tokens. Every dimension attains all of its L_j levels at every setting, and the joint codebook is essentially fully occupied.

E FLAT-SID EXPERIMENTS ADDITIONS

E.1 TRAINING CONFIGURATIONS

Table 5 shows the full pre-training configs. Table 6 shows the full fine-tuning configs.

Component	Configuration
Encoder and text decoder	
Model	Qwen3.5-2B, bfloat16
LoRA	$r = 16$, $\alpha = 32$, dropout 0.05; applied to Q/K/V/O and gate/up/down projections
Adapters	Two adapters: default (encoder) and decoder (caption), both using the same configuration
Tokenizer	
Register tokens	256
Nested dropout	$p = 1.0$, $\mathcal{K} = \{1, 4, 16, 64, 256\}$; one shared K per optimization step; truncation rather than null padding
Minimum tokens	1
Codebook	MLP head with norm matching and latent normalization enabled
Quantizer (LSQ)	
Type	`lsq_tied`, with shared projection W, mean, and step size
Dimensions	code dimension 14
Levels	$\left[8, \underbrace{2, \ldots, 2}_{13}\right]$, giving $8 \times 2^{13} = 2^{16}$ codes per token
Contrastive space	Quantized code space
Gradient scaling	Enabled
Objectives	
Loss weights	Contrastive: 1.0; text generation: 1.0; image generation: 1.0
Contrastive objective	InfoNCE with temperature 0.07
Image Decoder	
Model	SANA-1600M at 512 px, fully fine-tuned
Connector	16 Transformer layers with input scale 2.3452
Flow matching	Flow objective, shift 3.0
Data	
Size	$\sim$66M curated image-text pairs
Image resolution	512×512
Data workers	20 per rank
Optimization	
Optimizer	AdamW, $\beta_2 = 0.999$, weight decay 0.0
Learning rate	5.0×10^{-4}; 500-step linear warm-up followed by cosine decay to exactly zero
Batch size	16 per rank $\times$ 64 ranks $\times$ 1 accumulation step $= 1024$ global
Hardware	8 nodes $\times$ 8 GPUs
Training steps	200,000

Table 5: **Model and training configuration.**

E.2 ABLATION ON CONTRASTIVE SPACE: $\hat{u}$ VS $\hat{z}$

For each input FLAT-SID emits quantized $\hat{u} \in \mathbb{R}^c$ and its up-projection $\hat{z} = W_c^\top \hat{u} \in \mathbb{R}^d$. Since $\hat{u}$ maps bijectively to the semantic ID, $\hat{z}$ can be seen as the codebook embedding of that ID. We ablate where contrastive should be placed, on $\hat{u}$ or $\hat{z}$.

We conduct mid-training from the same pretrained FLAT-SID checkpoint, training for 36K steps with the full objective $\mathcal{L}_{\text{align}} + \mathcal{L}_{\text{txt}} + \mathcal{L}_{\text{img}}$. We vary whether $\mathcal{L}_{\text{align}}$ is computed on $\hat{u}$ (`code`) or $\hat{z}$. Learning rate used in this study is 1e-4. We evaluate both the MSCOCO retrieval and GenEval scores on the mid-trained checkpoints.

	I2T generation	T2I generation	Retrieval
Trainable scope	text decoder LoRA only + text connector	Two jobs for A/B test. A: image decoder full-FT + image connector; B: image decoder + image connector + encoder + quantizer	encoder and quantizer
Objective w (align / txt / img)	0 / 1 / 0	0 / 0 / 1	1 / 0 / 0
Optimizer	AdamW, β=(0.9, 0.999)	AdamW, β=(0.9, 0.999)	AdamW, β=(0.9, 0.999)
LR	1e-4	1e-4	1e-4
Weight decay	0	0	0
Schedule	linear warmup $\rightarrow$ cosine to 0	linear warmup $\rightarrow$ cosine to 0	linear warmup $\rightarrow$ cosine to 0
Warmup steps	2,213	500	500
Max steps	44,253	50,000	100,000
Grad clip	1.0	1.0	1.0
GPUs	4 (1 node)	64 (8 nodes)	64 (8 nodes)
Per-GPU batch $\times$ accum	16×4	16×1	16×1
Global batch	256	1,024	1,024
Precision	bf16	bf16	bf16
Grad checkpointing	off	on	on
Image resolution	448×448	512×512	512×512
Max text length	64	128	128
Max Registers N	256 / 256 / 256		
K ladder	{1,4,16,64,256}, uniform sample	{1,4,16,64,256}, uniform sample	{1,4,16,64,256}, uniform sample
Contrastive temp	—	—	0.07

Table 6: **Task-specific fine-tuning configurations.** I2T = caption decoder-only FT, T2I = Image decoder full-FT, Retrieval = joint stage-3 with all three losses (no task-specific FT).

Results are shown in Table 7. MSCOCO Retrieval is close in all combos. Surprisingly, the use of $\hat{u}$ and $\hat{z}$ does not have to be consistent for training and inference. For generation, contrastive on code is stronger. We therefore default to contrastive on the quantized sparse code $\hat{u}$ for main results in the paper

Train $\mathcal{L}_{\text{align}}$ on	Retrieval on	i2t R@5	t2i R@5	GenEval
$\hat{u}$ (code)	$\hat{u}$	40.36	51.84	**66.11**
	$\hat{z}$	**43.56**	51.26	
$\hat{z}$	$\hat{u}$	42.94	**52.74**	64.97
	$\hat{z}$	42.88	52.73	

Table 7: Contrastive space: $\hat{u}$ (code) vs $\hat{z}$. 36K-step mid-training ablation from FLAT-SID.

E.3 Test-time Retrieval with Integer IDs?

We also investigate whether direct cross-modal retrieval is possible using discrete integer semantic IDs at inference time, bypassing projection back to continuous latent spaces ($\hat{\mathbf{u}}$ or $\hat{\mathbf{z}}$). For a sequence of K register positions yielding K integer IDs, we evaluate two direct retrieval mechanisms: (1) *Aligned ID match*, which counts exact matches at corresponding register positions, and (2) *Bag-of-ID TF-IDF*, which represents each item as a sparse histogram over the $65{,}536$ IDs and computes TF-IDF cosine similarity via an inverted index.

We use the pre-trained FLAT checkpoint for this study. As shown in Figure 11, discrete integer IDs perform effective retrieval despite high sparsity, confirming that they retain substantial cross-modal structure. The sparse Bag-of-ID metric outperforms position-aligned matching at longer prefix lengths, indicating that semantic information is distributed across the register sequence rather than strictly localized to identical positions. Furthermore, unseen prefix lengths—including $K \in \{2, 3, 8, 12, 32, 128\}$—follow the same favorable scaling trends observed at trained lengths.

E.4 Linear-probing results

Table 8 reports the full numbers of Figure 7 ImageNet-1K linear probing. The storage is computed as the bytes required to store each representation. For FLAT-SID, we measure both the quantized code $\hat{u}$ and the up-projected embedding $\hat{z}$. The comparison additionally includes MAE-B (He et al., 2022) and DREAM (Li et al., 2026) as baselines.

E.5 GenEval Detailed Scores

Table **??** provides a category-level breakdown of GenEval performance across three configurations: the base pre-trained model, decoder-only fine-tuning, and joint fine-tuning of the encoder, quantizer, and T2I decoder. The pre-

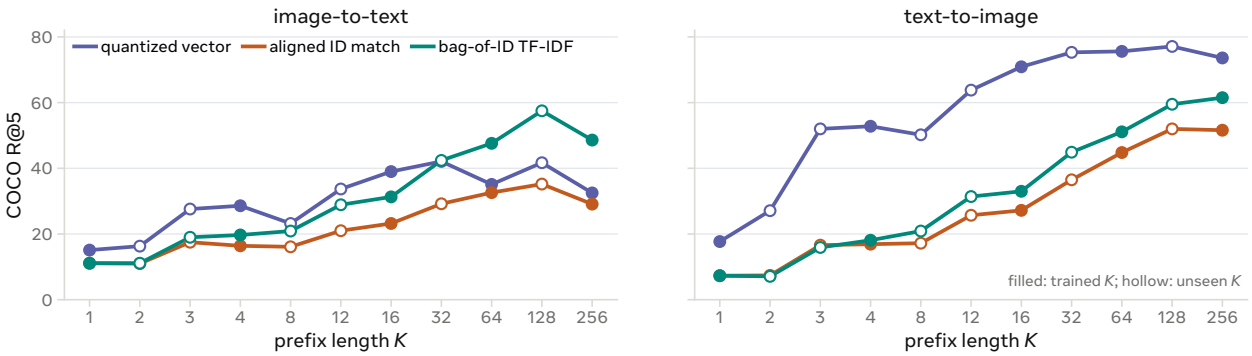


Figure 11: Cross-modal retrieval using discrete semantic IDs across varying prefix lengths K. Solid markers denote prefix lengths sampled during training, whereas hollow markers indicate unseen intermediate lengths. Discrete integer matching functions effectively across the entire prefix range, with Bag-of-ID TF-IDF achieving 48.6% I2T and 61.5% T2I R@5 at $K{=}256$ using the pre-trained FLAT checkpoint.

	FLAT-SID			Continuous FLAT
K	Storage (B)	Code $\hat{u}$	Up-projected $\hat{z}$	Storage (B) / Top-1 (%)
1	2	8.36	7.68	128 / 73.33
4	8	45.84	43.42	512 / 76.99
16	32	70.07	68.75	2,048 / 81.18
64	128	78.08	77.74	8,192 / 81.77
256	512	80.17	79.99	32,768 / 81.76

Baseline	Storage (B)	Top-1 (%)
MAE-B	1,536	68.00
DREAM	2,048	72.70
Qwen3.5-2B	4,096	79.70

Table 8: Full results on ImageNet-1K linear-probing. For FLAT-SID, both the code $\hat{u}$ and the up-projected embeddings $\hat{z}$ share the same packed-ID storage.

trained model establishes a strong baseline (0.75 overall score). Fine-tuning the T2I decoder on top of frozen representations raises overall accuracy to 0.78, driven by substantial gains in spatial position and color attribution. Ultimately, jointly fine-tuning the encoder, quantizer, and decoder yields the highest overall performance (0.81).

Category	**Pre-trained**	**Fine-tune T2I Decoder**	**Fine-tune Encoder + Quantizer + T2I Decoder**
Overall	0.7541	0.7872	**0.8117**
Single Object	**0.9625**	0.9531	**0.9625**
Two Objects	**0.8561**	**0.8561**	0.8510
Counting	0.7344	0.7469	**0.7719**
Colors	0.8697	0.8697	**0.8723**
Position	0.4725	0.6325	**0.7375**
Color Attr.	0.4825	0.6650	**0.6750**

Table 9: GenEval benchmark performance across pre-training and fine-tuning checkpoints.

F Qualitative Galleries

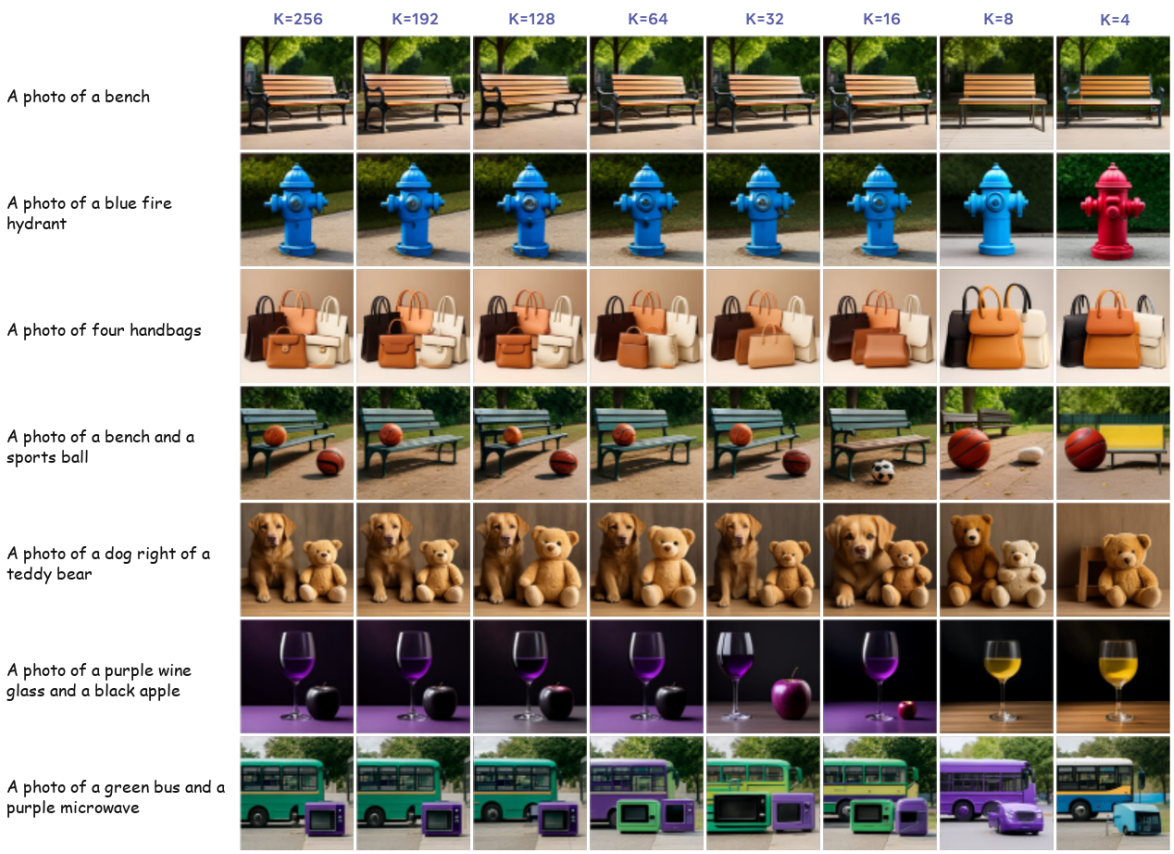
F.1 Image Generation with Different Prefix Lengths

Figure 12a shows text-to-image generation with different numbers of semantic IDs.

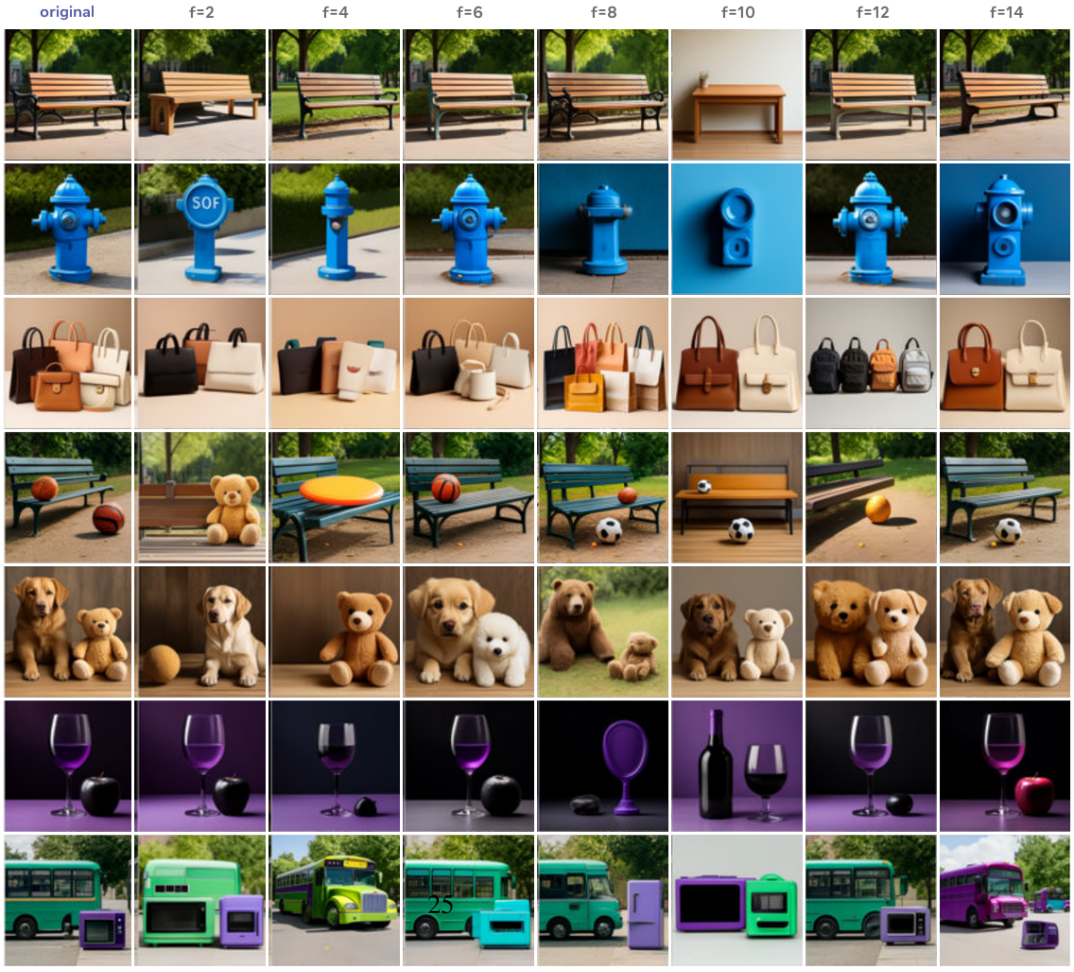
F.2 Generation after Bit Flipping

Each register's quantized latent is a 14-dimensional tensor with shape $[8, 2, 2, \ldots, 2]$, where dimensions 2 through 14 take binary values. We conduct an ablation study by flipping the bit along a specific dimension $f \in [2, 14]$. As shown

in Figure 12b, flipping the f-th dimension alters different fine-grained details while maintaining overall semantic coherence.



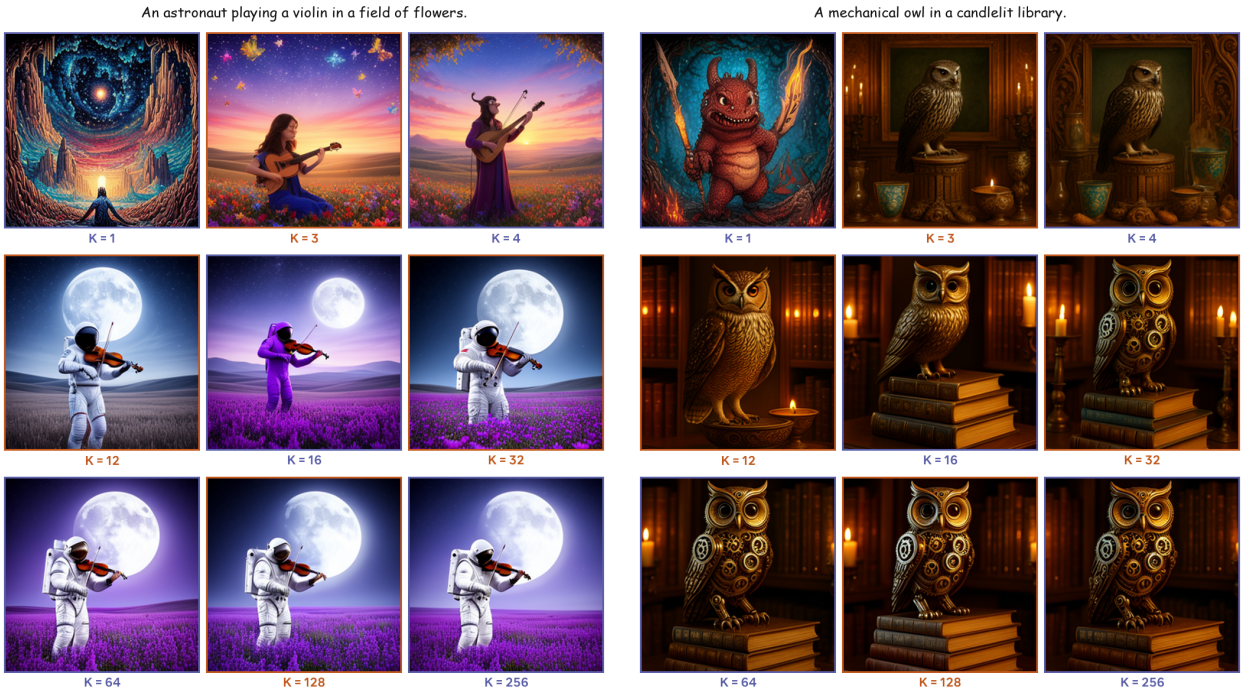
(a) Generation using a flexible number of semantic IDs K.



(b) Generation after flipping selected coordinates f of every register code.

F.3 Generalization to Unseen Prefix Lengths

FLAT-SID is trained using discrete prefix lengths $K \in \{1, 4, 16, 64, 256\}$. Figure 13 illustrates generation quality across intermediate, unseen prefix lengths without additional training or fine-tuning. Because the initial noise vector is held constant for a given prompt, variations across cells arise solely from the conditioning prefix. The model smoothly develops the target scene across unseen lengths and converges toward a stable visual composition as additional semantic IDs are provided.



(a) Astronaut and violin.

(b) Mechanical owl.

Figure 13: Text-to-image generation for unseen prefix lengths K. Blue borders mark lengths used during training; orange borders mark unseen lengths.

F.4 Captioning across Prefix Lengths

As shown in Figure 15 and Figure 16, we demonstrate that FLAT-SID generalizes to unseen prefix lengths K for image-to-text generation. Shorter prefixes primarily capture global semantic context, whereas longer prefixes progressively resolve specific objects and incorporate fine-grained, localized details.

F.5 Semantic Structure of Individual IDs

In Figure 17, we group images by semantic ID at each register position. Frequent semantic IDs in the first register position collect broad visual themes, such as animals, indoor fixtures, or food. Semantic IDs at the final register position are more context-dependent and do not form clusters independently.

F.6 Retrieval Successes and Failures

Figures 18 and 19 illustrate top-5 retrieval results at $K{=}256$. Near-miss failure cases remain semantically consistent with the query: candidate captions share the same overall dining context for the I2T top-1 miss, while retrieved images for a motorcycle query in T2I failure actually retrieve both the object and location.

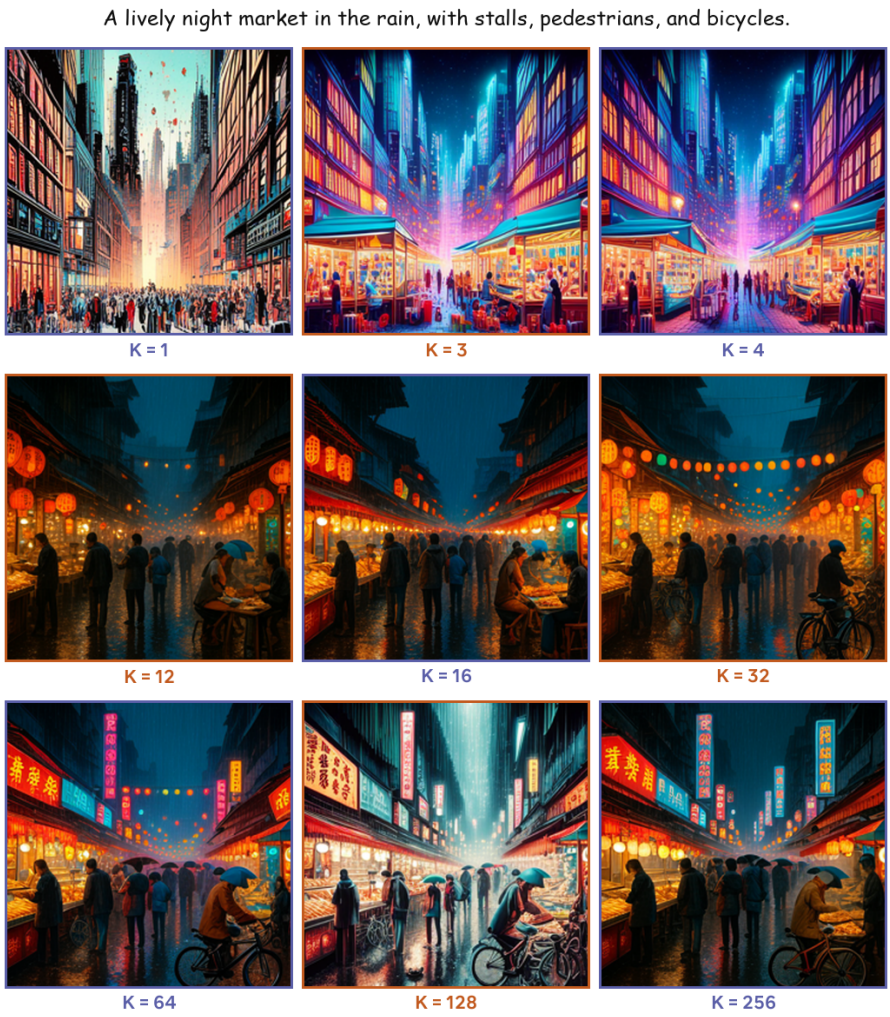


Figure 14: A night-market scene generated by the T2I decoder across trained and unseen prefix lengths K. Larger K introduces more details such as the market stalls, pedestrians and bicycles. while preserving the global semantics.

image	decoded caption	
	K=1	Sunset over a serene rural landscape with a winding road.
	K=3	A winding road through a lush, green landscape.
	K=4	A serene rural road stretches into the distance, flanked by lush greenery.
	K=12	A motorcyclist navigates a winding road through lush greenery.
	K=16	A motorcyclist navigates a winding road through lush greenery.
	K=32	A motorcyclist navigates a narrow, winding road surrounded by lush greenery.
	K=64	A motorcyclist navigates a narrow, winding road surrounded by lush greenery.
	K=128	A motorcyclist navigates a winding mountain road, surrounded by lush greenery.
	K=256	A motorcyclist navigates a winding mountain road, surrounded by lush greenery.

Figure 15: Image captioning of a motorcyclist scene across varying prefix lengths K. Shorter prefixes capture general road and landscape context, whereas the motorcyclist is explicitly identified from $K{=}12$ onward.

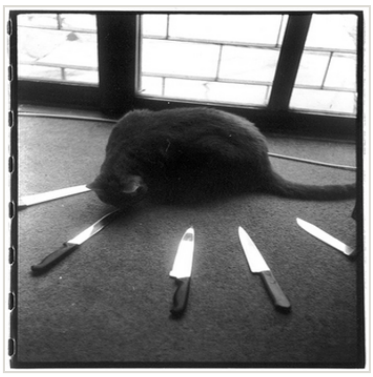
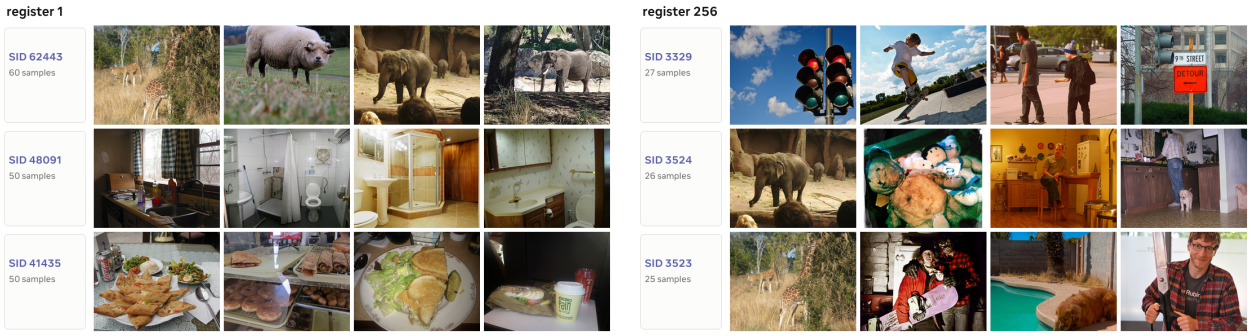
image	decoded caption	
	K=1	Adorable cat in a cozy sweater, looking up.
	K=3	A cat sits on a wooden surface, looking directly at the camera.
	K=4	A cat sits on a wooden surface, looking directly at the camera.
	K=12	Whimsical cat drawing in progress, capturing a playful moment.
	K=16	Curious cat explores a cluttered workspace.
	K=32	Curious cat explores a kitchen counter with a knife.
	K=64	Curious cat investigates knives on the floor.
	K=128	Curious cat surrounded by knives, black and white.
	K=256	Curious cat investigates kitchen knives by the window.

Figure 16: Image captioning of a cat surrounded by knives across varying prefix lengths K. Longer prefixes progressively resolve the central subject, surrounding knives, monochrome style, and overall spatial context.



(a) Frequent semantic IDs at the first register position.

(b) Frequent semantic IDs at the final register position.

Figure 17: Representative images mapped to high-frequency semantic IDs. Each row displays four sample images that share the same semantic ID at a specified register position.

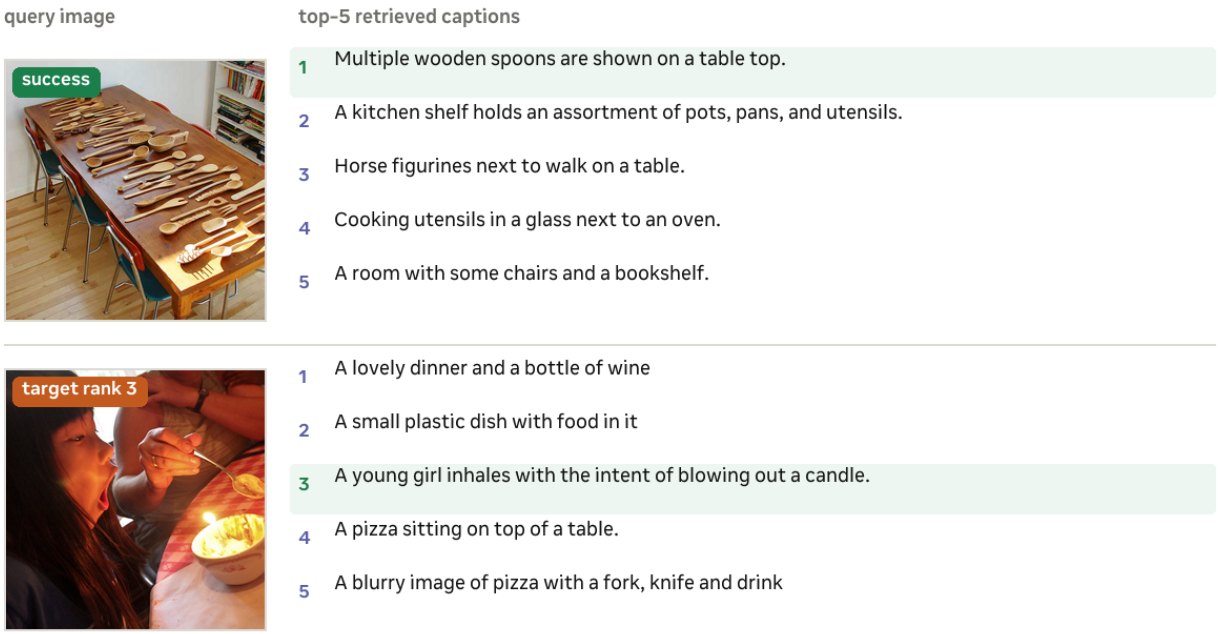


Figure 18: Image-to-text retrieval examples. The target caption is marked with green border; the bottom example is a near-miss failure with the target is ranked the 3rd.

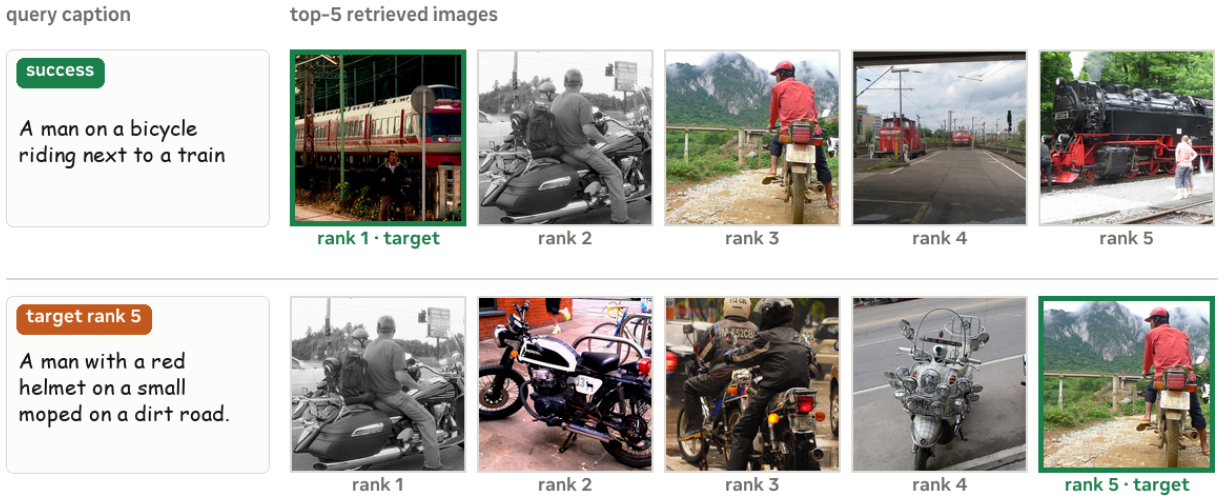


Figure 19: Text-to-image retrieval examples. The target image is marked with green border; the bottom example is a near-miss failure in which all top-ranked images are related to the query.