TAMING VISUAL TOKENIZER FRIENDLY FOR LARGE LANGUAGE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Extending pretrained language models to visual understanding and generation can compromise their existing language and reasoning capabilities. We investigate which properties of visual tokenization affect this trade-off and find that the organization of visual latents matters beyond their standalone visual quality. We develop LMTok, a 1D visual tokenizer whose latent organization is shaped by the next-token prediction objective of a pretrained language model, and compare it with a spatial 2D tokenizer under matched token lengths, codebook sizes, and downstream training budgets. Despite comparable reconstruction, generation, and visual understanding performance, training with LMTok helps the LLM better retain its language capabilities than training with spatial 2D tokens. With Qwen3-8B as the backbone, LMTok improves over spatial 2D tokenization by 1.72 points on MMLU-Pro, 3.40 points on MATH-500, and 5.40 points on MBPP, while maintaining comparable visual performance under unified training. This trend remains consistent when scaling the language-model backbone from 4B to 8B. Comparisons with existing 1D tokenizers and controlled ablations identify the autoregressive shaping as a key contributor to this improved compatibility. Overall, our results show that visual tokenizers should be evaluated not only by standalone visual quality, but also by how their sequence organization interacts with the autoregressive prior. The code and model will be released.

1 INTRODUCTION

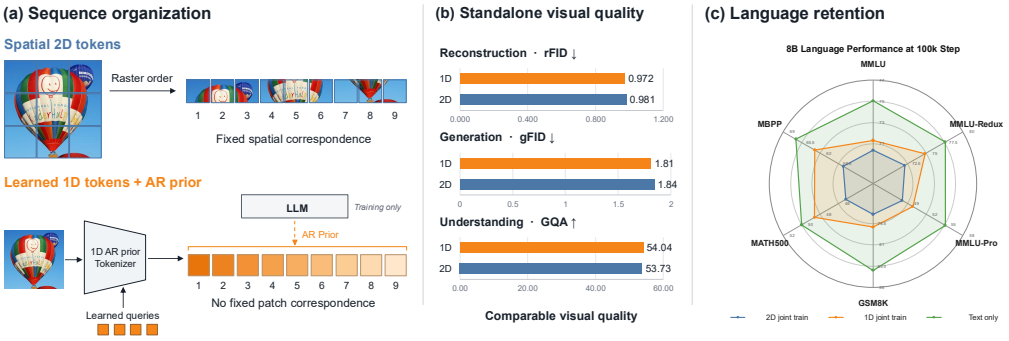

Figure 1: **Similar visual quality, different language retention.** Unlike spatial 2D tokens, learned 1D tokens allow flexible sequence organization that can be shaped by an AR prior. Despite comparable reconstruction, generation, and understanding performance, training with tokens from LMTok helps LLMs better retain their language and reasoning capabilities than training with 2D tokens.

Recent work has explored extending pretrained large language models to visual understanding and generation, bringing language and vision into a single model(Liu et al., 2025; Team, 2024; Wang et al., 2024; Wu et al., 2025a; Deng et al., 2025). An important goal of this extension is to retain the language and reasoning abilities acquired during pretraining while developing new visual capabilities. One approach represents images as discrete tokens and models them alongside text through autoregressive next-token prediction, retaining the language model's core modeling framework(Team, 2024; Wang et al., 2024). Yet even under a shared prediction objective, gains in visual

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.

capabilities can come at the expense of language performance(Wu et al., 2026b). Visual tokenization determines the sequences the language model must learn to process and predict, making it a natural consideration in this trade-off. This raises a question about the representations being unified: What properties of visual tokens help a pretrained language model acquire visual capabilities while better retaining its language and reasoning abilities?

Visual tokenizers determine not only what visual information is retained, but also how it is organized into sequences for unified multimodal modeling. Much of their development has focused on reconstruction, generation, and understanding (Esser et al., 2021; Ma et al., 2026; Chen et al., 2026), however, tokenizers with comparable visual performance can produce markedly different sequence structures. Recent work has explored compact, flexible 1D representations (Yu et al., 2024; Bachmann et al., 2025) and ways to better align visual tokenization with autoregressive prediction (Wu et al., 2026c; Wang et al., 2025). We argue that the value of learned 1D tokenization extends beyond compression: by removing the fixed correspondence between latent positions and a spatial grid, it opens up freedom to shape how visual information is distributed across a sequence. In this work, we investigate how this freedom can be used to make visual tokens more compatible with pretrained language models. We find that **1D tokens learned with an autoregressive (AR) prior better preserve language and reasoning capabilities under unified training** than spatial 2D tokens with comparable visual performance. Figure 1 illustrates this distinction. This finding motivates organizing visual representations around the language model's modeling preferences, rather than judging them by visual performance alone.

To this end, we introduce LMTok, a 1D visual tokenizer that separates spatial feature extraction from latent sequence organization. We retain a SigLIP2-initialized encoder and use learned queries to aggregate its spatial features through cross-attention, producing a quantized 1D sequence for image reconstruction. This removes the fixed correspondence between latent positions and the image grid, allowing visual information to be organized freely across the sequence. Such flexibility, however, does not by itself impose an ordering suited to autoregressive prediction. We therefore introduce a frozen language model as an AR prior during tokenizer training, using its next-token prediction objective to shape the learned sequence. To preserve semantic information without restoring a fixed spatial correspondence, we further introduce cross-alignment: spatial queries attend to the 1D tokens to recover a feature map supervised by SigLIP2. Together, these components allow the tokenizer to learn an AR-oriented sequence organization while retaining the visual information needed for reconstruction and semantic understanding.

We evaluate this design through controlled comparisons with spatial 2D tokenizers, matching token lengths, codebook sizes, and downstream training budgets. Under matched standalone evaluations, LMTok and the spatial 2D baseline achieve comparable reconstruction, visual understanding, and class-conditional image generation performance, with gFID scores of 1.81 and 1.84, respectively. Despite this similar visual capability, their effects on pretrained language performance differ substantially under unified training. With Qwen3-8B, the gap to text-only continued training is reduced from 8.18 to 6.46 points on MMLU-Pro and from 5.4 to 2.0 points on MATH-500. These results show that visual representations with similar standalone visual utility can differ substantially in their compatibility with a pretrained language model.

Our contributions are threefold:

- We identify language retention under unified multimodal training as an important evaluation dimension for visual tokenizers, showing that representations with comparable visual performance can induce substantially different degradation of a pretrained LLM's language and reasoning capabilities.
- We introduce LMTok, a learned 1D visual tokenizer that decouples latent positions from the image grid and shapes the resulting sequence with an autoregressive next-token objective, preserving visual semantics through 1D-to-2D cross-alignment.
- Under controlled, matched settings, LMTok consistently reduces language degradation relative to spatial 2D tokenization across Qwen3-4B and Qwen3-8B while maintaining comparable visual understanding and generation performance; controlled ablations further examine the roles of autoregressive supervision and semantic alignment.

2 Related Work

Unified multimodal models. Unified multimodal models extend language modeling to visual understanding and generation through different choices of representation and prediction objective. Pioneering works (Liu et al., 2025; Team, 2024; Wang et al., 2024; Wu et al., 2026b) explore autoregressive modeling of discrete visual and textual tokens within a shared sequence framework. Janus (Wu et al., 2025a) retains autoregressive generation while decoupling the visual encoding pathways for understanding and generation. Beyond discrete next-token prediction, continuous autoregressive approaches predict visual embeddings, as in Nexus-Gen (Zhang et al., 2025a), while hybrid approaches combine text autoregression with visual diffusion or flow matching, as in Transfusion (Zhou et al., 2025), JanusFlow (Ma et al., 2025), and BAGEL (Deng et al., 2025). These choices can overlap: Nexus-Gen uses a diffusion decoder, whereas Show-o (Xie et al., 2025a) combines text autoregression with discrete diffusion. Architectural alternatives include query-based connections to diffusion decoders in MetaQueries (Pan et al., 2025) and CLIP-feature diffusion in BLIP3-o (Chen et al., 2025b); MetaQueries can keep the multimodal language backbone frozen during generative adaptation. More recently, Tuna-2 (Liu et al., 2026) and SenseNova-U1/U1.5 (Diao et al., 2026b;a) explore encoder-free pixel-space modeling, replacing standalone visual encoders with patch-based interfaces. Our focus is on extending a pretrained LLM through a trainable, shared autoregressive backbone. Within this setting, we investigate how visual sequence design affects the balance between acquiring visual capabilities and retaining the language and reasoning capabilities acquired during pretraining.

Spatial visual tokenizers. Discrete visual tokenization builds on VQ-VAE (Van Den Oord et al., 2017) and VQGAN (Esser et al., 2021), which learn quantized spatial representations for reconstruction and generation. Subsequent work improves several distinct aspects of this interface: FSQ (Mentzer et al., 2024) and BSQ (Zhao et al., 2025) explore alternative quantization mechanisms, while VAR (Tian et al., 2024) organizes spatial codes across scales for next-scale prediction. AliTok (Wu et al., 2026c) addresses autoregressive compatibility by using a causal decoder to shape token dependencies while retaining a spatial token layout. Another line strengthens the semantic content of discrete representations. UniTok (Ma et al., 2026) investigates the role of quantization capacity in unifying understanding and generation. DualToken (Song et al., 2026) separates semantic and detail vocabularies, SemHiTok (Chen et al., 2026) introduces a semantic-guided hierarchical codebook, and TA-Tok (Han et al., 2026) constructs a text-aligned visual codebook using an LLM vocabulary. These advances improve quantization, semantic fidelity, and visual modeling without requiring a learned 1D latent. Similar visual quality can arise from different token organizations, which may interact differently with a pretrained LLM.

Learned 1D visual tokenizers. Learned 1D tokenizers relax the fixed correspondence between latent positions and spatial locations. TiTok (Yu et al., 2024) demonstrates compact image representations using a small set of learned latent tokens. FlexTok (Bachmann et al., 2025) and One-D-Piece (Miwa et al., 2025) explore flexible token budgets and ordered information allocation, while GigaTok (Xiong et al., 2025) and SemTok (Qu et al., 2026) investigate scaling and semantic supervision for 1D representations. Sequence structure has also received explicit attention: SEED (Ge et al., 2023) introduces causal visual tokenization for LLM integration, and NativeTok (Wu et al., 2026a) explores sequential token formation. Particularly relevant to our method, LARP (Wang et al., 2025) trains a holistic video tokenizer with an AR prior to improve autoregressive visual generation. EOSTok (Chu et al., 2026) jointly trains a 1D semantic tokenizer and an autoregressive image generator end to end, combining generation-aware reconstruction supervision with vision-foundation-model alignment. Inspired by LARP, we use a frozen pretrained language model to provide an AR prior, and shift the investigation from standalone visual generation to language capability retention under unified training. Our study specifically compares spatial 2D tokens and learned 1D tokens with an AR prior under matched token lengths, codebook sizes, and downstream training budgets. In our evaluated setting, comparable reconstruction and standalone visual understanding and generation coexist with different language-retention outcomes. The value of learned 1D structure in this setting is therefore not necessarily higher visual quality or shorter sequences, but the freedom to organize visual information around the modeling preferences of a pretrained LLM.

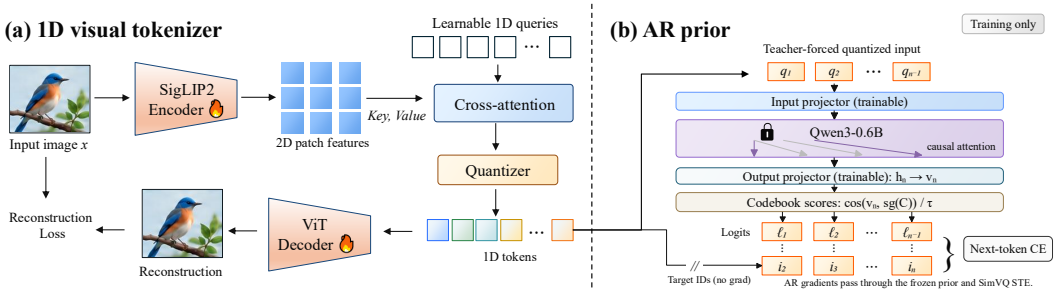

Figure 2: **Overview of LMTok. (a) Tokenization.** Learned queries aggregate spatial features into quantized 1D tokens for image reconstruction. **(b) AR prior.** A frozen pretrained language model provides next-token supervision to shape the 1D token sequence, with the AR-prior branch used only during tokenizer training.

3 METHOD

We introduce LMTok, a visual tokenizer that learns spatially decoupled 1D representations and shapes their organization with a pretrained language model's AR prior. Our design combines three components: a query-based tokenizer for image reconstruction, an AR objective that guides latent sequence organization, and cross-alignment that transfers spatial semantic supervision without imposing a fixed token-to-patch correspondence. The AR prior and semantic alignment branches are used only during tokenizer training. Figure 2 provides an overview.

3.1 LEARNING SPATIALLY DECOUPLED VISUAL TOKENS

We initialize the image encoder with SigLIP2 (Tschannen et al., 2025) and retain its architecture. Given an image x, the encoder produces a grid of P patch features $F = E(x)$. A set of N learnable queries then aggregates these features through three cross-attention layers, with the queries attending to the patch features as keys and values. The resulting representations are projected into the quantization space to obtain a latent sequence $Z = (z_1, \ldots, z_N)$. Unlike spatial 2D tokens, these latent positions are not tied to predefined image patches, allowing information to be distributed across a learned sequence. We quantize the latent sequence with SimVQ (Zhu et al., 2025), which parameterizes code vectors through a shared learnable linear projection of a Gaussian-initialized codebook embedding. Each normalized latent z_n is assigned to its nearest normalized code vector in a K-entry codebook:

$$i_n = \arg\min_k \|z_n - c_k\|_2^2, \qquad q_n = c_{i_n}. \tag{1}$$

This produces both discrete token indices $I = (i_1, \ldots, i_N)$ and quantized vectors $Q = (q_1, \ldots, q_N)$. A ViT decoder reconstructs the image from Q and a set of spatially positioned 2D mask tokens. Spatial positions are thus specified at the reconstruction stage without requiring a one-to-one correspondence between the 1D latents and image patches.

This architecture provides flexibility in latent organization, but reconstruction alone does not explicitly encourage an organization suited to autoregressive modeling. We next introduce an AR prior to guide this otherwise unconstrained sequence structure.

3.2 SHAPING THE 1D SEQUENCE WITH AN AR PRIOR

Reconstruction alone does not explicitly constrain the latent sequence to follow an autoregressive organization. Inspired by LARP (Wang et al., 2025), we introduce a pretrained language model as an auxiliary AR prior during tokenizer training. We use a frozen Qwen3-0.6B-Base backbone (Yang et al., 2025), replacing its text embedding and output head with trainable visual projections.

Codebook-anchored prediction. The quantizer provides code vectors as inputs and their corresponding discrete indices as next-token targets. During tokenizer training, both token assignments and code embeddings evolve. To avoid requiring an independently parameterized classification head

to track this changing correspondence, we anchor the prediction logits to the current codebook. An input projection maps quantized vectors into the language model, and an output projection maps its hidden states back into the codebook space:

$$h_{1:N-1} = f_\phi(P_{\text{in}}(q_{1:N-1})), \qquad \ell_{n,k} = \frac{\cos(P_{\text{out}}(h_n), \text{sg}(c_k))}{\tau}. \tag{2}$$

Here, $\ell_{n,k}$ is the logit for predicting code k at position $n+1$ from the prefix $q_{1:n}$. The backbone f_ϕ is frozen, P_{in} and P_{out} are trainable linear projections, τ is a temperature, and sg denotes stop-gradient. We optimize teacher-forced next-token prediction:

$$\mathcal{L}_{\text{AR}} = \frac{1}{N-1} \sum_{n=1}^{N-1} \text{CE}(\ell_n, i_{n+1}). \tag{3}$$

At each training step, the logits are computed against the quantizer's current codebook, with gradients stopped through these codebook embeddings. This blocks a direct route by which the AR objective could reshape the codebook to favor fewer codes and thereby simplify the prediction task, rather than improve sequence organization.

Backpropagation to the tokenizer. AR supervision reaches the continuous latents through the quantizer's existing straight-through estimator:

$$q_n = z_n + \text{sg}(c_{i_n} - z_n). \tag{4}$$

Its forward value is the selected code vector, while its backward derivative with respect to the normalized latent z_n is the identity. Freezing the language-model parameters does not block gradients with respect to its inputs: the AR loss propagates through the visual projections and frozen backbone, then through the quantizer's STE to the encoder and learned queries. Discrete target indices carry no gradient. Together, the input-side STE and output-side codebook detachment prevent the AR loss from directly updating the codebook, which is optimized through separate quantization objectives. Our ablations (Table 4) show that adding the same AR prior to spatial 2D tokens does not improve language retention, consistent with their fixed token-to-patch correspondence and raster order limiting the freedom to reorganize the sequence.

3.3 Semantic Alignment without Spatial Binding

To preserve visual semantics, we distill features from a frozen SigLIP2 teacher. Directly aligning the 1D latent sequence with flattened teacher features would impose a fixed token-to-patch correspondence, reintroducing the spatial binding that our tokenizer is designed to relax. Instead, we apply semantic supervision to a spatial feature map recovered from the sequence through cross-alignment, rather than to individual latent positions.

Spatial positional queries recover a 2D semantic map from the quantized sequence Q. We additionally align the encoder's patch features F with the same teacher, anchoring its spatial representations to the pretrained features to limit representational drift and support stable early training. The two branches are optimized with a weighted cosine-distillation objective:

$$\begin{gathered} T = \text{sg}(E_{\text{tea}}(x)), \qquad \widehat{T} = \text{CrossAttn}(S, P_{\text{align}}(Q)), \\ \mathcal{L}_{\text{sem}} = \lambda d_{\cos}(F, T) + (1-\lambda) d_{\cos}(\widehat{T}, T). \end{gathered} \tag{5}$$

Here, $d_{\cos}$ denotes cosine distance averaged over spatial positions. P_{align} is a separate linear projection to the teacher feature dimension. CrossAttn comprises two residual cross-attention layers followed by layer normalization, using spatial positional queries S as queries and $P_{\text{align}}(Q)$ as keys and values.

The lower-weight encoder term anchors spatial feature extraction, while the cross-alignment term requires semantic information to remain recoverable from the quantized sequence. Neither imposes a fixed correspondence between individual 1D latents and teacher patches. Architectural details and loss weights are provided in Appendices D.3 and A.1.

3.4 Integration into Unified Autoregressive Models

We integrate LMTok into a modern large language model, Qwen3 (Yang et al., 2025), by extending its vocabulary with discrete visual tokens. The tokenizer remains frozen during unified training, encoding images into 1D token sequences that are modeled alongside text through the same autoregressive next-token prediction objective. We jointly train on text-only, text-to-image, and image-to-text data, enabling language modeling, visual generation, and visual understanding within a shared backbone. Generated visual tokens are decoded into images using the tokenizer's decoder. The auxiliary AR prior and semantic alignment branches are used only for tokenizer training and are not transferred to the unified model. We use the same downstream integration and training recipe for the 1D and 2D tokenizers; implementation details are provided in Appendix A.2.

4 Experiments

Our experiments distinguish the standalone visual utility of a tokenizer from its compatibility with a pretrained language model. We first examine language retention under unified training, evaluating text-to-image generation alongside language performance, and then assess the tokenizers in standalone visual tasks.

4.1 Experimental Setup

Visual tokenizers. We train LMTok and a spatial 2D baseline on ImageNet for 50 epochs at 256×256 resolution. Both use SimVQ (Zhu et al., 2025) with 256 tokens and a 16,384-entry codebook, sharing the training data, optimization settings, and weights of the reconstruction and semantic objectives. The 2D baseline quantizes spatial encoder features directly and applies spatial semantic alignment, without learned 1D queries or cross-alignment. Its tokens retain their grid correspondence and are serialized in raster order. The main 2D baseline does not use an AR prior; we separately ablate latent structure and AR supervision to examine their effects.

Unified training. We initialize the unified models from Qwen3-4B-Base and Qwen3-8B-Base (Yang et al., 2025) and freeze the visual tokenizers. Training combines text-only language modeling, text-to-image generation, and image-to-text understanding, using a 30B-token text corpus, 40M synthetic image–text pairs from Infinity (Han et al., 2025), and LLaVA-OneVision-1.5 SFT data (An et al., 2025), respectively. We train for 100K steps with an effective batch size of 512, a context length of 4,096, and a cosine learning-rate schedule starting at 10^{-4}. Paired 1D/2D runs use the same downstream training settings. We include both the original pretrained models and text-only continued-training references; the latter use the same number of training steps and comparable text-token exposure.

Evaluation. We evaluate language, visual understanding, and text-to-image generation using the benchmarks in Table 1. All unified-model results are measured directly from the jointly trained checkpoints, without additional task-specific fine-tuning. To assess standalone visual performance separately, we evaluate tokenizer reconstruction, train class-conditional generators following LlamaGen (Sun et al., 2024) for 400 epochs, and train understanding models using the LLaVA-1.5 recipe (Liu et al., 2024) at 256×256 resolution. Backbone configurations are reported in the corresponding tables. Detailed training and evaluation protocols are provided in Appendix A.

4.2 Language Retention under Unified Training

Better language retention under unified training. Compared with matched text-only continued training, unified training reduces language performance across all six evaluated benchmarks in Table 1, consistent with the language–vision trade-off previously reported in Liquid (Wu et al., 2026b). The extent of this degradation, however, depends on visual tokenization: LMTok preserves more language capability than the spatial 2D baseline on every benchmark at both 4B and 8B. At 8B, it improves MMLU-Pro, MATH-500, and MBPP by 1.72, 3.4, and 5.4 percentage points, respectively. Relative to the text-only reference, these gains reduce the performance gaps on MMLU-Pro and MATH-500 by approximately 21% and 63%. Thus, while neither representation eliminates the trade-off, LMTok consistently mitigates it across the two model scales.

Table 1: **Language retention and visual performance under unified training.** Qwen3-Base denotes the pretrained backbone; Text-only denotes continued training on text alone. LMTok (1D) and Spatial 2D denote unified training with the respective visual tokenizers under matched downstream settings. All metrics are higher-is-better. Dashes indicate non-applicable evaluations.

Scale	Model / Setting	Language Tasks						Visual Und.			Visual Gen.	
		MMLU	MMLU-Redux	MMLU-Pro	MATH-500	GSM8K	MBPP	GQA	POPE	MME-P	DPG	GenEval
4B	Qwen3-4B-Base	73.19	71.78	51.25	53.80	83.78	67.2	–	–	–	–	–
	Text-only	70.20	67.11	50.67	47.0	80.59	60.2	–	–	–	–	–
	2D-Baseline	65.73	63.22	43.70	38.8	73.91	53.8	49.91	79.33	1253.8	84.2	**67.6**
	LMTok (ours)	**66.37**	**64.87**	**46.46**	**43.8**	**75.96**	**56.0**	**51.87**	**82.51**	**1310.4**	**84.5**	67.1
8B	Qwen3-8B-Base	76.95	80.04	56.12	57.00	86.13	69.0	–	–	–	–	–
	Text-only	75.02	77.59	56.29	50.6	82.25	65.4	–	–	–	–	–
	2D-Baseline	70.38	72.10	48.11	45.2	77.40	57.6	50.33	**80.99**	1271.9	**85.9**	68.3
	LMTok (ours)	**71.31**	**74.84**	**49.83**	**48.6**	**78.92**	**63.0**	**51.59**	80.92	**1396.9**	85.3	**68.5**

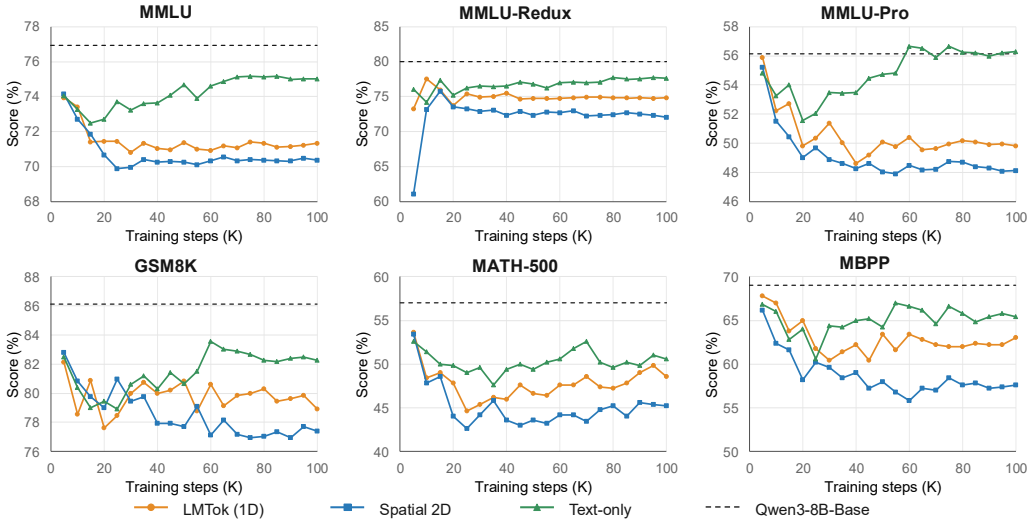

Figure 3: **Language performance throughout 8B training.** We compare unified training with LMTok (orange) and spatial 2D tokens (blue) against matched text-only continued training (green). Black dashed lines indicate the original Qwen3-8B-Base scores. The horizontal axis reports optimizer steps without rescaling.

Visual performance under unified training. Better language retention is accompanied by generally stronger visual understanding. LMTok improves all three understanding metrics at 4B and achieves higher GQA and MME-P scores at 8B, with nearly identical POPE performance. Better preservation of the language backbone may contribute to these understanding gains, although differences in standalone visual representations, evaluated in Section 4.3, also remain a possible factor. Meanwhile, text-to-image generation remains comparable: at 8B, LMTok scores 85.3 versus 85.9 on DPG and 68.5 versus 68.3 on GenEval. Overall, the language-retention gains are not accompanied by a consistent decline in the evaluated visual capabilities.

Language retention throughout training. Figure 3 tracks language performance during 8B training. Both text-only and unified training show an early decline on several benchmarks, followed by partial recovery or stabilization. Despite these fluctuations, LMTok outperforms the spatial 2D baseline on MMLU-Pro, MATH-500, and MBPP at every evaluated checkpoint, with advantages across all six benchmarks toward the end of training. These trajectories show that the language-retention gains reflect a sustained training trend rather than an outcome specific to a selected checkpoint.

Table 2: ImageNet 256×256 reconstruction and class-conditional generation. Our 343M-generator rows provide a matched 1D/2D comparison; additional rows evaluate a larger generator and MCQ. Published results are contextual, not budget-matched.

Tokenizer	Layout	Tokens	rFID↓	Generator	Size	gFID↓
Autoregressive next-token prediction						
VQGAN (Esser et al., 2021)	2D	256	4.98	Transformer	1.4B	15.78*
LlamaGen (Sun et al., 2024)	2D	256	2.19	LlamaGen	1.4B	3.09
IBQ (Shi et al., 2025)	2D	256	1.37	IBQ-XXL	2.1B	2.05
FlexTok-d18-d28 (Bachmann et al., 2025)	1D	128	1.20	LlamaGen	1.3B	1.89
FlexTok-d18-d28	1D	256	1.08	LlamaGen	1.3B	2.45
GigaTok-B-L (Xiong et al., 2025)	1D	256	0.81	LlamaGen	111M	3.26
GigaTok-B-L	1D	256	0.81	LlamaGen	1.4B	2.03*
Other generation paradigms (context)						
VAR (Tian et al., 2024)	Multi-scale	680	1.00	VAR-d30	2.0B	1.92
TiTok-S (Yu et al., 2024)	1D	128	1.71	MaskGIT-UViT-L	287M	1.97
Our comparison: 50 tokenizer epochs / 400 C2I epochs						
Spatial 2D baseline	2D	256	0.98	LlamaGen	343M	1.84
LMTok (ours)	1D	256	0.97	LlamaGen	343M	1.81
LMTok (ours)	1D	256	0.97	LlamaGen	775M	1.61
LMTok-MCQ (ours)	1D	256	0.35	LlamaGen	1.4B	1.71

*Without classifier-free guidance (CFG), as marked in GigaTok. Other entries retain the reported sampling settings, not a common CFG scale. Size denotes generator parameters.

Table 3: Standalone visual understanding at 256×256 resolution using the LLaVA-1.5 training recipe. Results are grouped by language backbone; all metrics are higher-is-better. LMTok-MCQ extends LMTok with multi-codebook quantization to increase its effective codebook capacity.

Visual representation	Backbone	POPE	MME-P	SEED	GQA
SigLIP2 (Tschannen et al., 2025)	Vicuna-7B	84.88	1524.1	67.13	62.98
LlamaGen (Sun et al., 2024)	Vicuna-7B	65.60	716.8	35.00	39.80
VILA-U (Wu et al., 2025b)	Vicuna-7B	81.60	1311.6	56.90	55.30
SDE (MUSE-VL) (Xie et al., 2025b)	Vicuna-7B	77.30	1240.0	56.70	58.00
SemHiTok (Chen et al., 2026)	Vicuna-7B	82.50	1355.8	62.90	60.30
2D-Baseline (ours)	Vicuna-7B	78.65	1229.3	48.51	53.73
LMTok (ours)	Vicuna-7B	81.17	1246.6	50.70	54.04
LMTok-MCQ (ours)	Vicuna-7B	85.25	1483.4	65.35	61.67
SigLIP2 (Tschannen et al., 2025)	Qwen2.5-7B-IT	85.19	1532.5	70.74	63.09
TokLIP (Lin et al., 2025)	Qwen2.5-7B-IT	81.20	1346.8	59.80	57.40
LMTok-MCQ (ours)	Qwen2.5-7B-IT	86.07	1500.3	68.79	61.60

4.3 Standalone Visual Performance

We evaluate the same LMTok and spatial 2D baseline tokenizers used in Table 1 on standalone reconstruction, class-conditional generation, and visual understanding, aim to assess whether differences in visual capability account for the language-retention gap observed under unified training.

Reconstruction and generation. Table 2 compares reconstruction and class-conditional generation on ImageNet 256×256. Under matched settings, LMTok and the spatial 2D baseline achieve similar reconstruction (rFID 0.97 vs. 0.98) and generation quality (gFID 1.81 vs. 1.84 with the same 343M generator). Scaling the generator to 775M improves LMTok's gFID to 1.61, yielding competitive generation performance. These results suggest that a Transformer can effectively model either raster-ordered 2D tokens or learned 1D sequences for standalone generation. The distinction becomes more consequential under unified training, where visual sequences must be modeled alongside language within a shared pretrained backbone. **Thus, an organization that supports strong standalone generation need not be equally compatible with joint language–vision modeling.** Detailed C2I results and ablations are provided in Appendices C and D.5.

Visual understanding. Table 3 evaluates standalone visual understanding using the LLaVA-1.5 training recipe. The single-codebook LMTok and spatial 2D baseline exhibit broadly comparable understanding performance, with benchmark-dependent differences. We additionally follow the multi-codebook quantization setting of UniTok (Ma et al., 2026) to expand the effective codebook capacity, obtaining LMTok-MCQ. This variant achieves visual understanding performance broadly comparable to the original SigLIP2 features. These results show that our learned 1D organization remains compatible with increased representational capacity: removing fixed spatial correspondence does not preclude strong visual understanding. Additional evaluations of LMTok-MCQ, including its effect on language retention under unified training, are provided in Appendix B.1.

4.4 Ablation and Analysis

Latent organization and the AR prior. Table 4 examines whether flexible latent organization alone is sufficient for compatibility with a pretrained language model. Without an AR prior, the learned 1D sequence lacks explicit supervision for autoregressive organization; spatial flexibility alone does not make it AR-native. Unified training with this variant incurs substantial language degradation, even exceeding that of the spatial 2D baseline. Conversely, adding an AR loss to the 2D tokenizer does not remove its fixed token-to-patch correspondence or raster-order serialization, and does not improve language retention in our experiments. This pattern is consistent with our motivation: learned 1D latents provide freedom to reorganize visual information under AR supervision, whereas spatial 2D tokens retain a fixed spatial organization. To distinguish the role of autoregressive supervision from model-family-specific pretrained biases, we also train the tokenizer jointly with a randomly initialized 100M autoregressive Transformer (AR-100M). It improves language retention over the no-prior variant and performs comparably to the frozen prior, suggesting that gains stem from AR-guided sequence organization rather than model-family-specific pretrained biases.

Table 4: Latent structure and AR prior: language scores after unified training (↑). Top: external 1D tokenizers with their original training recipes (—). Bottom: our variants without a prior (✗), with frozen Qwen3-0.6B (✓), or with AR-100M, a randomly initialized 100M Transformer jointly trained with the tokenizer.

Tokenizer	AR prior	MMLU-Pro	MATH-500	GSM8K	MBPP
TiTok	—	42.04	38.2	67.70	53.2
GigaTok	—	42.17	37.2	71.11	50.0
Spatial 2D	✗	43.70	38.8	73.91	53.8
Spatial 2D	✓	42.25	36.0	67.02	49.8
Learned 1D	✗	41.24	36.2	69.82	49.4
LMTok (1D)	✓	**46.46**	**43.8**	**75.96**	56.0
Learned 1D	AR-100M	45.39	42.1	73.56	**56.5**

Validation with existing 1D tokenizers. We also evaluate TiTok (Yu et al., 2024) and GigaTok (Xiong et al., 2025) under the same downstream unified-training setting. Neither uses the pretrained-language-model AR prior employed in LMTok. Both exhibit substantial language degradation, with retention no better than the spatial 2D baseline on the reported benchmarks. This extends our observation beyond our own tokenizer variants: a 1D representation alone does not ensure compatibility with autoregressive language modeling.

Additional details and analyses. The supplementary material provides training details, hyperparameters, and evaluation protocols in Appendix A. Appendix B presents further analyses of unified training and examines visual latent organization through progressive token replacement. Appendix C examines AR-prior configurations and interfaces, while Appendix D reports ablations on semantic supervision, cross-alignment, codebook capacity, and variable-length representations.

5 Conclusion

Visual tokenizers with comparable standalone quality can yield different language retention under unified training. LMTok reduces language degradation relative to spatial 2D tokenization. Our ablations support AR-guided organization of flexible 1D sequences, rather than a specific pretrained prior, as a source of this benefit. These findings motivate designing visual representations for both visual utility and compatibility with language models.

AI USE STATEMENT

Generative AI tools were used solely to assist with language editing and writing, including improving grammar and readability of the manuscript. They were not used to formulate research hypotheses, design the methodology or experiments, implement the proposed methods, analyze or interpret experimental results, or generate scientific claims or conclusions. All AI-assisted text was reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of our results. Detailed implementation settings, training configurations, data preprocessing procedures, and hyperparameter choices are provided in Appendix A, with evaluation protocols in Appendix A.3. In addition, we provide a simplified implementation of the visual tokenizer updated with gradients from the autoregressive (AR) objective, illustrating the core optimization procedure used in our method. Together, these materials provide the information necessary to reproduce the main experimental results reported in this work.

REFERENCES

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, et al. Llava-onevision-1.5: Fully open framework for democratized multimodal training. *arXiv preprint arXiv:2509.23661*, 2025.

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025.

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

Jiuhai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, et al. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset, 2025. *URL https://arxiv. org/abs/2505.09568*, 2025b.

Zisheng Chen, Chunwei Wang, Runhui Huang, Hongbin Xu, Xiuwei Chen, Jun Zhou, Jianhua Han, Hang Xu, and Xiaodan Liang. Semhitok: A unified image tokenizer via semantic-guided hierarchical codebook for multimodal understanding and generation. In *International Conference on Learning Representations*, volume 2026, pp. 44597–44620, 2026.

Wenda Chu, Bingliang Zhang, Jiaqi Han, Yizhuo Li, Linjie Yang, Yisong Yue, and Qiushan Guo. End-to-end autoregressive image generation with 1d semantic tokenizer. In *arXiv preprint arXiv:2605.00503*, 2026.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining, 2025. *URL https://arxiv. org/abs/2505.14683*, 8, 2025.

Haiwen Diao, Jiahao Wang, Chenjing Ding, Hanming Deng, Jiangnan Chen, Ruixi Zhang, Ruohui Wang, Wenwen Tong, Xiangyu Fan, Yubo Wang, et al. Sensenova-u1. 5: Towards native unified visual intelligence. *arXiv preprint arXiv:2609.11929*, 2026a.

Haiwen Diao, Penghao Wu, Hanming Deng, Jiahao Wang, Shihao Bai, Silei Wu, Weichen Fan, Wenjie Ye, Wenwen Tong, Xiangyu Fan, et al. Sensenova-u1: Unifying multimodal understanding and generation with neo-unify architecture. *arXiv preprint arXiv:2605.12500*, 2026b.

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.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

Yuying Ge, Yixiao Ge, Ziyun Zeng, Xintao Wang, and Ying Shan. Planting a seed of vision in large language model. *arXiv preprint arXiv:2307.08041*, 2023.

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

Jiaming Han, Hao Chen, Yang Zhao, Hanyu Wang, Qi Zhao, Ziyan Yang, Hao He, Xiangyu Yue, and Lu Jiang. Vision as a dialect: Unifying visual understanding and generation via text-aligned representations. In *Advances in Neural Information Processing Systems*, volume 38, pp. 158430–158459, 2026.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity∞: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15733–15744. IEEE, 2025.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

Haokun Lin, Teng Wang, Yixiao Ge, Yuying Ge, Zhichao Lu, Ying Wei, Qingfu Zhang, Zhenan Sun, and Ying Shan. Toklip: Marry visual tokens to clip for multimodal comprehension and generation. *arXiv preprint arXiv:2505.05422*, 2025.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with blockwise ringattention. In *International Conference on Learning Representations*, volume 2025, pp. 45953–45977, 2025.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26286–26296. IEEE, 2024.

Zhiheng Liu, Weiming Ren, Xiaoke Huang, Shoufa Chen, Tianhong Li, Mengzhao Chen, Yatai Ji, Sen He, Jonas Schult, Belinda Zeng, et al. Tuna-2: Pixel embeddings beat vision encoders for multimodal understanding and generation. *arXiv preprint arXiv:2604.24763*, 2026.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*, 2024.

Chuofan Ma, Yi Jiang, Junfeng Wu, Jihan Yang, Xin Yu, Zehuan Yuan, Bingyue Peng, and Xiaojuan Qi. Unitok: A unified tokenizer for visual generation and understanding. In *Advances in Neural Information Processing Systems*, volume 38, pp. 129274–129297, 2026.

Yiyang Ma, Xingchao Liu, Xiaokang Chen, Wen Liu, Chengyue Wu, Zhiyu Wu, Zizheng Pan, Zhenda Xie, Haowei Zhang, Xingkai Yu, et al. Janusflow: Harmonizing autoregression and rectified flow for unified multimodal understanding and generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7739–7751. IEEE, 2025.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *International Conference on Learning Representations*, volume 2024, pp. 51772–51783, 2024.

Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-d-piece: Image tokenizer meets quality-controllable compression. In *arXiv preprint arXiv:2501.10064*, 2025.

Xichen Pan, Satya Narayan Shukla, Aashu Singh, Zhuokai Zhao, Shlok Kumar Mishra, Jialiang Wang, Zhiyang Xu, Jiuhai Chen, Kunpeng Li, Felix Juefei-Xu, et al. Transfer between modalities with metaqueries. *arXiv preprint arXiv:2504.06256*, 2025.

Yunpeng Qu, Kaidong Zhang, Yukang Ding, Ying Chen, and Jian Wang. Semantic one-dimensional tokenizer for image reconstruction and generation. *arXiv preprint arXiv:2603.16373*, 2026.

Fengyuan Shi, Zhuoyan Luo, Yixiao Ge, Yujiu Yang, Ying Shan, and Limin Wang. Scalable image tokenization with index backpropagation quantization. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16037–16046. IEEE, 2025.

Wei Song, Yuran Wang, Zijia Song, Yadong Li, Zenan Zhou, Long Chen, Xu Jhua, Jiaqi Wang, and Kaicheng Yu. Dualtoken: Towards unifying visual understanding and generation with dual visual vocabularies. In *International Conference on Learning Representations*, volume 2026, pp. 8086–8105, 2026.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in neural information processing systems*, volume 37, pp. 84839–84865, 2024.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in neural information processing systems*, volume 30, 2017.

Hanyu Wang, Saksham Suri, Yixuan Ren, Hao Chen, and Abhinav Shrivastava. Larp: Tokenizing videos with a learned autoregressive generative prior. In *International Conference on Learning Representations*, volume 2025, pp. 60671–60697, 2025.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

Bin Wu, Mengqi Huang, Weinan Jia, and Zhendong Mao. Nativetok: Native visual tokenization for improved image generation. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1991–1995. IEEE, 2026a.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12966–12977. IEEE, 2025a.

Junfeng Wu, Yi Jiang, Chuofan Ma, Yuliang Liu, Hengshuang Zhao, Zehuan Yuan, Song Bai, and Xiang Bai. Liquid: Language models are scalable and unified multi-modal generators. *International Journal of Computer Vision*, 134(1):39, 2026b.

Pingyu Wu, Kai Zhu, Yu Liu, Longxiang Tang, Jian Yang, Yansong Peng, Wei Zhai, Yang Cao, and Zheng-Jun Zha. Towards sequence modeling alignment between tokenizer and autoregressive model. In *International Conference on Learning Representations*, volume 2026, pp. 132194–132217, 2026c.

Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. In *International Conference on Learning Representations*, volume 2025, pp. 93620–93638, 2025b.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *International Conference on Learning Representations*, volume 2025, pp. 28240–28264, 2025a.

Rongchang Xie, Chen Du, Ping Song, and Chang Liu. Muse-vl: Modeling unified vlm through semantic discrete encoding. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24135–24146. IEEE, 2025b.

Tianwei Xiong, Jun Hao Liew, Zilong Huang, Jiashi Feng, and Xihui Liu. Gigatok: Scaling visual tokenizers to 3 billion parameters for autoregressive image generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18770–18780. IEEE, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 128940–128966, 2024.

Hong Zhang, Zhongjie Duan, Xingjun Wang, Yuze Zhao, Weiyi Lu, Zhipeng Di, Yixuan Xu, Yingda Chen, and Yu Zhang. Nexus-gen: Unified image understanding, generation, and editing via prefilled autoregression in shared embedding space. *arXiv preprint arXiv:2504.21356*, 2025a.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, et al. Lmms-eval: Reality check on the evaluation of large multimodal models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 881–916, 2025b.

Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. In *International Conference on Learning Representations*, volume 2025, pp. 90844–90868, 2025.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *International Conference on Learning Representations*, volume 2025, pp. 6446–6469, 2025.

Yongxin Zhu, Bocheng Li, Yifei Xin, Zhihua Xia, and Linli Xu. Addressing representation collapse in vector quantized models with one linear layer. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22968–22977. IEEE, 2025.

SUPPLEMENTARY MATERIAL

This supplement is organized into four parts. Appendix A details visual tokenizer training, unified multimodal training, and evaluation protocols. Appendix B presents additional training trajectories, qualitative image-generation comparisons, progressive token replacement, and codebook-utilization analysis. Appendix C examines the autoregressive prior, including its effect on visual generation, model configuration, input/output interfaces, and auxiliary reconstruction. Appendix D studies semantic supervision, cross-alignment, codebook capacity, and variable-length representations.

A IMPLEMENTATION AND EVALUATION DETAILS

A.1 VISUAL TOKENIZER TRAINING

Architecture and controlled comparisons. We train both tokenizers on ImageNet at 256×256 resolution, using 256 visual tokens and a codebook of 16,384 entries. They share the same SigLIP2-initialized encoder and reconstruction decoder architectures. The spatial 2D baseline directly quantizes encoder patch features and aligns the resulting code vectors with the corresponding teacher features using a cosine loss. LMTok instead uses learned 1D queries and cross-alignment. Its semantic objective combines distillation of the encoder patch features before quantization with distillation of a spatial feature map recovered from the quantized 1D codes, weighted by $1/3$ and $2/3$, respectively. The 2D baseline uses a single post-quantization distillation term with weight 1. Both therefore use the same overall semantic-loss weight, whose effect is studied in Appendix D.1. The main 2D baseline has no AR prior. Our factorial ablation changes only the presence of AR supervision within each architecture: adding the same prior and next-token cross-entropy objective to the raster-ordered 2D sequence, or removing the prior branch from LMTok, while keeping all other settings unchanged.

Quantization and AR supervision. We use SimVQ (Zhu et al., 2025) with 256-dimensional codes, a frozen Gaussian-initialized base codebook, and a trainable linear codebook projection. Latents and effective code vectors are L2-normalized. The commitment and entropy loss weights are 0.2 and 0.01, respectively. The AR prior is a frozen Qwen3-0.6B-Base model (Yang et al., 2025). Its input and output adapters are single linear layers with biases, mapping from 256 to 1,024 dimensions and back, respectively. Next-token logits are computed from normalized output vectors and the detached, normalized current codebook, with temperature 1. The AR loss has weight 0.05 from the start of training, without warmup. Teacher forcing predicts tokens 2 through 256 from tokens 1 through 255, without an additional class or beginning-of-sequence token.

Optimization and reconstruction. We use AdamW with an effective batch size of 256, an initial learning rate of 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 10^{-4}. The encoder learning rate is reduced by a factor of 10. We apply cosine learning-rate decay without warmup, do not use EMA, and use the checkpoint at 50K training steps for evaluation. Following the reconstruction recipe of SoftVQ-VAE (Chen et al., 2025a), the reconstruction objective combines pixelwise L1 loss, VGG-based LPIPS, and an adversarial loss, with final weights of 1, 1, and 0.2, respectively. The LPIPS weight increases linearly from 0 to 1 over the first 10,000 steps, while L1 supervision is active throughout training. Adversarial training uses a DINO discriminator with a hinge-loss objective and a generator loss of $-\mathbb{E}[D(\hat{x})]$, without adaptive adversarial weighting.

A.2 UNIFIED MULTIMODAL TRAINING

Model initialization and visual interface. We initialize the unified models from Qwen3-4B-Base and Qwen3-8B-Base (Yang et al., 2025) and keep the visual tokenizer frozen throughout training. Each image is encoded into 256 discrete tokens, and the language vocabulary is extended with visual codes and boundary tokens. A two-layer MLP projects the frozen codebook into the language model's hidden space. The projected vectors are mean-centered across the codebook and shared between visual input embeddings and output classification weights. Before joint training, we warm up the projector for 2,000 updates at a learning rate of 10^{-3}, using sampled T2I and I2T data with equal task sampling probabilities while keeping the language model frozen. This stage is not included in the subsequent 100K joint-training updates. During joint training, both the projector and

all language-model parameters are optimized. The tokenizer's auxiliary AR prior and semantic-alignment branches are not used, and no prior-model weights are transferred to the downstream backbone. Both tokenizers use the same integration procedure.

Training data and task scheduling. Training combines a 30B-token text corpus, 40M synthetic image–text pairs from Infinity (Han et al., 2025), and LLaVA-OneVision-1.5 (An et al., 2025) SFT data for text-only language modeling, T2I generation, and I2T understanding, respectively. Following the modality-grouped training strategy of DeepSeek-VL (Lu et al., 2024), we use task-specific optimizer updates rather than mixing tasks within an update. The three tasks follow an update ratio of 2:6:1, allocating more updates to acquiring image-generation capability. Combining text-only and I2T updates yields a text-generation-to-image-generation update ratio of 1:2. We repeatedly sample from the respective data pools according to this schedule; dataset sizes describe the available pools rather than total training exposure. The unified runs consume approximately 45B tokens from the pure-text corpus, excluding captions and I2T responses. Text-only references use approximately the same pure-text token budget, with additional comparisons provided in the supplementary unified-training analyses.

Sequence packing and training objective. Each optimizer update uses an effective batch of 512 sequences with a maximum context length of 4,096. Text data are pre-chunked into 4K-token sequences. T2I and I2T examples are instead packed sequentially, with attention masks separating examples and preventing cross-example interactions. Packed multimodal sequences contain approximately four examples on average, corresponding to roughly 2,048 examples per update. We retain the backbone's positional encoding and causal attention within each example. All tasks use standard next-token cross-entropy over the full joint text–visual vocabulary, without modality-specific softmax masking or additional task-specific loss weights. Text-only updates supervise non-padding text targets; T2I updates supervise visual codes and boundary/end tokens while masking the conditioning prompt; I2T updates supervise assistant responses and their closing delimiters while masking image tokens and user context. Task balance is controlled through update frequency rather than explicit loss reweighting.

Optimization. Joint training runs for 100K optimizer updates using fused AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$. We use a cosine learning-rate schedule with 3,000 warm-up updates, a peak learning rate of 10^{-4}, and a minimum learning rate of 10^{-5}. Weight decay is 0.1 for parameters assigned to the decay group, and the gradient-clipping threshold is 1.0. Training uses bfloat16 precision. Paired 1D and 2D runs share the same data mixture, task schedule, visual interface, and downstream optimization settings.

A.3 Evaluation Protocols

Checkpoint selection and language evaluation. We evaluate the final training checkpoint of each run. For unified models, the same checkpoint is used for language, visual understanding, and image-generation benchmarks, without additional task-specific fine-tuning. We use the Language Model Evaluation Harness (Gao et al., 2021), with few-shot configurations following Qwen3 (Yang et al., 2025). Table 5 summarizes the task-specific settings. All generation-based language evaluations use deterministic decoding. MMLU scores answer choices by conditional likelihood without generating chain-of-thought (CoT). For MMLU-Redux and MMLU-Pro, exact match (EM) is computed after extracting the predicted option; MMLU-Pro is run as a separate evaluation task. GSM8K uses flexible numerical extraction rather than strict string matching, while MATH-500 uses `math_verify` to check normalized mathematical equivalence. We evaluate the

Table 5: Language evaluation protocols. All scores are reported as percentages.

Task	Prompting / output limit	Metric
MMLU	5-shot, no CoT choice likelihood	Accuracy
MMLU-Redux	5-shot, no CoT 8 tokens	EM
MMLU-Pro	5-shot CoT 2,048 tokens	EM
GSM8K	4-shot CoT 512 tokens	EM
MATH-500	4-shot CoT 2,048 tokens	`math_verify`
MBPP	3-shot one completion	pass@1

original MBPP with three-shot prompting, not MBPP+, and report pass@1 by executing one generated program against the test cases.

Visual understanding. We use lmms-eval (Zhang et al., 2025b), following the benchmark protocols of LLaVA-1.5 (Liu et al., 2024). Unified-model results are obtained directly from the final unified checkpoint. To assess standalone tokenizer understanding, we instead feed quantized visual features from a frozen tokenizer into the standard two-stage LLaVA pipeline: projector pretraining followed by supervised fine-tuning of the projector and language model. The language backbones are specified in the result tables. We additionally evaluate a multi-codebook (MCQ) variant, following UniTok (Ma et al., 2026), to assess visual understanding with increased quantization capacity.

Image reconstruction and generation. For ImageNet 256×256 reconstruction and class-conditional generation, we follow the evaluation protocol of LlamaGen (Sun et al., 2024). We distinguish reconstruction FID (rFID), computed on reconstructed images, from generation FID (gFID), computed on generated images; the accompanying reconstruction and generation metrics are reported in the corresponding tables. For text-to-image generation, we use the standard DPG evaluation protocol (Hu et al., 2024). GenEval (Ghosh et al., 2023) is evaluated with rewrite prompts.

B Additional Analyses of Visual Representations and Unified Training

B.1 Training Dynamics across Tokenizer Configurations

Training dynamics at 4B. Figure 4 complements the 8B trajectories in the main paper with six language benchmarks throughout 4B training. Both unified-training runs exhibit an initial decline, followed by benchmark-dependent recovery or stabilization. LMTok generally achieves better language retention than the spatial 2D baseline during the later stages, particularly on MMLU-Pro, MATH-500, and MBPP. These trajectories extend the main observation to the smaller backbone without relying solely on the final checkpoint.

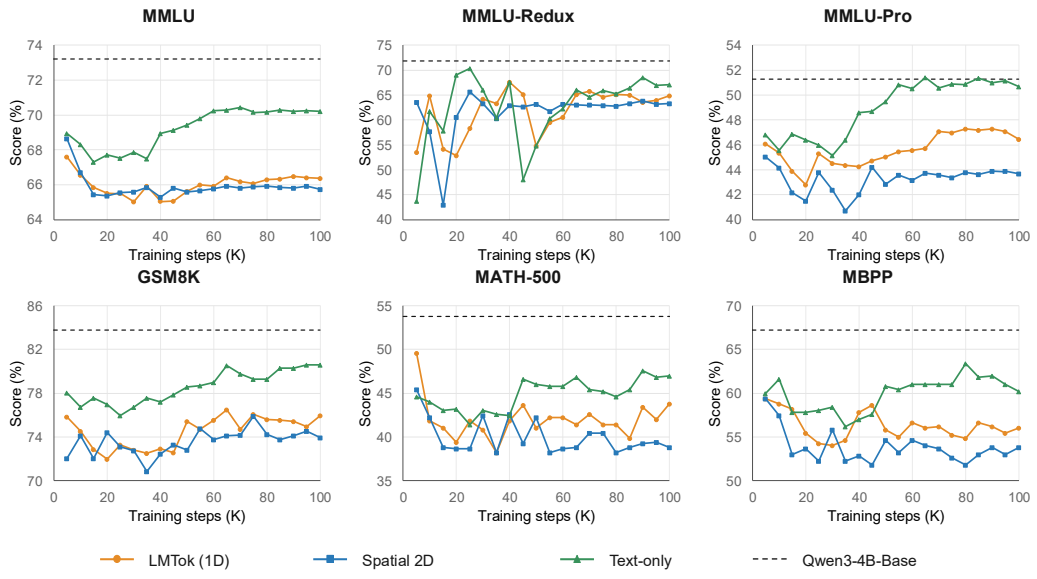

Figure 4: Language performance throughout 4B training with the single-codebook tokenizers. Orange and blue curves show unified training with LMTok and the spatial 2D baseline, respectively; green shows text-only continued training. Black dashed lines indicate the original Qwen3-4B-Base scores. The horizontal axis reports optimizer steps without rescaling.

Extension to multi-codebook tokenization. We further examine the MCQ tokenizer configuration with the same 4B backbone (Figure 5). The learned 1D representation again achieves higher late-stage scores on MMLU-Pro, GSM8K, MATH-500, and MBPP than its spatial 2D counterpart. This provides evidence that the language-retention benefit is not restricted to the single-codebook tokenizer used in the main experiments, although its magnitude and consistency remain benchmark-dependent.

Benchmark-dependent variability. MMLU-Redux exhibits pronounced checkpoint-to-checkpoint fluctuations, including in the text-only reference. Our protocol evaluates generated answer choices rather than option likelihoods, so changes in answer formatting or extraction may contribute to these fluctuations. However, the curves alone do not establish their cause, and the MCQ variant does not show a consistent advantage on this benchmark.

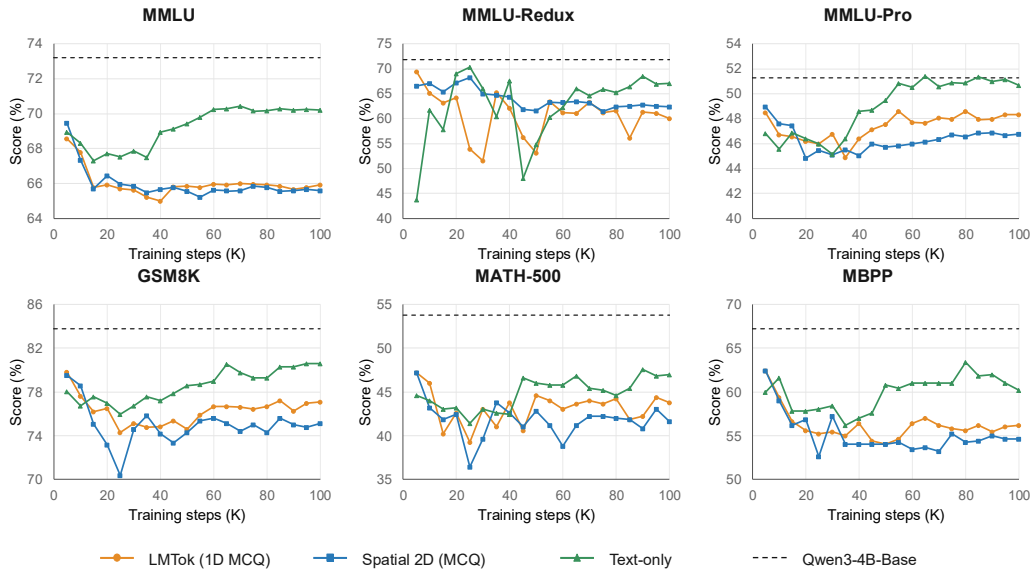

Figure 5: Language performance throughout 4B training with the MCQ tokenizer configuration. Orange and blue curves compare the learned 1D and spatial 2D multi-codebook tokenizers. The text-only curve and Qwen3-4B-Base reference are the same as in Figure 4. All recorded checkpoints are shown.

B.2 Qualitative Text-to-Image Comparison at 8B

Figure 6 compares selected text-to-image outputs from the 8B unified models using LMTok and the spatial 2D baseline. Across these examples, both produce coherent objects, detailed textures, and plausible scenes, with no clear overall visual-quality advantage for either tokenizer. This qualitative observation is consistent with their comparable text-to-image benchmark results; the selected examples do not establish statistical equivalence.

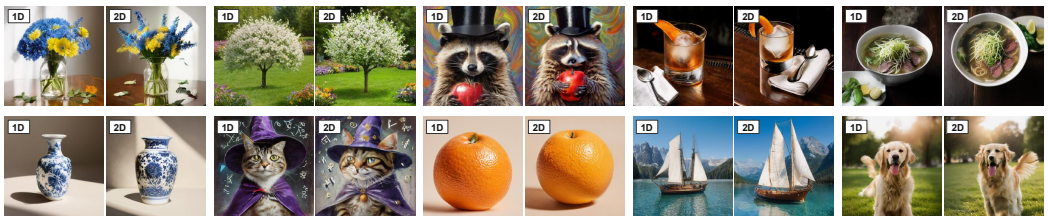

Figure 6: Selected text-to-image outputs from the 8B unified models. Each adjacent pair compares LMTok (1D, left) with the spatial 2D baseline (right). Both exhibit broadly comparable visual quality across the displayed examples.

B.3 Codebook Utilization and Tokenizer Health

We check the two main-experiment tokenizers on all 50,000 ImageNet validation images, using identical 256×256 center crops and 256 tokens per image (12.8M assignments each). Code frequencies are ranked independently for each tokenizer. The effective code count is 2^H, where H is the entropy in bits of the aggregate code-frequency distribution; one unused 1D code is omitted from the logarithmic axis.

Figure 7 shows broad code usage for both LMTok and the spatial 2D baseline. Their codebook coverage is 99.99% and 100%, respectively, and their effective code counts are 98.17% and 98.31% of the 16,384-entry vocabulary. These statistics indicate healthy codebook utilization, with no evidence of substantial collapse on this evaluation set.

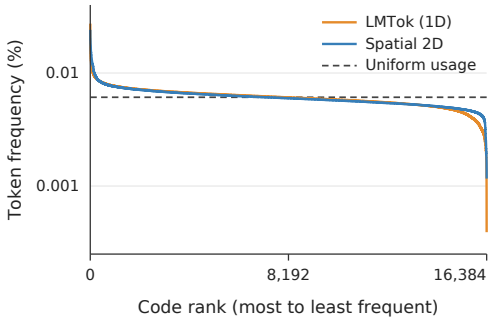

Figure 7: Ranked code frequencies for LMTok and the 2D baseline. The dashed line indicates uniform usage.

B.4 Latent Organization through Progressive Token Replacement

We probe how sequence positions relate to decoded image content by progressively replacing tokens between two images. This analysis operates directly on the tokenizer and decoder, without a downstream autoregressive generator.

Protocol. For each tokenizer separately, we encode a source image A and a target image B into two sequences of 256 discrete token IDs, $a_{1:256}$ and $b_{1:256}$. At replacement step N, we construct

$$\widetilde{\boldsymbol{i}}^{(N)} = (b_1, \ldots, b_N, a_{N+1}, \ldots, a_{256}), \qquad N \in \{0, \ldots, 256\}, \tag{6}$$

and decode the mixed sequence using that tokenizer's codebook lookup and image decoder. Thus, $N = 0$ and $N = 256$ are reconstructions of A and B, respectively. Intermediate images result from discrete token replacement, not pixel interpolation or autoregressive sampling. For the 2D tokenizer, positions follow raster order on a 16×16 grid, from left to right and then top to bottom. For LMTok, replacement follows the learned 1D latent order, without assigning positions to a fixed spatial grid.



Figure 8: Progressive prefix token replacement for two image pairs. Columns show the same approximately uniformly spaced replacement counts for both tokenizers, including the source and target reconstructions. The 2D transitions follow spatial raster order, whereas the 1D transitions exhibit changes across image regions rather than a spatial sweep.

Figure 8 shows two selected pairs, school bus to panda and zebra to hot-air balloon. We display 12 approximately uniformly spaced steps, $N \in \{0, 23, 47, 70, 93, 116, 140, 163, 186, 209, 233, 256\}$, using identical replacement counts for the 1D and 2D rows.

Observations. For 2D tokens, target content appears progressively from the upper image regions downward, while lower regions retain source content, consistent with raster-order spatial correspondence. In contrast, the 1D transitions reorganize content across multiple regions: object appearance, shape, and background change without a comparable top-to-bottom replacement boundary. These examples indicate that the learned 1D sequence is not simply a flattened spatial grid; it exhibits a latent organization distinct from 2D raster order. This qualitative probe illustrates the structural freedom of 1D representations, rather than establishing a unique semantic meaning for individual tokens or isolating the causal effect of the AR prior.

C Ablations on the Autoregressive Prior

We examine how the AR prior affects visual tokenization and which choices matter when incorporating this supervision. Unless otherwise specified, these ablations use a 1D SimVQ tokenizer with a 16,384-entry codebook, a code dimension of 256, semantic-alignment weight 1, and a GPT-B prior with approximately 100M parameters. The GPT-B prior is trained jointly with the tokenizer from random initialization, unlike the frozen pretrained Qwen3 prior in the main experiments. The default ablation configuration achieves a class-conditional generation FID (gFID) of 1.89 and should be distinguished from the main-experiment configuration. Reconstruction FID is denoted by rFID. C2I results use classifier-free guidance with a power-controlled cosine schedule over autoregressive decoding positions (CFG-power): the guidance scale increases from 1 toward its configured value, while the power exponent controls the schedule's shape. This is an inference-time sampling choice, separate from the AR-prior training objective. Comparisons are made within each experimental group; reported decimal precision varies across tables, and small differences are not treated as statistically established improvements.

C.1 Effect of the AR Prior on 1D and 2D Tokenization

Learned 1D queries remove the fixed correspondence between latent positions and image patches, but this flexibility does not itself specify an organization suitable for next-token prediction. We compare tokenizers trained with and without an AR prior for both 1D and spatial 2D representations. The no-prior 1D tokenizer still learns its latent queries through the remaining objectives; it is not a random or untrained representation.

AR supervision improves the generation FID of learned 1D tokens, while spatial 2D tokens already support strong autoregressive generation without it. In Table 6, adding the prior reduces 1D gFID

Table 6: Effect of the AR prior on class-conditional image generation. The prior is used during tokenizer training.

Representation	Tokenizer prior	gFID ↓	IS ↑	Precision ↑
1D	None	2.22	262.66	0.80
1D	GPT-B	1.89	307.70	0.81
2D	None	1.79	264.01	0.81
2D	GPT-B	1.86	293.36	0.81

from 2.22 to 1.89; for 2D tokens, gFID changes from 1.79 to 1.86, although IS improves for both representations. These results support guiding flexible 1D latent organization with an AR prior. This comparison concerns standalone generation; its effect on language retention is evaluated separately through the unified-training ablations in Section 4.4.

C.2 Prior Model Configurations

We compare GPT-B and GPT-L priors trained from scratch with a frozen pretrained Qwen3-0.6B prior. GPT-B and GPT-L have approximately 100M and 300M parameters, respectively. These are tokenizer-training priors, not the downstream C2I generators.

Table 7: AR-prior configurations. The Qwen3 comparison changes pretraining and freezing as well as model size, and is therefore not a pure size ablation.

Prior	Initialization / training	gFID ↓	IS ↑	Precision ↑
GPT-B (100M)	From scratch	1.89	307.70	0.81
GPT-L (300M)	From scratch	1.75	304.02	0.81
Qwen3-0.6B	Pretrained, frozen	1.81	302.76	0.79

Increasing the scratch-trained prior from GPT-B to GPT-L improves gFID from 1.89 to 1.75 in this comparison, while IS changes from 307.70 to 304.02. The frozen Qwen3 prior obtains gFID 1.81, between the two scratch-trained priors. A pretrained language prior can therefore guide tokenization without preventing strong standalone generation, but it is not the best configuration on every visual metric. Its value for our main question must also be assessed through language retention during unified training, rather than selected solely by C2I quality.

C.3 Input and Output Parameterizations

We compare AR-prior interfaces for IBQ and SimVQ to determine a suitable default configuration. The input interface controls how AR supervision reaches the tokenizer, while the output head defines how next-token predictions are scored.

Input interfaces. IBQ uses an indexing-style interface with a differentiable assignment: $A = A_{\text{hard}} - \text{sg}(P) + P$, where P contains soft code assignments and A_{hard} contains hard one-hot selections. The prior receives $U = AV$, with a separate trainable visual embedding table V. Unlike integer indexing, this assignment-side STE allows AR gradients to reach both the encoder and the codebook. SimVQ instead projects quantized code vectors into the prior's hidden space using a trainable linear projector. Its vector STE, $q_n = z_n + \text{sg}(c_{i_n} - z_n)$, passes the input-side AR gradient to the normalized encoder latents z_n, but not directly to the codebook. This projected-vector interface does not require IBQ-specific differentiable assignments. A frozen prior still permits backpropagation through its inputs.

Output heads. The linear head independently maps prior hidden states to vocabulary logits. The codebook-based logits head instead projects hidden states to the code dimension, normalizes them, and computes their dot products with the current normalized, stop-gradient codebook. Its classification references therefore follow the evolving code representations without receiving direct output-side AR gradients. Both heads use next-token cross-entropy with hard token IDs as non-differentiable targets.

Table 8: AR-prior interface ablations. Usage is reported as a percentage. Quantizer and input interface change together between the IBQ and SimVQ groups; within each group, the output parameterization is varied.

Quantizer	Input	Output	rFID ↓	PSNR ↑	SSIM ↑	Usage	gFID ↓	IS ↑	Prec. ↑
IBQ	Indexing	Linear head	1.048	20.25	0.649	99.85	1.79	293.70	0.79
IBQ	Indexing	Codebook scores	0.936	20.82	0.676	99.96	2.00	289.07	0.80
SimVQ	Projection	Linear head	1.094	20.19	0.649	100.00	2.70	228.34	0.76
SimVQ	Projection	Codebook scores	1.015	20.62	0.671	100.00	1.89	307.70	0.81

Default configuration. For general applicability and training robustness, we adopt SimVQ with the projected-vector input interface and select its output head using Table 8. Within SimVQ, codebook-based scoring improves both reconstruction and generation over the independent linear head: rFID decreases from 1.094 to 1.015, gFID from 2.70 to 1.89, and IS increases from 228.34 to 307.70, with 100% codebook usage. We therefore use *SimVQ + Projector + codebook-based logits head* as our default. The preferred head is interface-dependent: for IBQ, codebook-based scoring improves reconstruction but worsens gFID (1.79 to 2.00). Thus, this ablation supports our interface choice within SimVQ, rather than a universal ranking of quantizers.

Codebook optimization. With SimVQ's input-side vector STE and detached output-side codebook, AR supervision shapes the encoder latents without directly updating the codebook parameters. Separate quantization objectives train the effective codebook. Writing Q_{hard} for selected code vectors, the fitting and commitment terms are $\mathrm{MSE}(Q_{\mathrm{hard}}, \mathrm{sg}(Z))$ and $0.2\,\mathrm{MSE}(\mathrm{sg}(Q_{\mathrm{hard}}), Z)$, respectively. We also use $0.01[\mathbb{E}_n H(\rho_n) - H(\mathbb{E}_n \rho_n)]$, where ρ_n is the soft code distribution and the expectation includes the batch. In contrast, IBQ retains an input-side AR gradient to the codebook through its assignments, even when the output scoring codebook is detached.

C.4 Auxiliary Reconstruction from Prior Predictions

EOSTok (Chu et al., 2026) introduces autoregressive prediction reconstruction (APR), which decodes teacher-forced AR predictions and supervises them in pixel space to mitigate codebook collapse induced by next-token prediction. We evaluate this auxiliary branch in our setting, comparing the default without APR against APR configurations with and without GAN supervision (Table 9).

Table 9: Auxiliary reconstruction from prior predictions. The row names distinguish the APR/GAN configurations used in this experiment.

Configuration	gFID ↓	IS ↑	Precision ↑
Without APR	1.89	307.70	0.81
APR with GAN	1.82	307.49	0.81
APR without GAN	1.70	319.81	0.80

APR provides modest additional gains: gFID decreases from 1.89 to 1.82 with GAN supervision and to 1.70 without it. However, this branch adds decoding and reconstruction-loss computation during training. Our default tokenizer already maintains broad codebook usage without evidence of substantial collapse (Appendix B.3). Given the limited gains and additional training cost, we do not use APR in our default recipe.

D Additional Visual Tokenizer Ablations

We examine the remaining representation-design choices in four experimental settings. The semantic-weight sweep uses the default discrete SimVQ tokenizer and GPT-B prior described in Appendix C, varying the overall semantic-loss weight. The direct- and cross-alignment comparisons use continuous 1152-dimensional representations without quantization. The SimVQ codebook-size sweep uses 256-dimensional codes without an AR prior; SQ, MCQ, and continuous representations are reported separately as capacity references. The variable-length study uses a separately trained tail-drop tokenizer with a GPT-B prior. These groups do not evaluate one shared checkpoint: comparisons and default values are specific to the stated configuration.

D.1 Semantic Supervision Strength

We vary the semantic-alignment weight from 0 to 1 and 5, retaining the AR prior. This tests both the need for semantic supervision and the trade-off induced by increasing its strength.

Table 10: Semantic-alignment weight. Understanding scores are reported on a 0–100 scale except for MME-P.

Weight	POPE ↑	MME-P ↑	GQA ↑	SEED ↑	PSNR ↑	rFID ↓	gFID ↓
0	67.60	738.2	41.53	35.70	21.02	1.16	2.09
1	81.37	1197.6	55.13	51.30	20.61	1.01	1.89
5	83.41	1317.9	59.70	60.42	18.05	1.20	1.90

Adding semantic supervision substantially improves understanding: from weight 0 to 1, GQA increases by 12.51 points and SEED by 15.00 points, while gFID improves from 2.09 to 1.89. Increasing the weight to 5 further raises GQA to 59.70 and SEED to 60.42, but reduces PSNR from 20.61 to 18.05 and increases rFID from 1.01 to 1.20. C2I gFID remains close (1.89 versus 1.90). The stronger alignment therefore trades reconstruction fidelity for semantic performance in this range, rather than causing a clearly established degradation in generation. Weight 1 provides a balanced default for these ablations.

D.2 Direct Alignment vs. Cross-Alignment

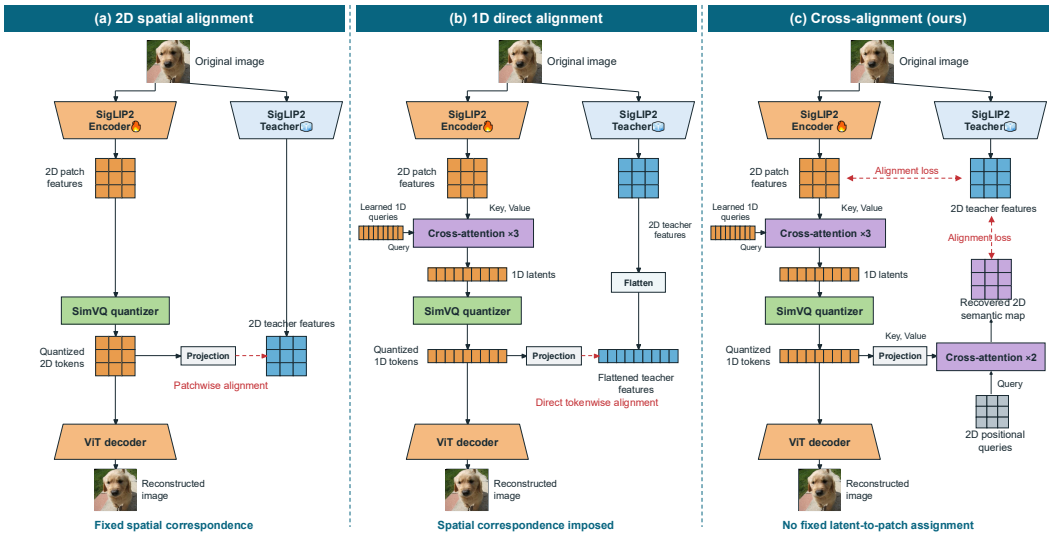

Figure 9: Comparison of semantic alignment strategies. Spatial 2D alignment and direct alignment to flattened teacher features impose fixed patch correspondence. Cross-alignment instead uses 2D positional queries to recover spatial features from the full 1D sequence, providing semantic supervision without assigning individual latents to fixed patches.

Figure 9 contrasts the three alignment pathways.

Direct alignment matches each 1D latent to the corresponding position in the flattened SigLIP2 feature map. Although stored as a sequence, the representation is thereby supervised to follow a fixed spatial correspondence. Cross-alignment instead uses 2D positional queries to recover teacher-aligned spatial features from the full 1D sequence, without requiring each latent to represent a designated patch.

To isolate this structural choice from quantization, we evaluate continuous 1152-dimensional representations. Table 11 includes the direct-alignment baseline and the cross-alignment variants. Relative to direct alignment, the selected two-layer design with a learnable query offset obtains similar GQA (62.04 versus 61.78), MME (1485 versus 1469), and SEED (64.80 versus 64.89), while

DocVQA decreases from 20.46 to 19.03 and rFID changes from 0.125 to 0.134. The result is not a uniform improvement over direct alignment. Rather, it shows that removing the explicit latent-to-patch matching constraint can retain comparable performance on several understanding benchmarks, leaving the latent organization available for AR supervision.

D.3 Cross-Alignment Architecture

Each cross-alignment layer applies cross-attention from 2D positional queries to the 1D latent keys and values. We vary the number of layers and optionally add query self-attention before cross-attention or an FFN afterward. All sublayers use pre-layer normalization and residual connections, followed by an output layer normalization. The query-offset variant adds a zero-initialized, image-independent learnable offset to the frozen positional queries; it does not denote the presence or absence of ordinary Transformer residual connections.

Table 11: Alignment design in the continuous 1152-dimensional setting, without quantization. "Offset" denotes a learnable residual on the initial positional queries. DocVQA, GQA, and SEED are expressed on a 0–100 scale.

Alignment design	rFID ↓	PSNR ↑	SSIM ↑	DocVQA ↑	GQA ↑	MME ↑	SEED ↑
Direct alignment	0.125	32.10	0.949	20.46	61.78	1469	64.89
1 cross-attention layer	0.159	31.03	0.937	11.69	61.04	1382	62.18
2 layers	0.151	31.57	0.944	18.82	61.96	1479	65.28
2 layers + self-attention	0.096	34.29	0.969	15.99	59.87	1423	60.91
2 layers + offset (default)	0.134	31.72	0.946	19.03	62.04	1485	64.80
2 layers + FFN	0.143	31.90	0.947	17.16	60.65	1456	62.70
3 layers	0.116	33.13	0.960	15.89	61.42	1412	62.87

The cross-alignment variants use cosine supervision with overall weight 1 and a patch-to-cross loss ratio $r = 0.5$. Specifically, $\mathcal{L}_{\text{align}} = (r\mathcal{L}_{\text{patch}} + \mathcal{L}_{\text{cross}})/(r+1)$, where the patch term aligns the encoder's spatial features and the cross term aligns spatial features recovered from the 1D latents. Thus, the two terms receive weights $1/3$ and $2/3$, respectively. In this continuous ablation, the cross branch reads unquantized latents; in the full tokenizer it reads quantized representations.

Two cross-attention layers improve the understanding results over one layer. Adding query self-attention yields the best reconstruction in this table, but lowers GQA and SEED relative to the two-layer design. Neither adding an FFN nor increasing depth to three layers consistently improves understanding. The two-layer offset design offers a compact compromise, with the highest GQA and MME among the tested cross-alignment variants, although the plain two-layer design has slightly higher SEED. These results distinguish semantic preservation from reconstruction capacity rather than treating either as a proxy for the other.

D.4 Codebook Capacity

We vary the SimVQ codebook size without an AR prior, keeping the code dimension at 256. Table 12 reports reconstruction, code usage, and understanding. Increasing the vocabulary from 16k to 262k reduces rFID from 1.013 to 0.916 and raises PSNR from 20.38 to 20.73. GQA increases from 51.54 to 52.92 and SEED from 46.81 to 49.58; DocVQA and MME do not improve consistently. Larger capacity therefore benefits reconstruction and some semantic metrics, but not every understanding benchmark.

Table 12: SimVQ codebook-size sweep without an AR prior, at code dimension 256. Usage is a percentage; understanding scores other than MME are expressed on a 0–100 scale.

Codebook	rFID ↓	PSNR ↑	SSIM ↑	Usage	DocVQA ↑	GQA ↑	MME ↑	SEED ↑
16k	1.013	20.38	0.664	100.00	8.13	51.54	1127	46.81
32k	0.980	20.57	0.671	100.00	8.24	51.69	1103	47.79
131k	0.956	20.68	0.676	100.00	8.35	51.86	1128	48.99
262k	0.916	20.73	0.678	99.58	8.00	52.92	1124	49.58

We retain a 16k vocabulary for the main comparison rather than maximizing codebook capacity. This keeps the discrete prediction space compact; the present sweep does not measure how increasing the vocabulary changes C2I quality or unified language retention.

Additional capacity references. For context, Table 13 includes scalar-quantized (SQ), multi-codebook (MCQ), frozen SigLIP2, and continuous learned representations. SQ uses dimension 24 and MCQ uses dimension 72, so these rows are not part of the controlled 256-dimensional SimVQ sweep. MCQ has eight 2048-entry codebooks; this is not equivalent to a single 16,384-way symbol. These references show the visual information that alternative capacity allocations can preserve, without establishing a ranking of quantization methods under matched rate or computation.

Table 13: Additional representation-capacity references, separate from the controlled codebook sweep. Continuous representations have no discrete vocabulary or codebook utilization.

Representation	Codebook	rFID ↓	PSNR ↑	SSIM ↑	Usage (%)	DocVQA ↑	GQA ↑	MME ↑	SEED ↑
SQ ($d = 24$)	195k	0.719	20.92	0.680	99.98	8.76	57.82	1350	56.95
MCQ ($d = 72$)	8×2048	0.344	25.26	0.836	100.00	13.07	60.92	1436	61.30
Frozen SigLIP2	—	0.549	20.36	0.668	—	20.50	61.65	1479	64.80
Learned continuous	—	0.215	29.17	0.909	—	20.91	62.16	1440	65.47

D.5 Variable-Length Representations

Tail-drop training. We train a separate tokenizer with tail-drop, which removes a suffix of the latent sequence during training and retains its prefix. This tokenizer uses a three-layer cross-attention sequence extractor, a GPT-B prior, and a maximum length of 256 tokens. At evaluation, the same tokenizer checkpoint is used at every reconstruction budget by keeping its first L tokens. The reconstruction decoder reads the retained latent tokens through attention while maintaining the image reconstruction queries. No tokenizer retraining is required to change the reconstruction budget.

Table 14: Variable-length reconstruction using prefixes of a single tail-drop tokenizer checkpoint. This checkpoint is separate from the fixed-length tokenizer used in the preceding default ablations.

Retained tokens	PSNR ↑	rFID ↓
32	16.335	5.665
64	17.506	2.837
128	18.677	1.683
256	19.264	1.472

Reconstruction improves as more tokens are retained: PSNR increases from 16.335 at 32 tokens to 19.264 at 256 tokens, while rFID decreases from 5.665 to 1.472. This demonstrates a controllable quality–length trade-off within one tokenizer. It does not imply that the model automatically chooses an image-dependent stopping point.

Generation at specified token budgets. We also train GPT-L C2I generators separately on 128-token and 256-token prefixes produced by this tail-drop tokenizer. The token budget is specified manually for each generator; this is not a test of one generator switching its output length without retraining.

Table 15: C2I generation with the separately trained tail-drop tokenizer. A GPT-L generator is trained for each specified token length; results use CFG-power sampling.

Tokens	gFID ↓	IS ↑	sFID ↓	Precision ↑	Recall ↑
256	2.18	290.10	6.71	0.80	0.59
128	2.22	274.18	6.54	0.80	0.60

Halving the visual sequence length changes gFID from 2.18 to 2.22 and IS from 290.10 to 274.18, while precision remains 0.80; sFID and recall change in the opposite direction. The results support

generating from shorter prefixes with a small observed gFID difference, without a uniform ranking across all metrics. The 256-token result is not the fixed-length baseline's gFID of 1.89: tail-drop changes tokenizer training, so the two checkpoints must be distinguished even when their inference lengths coincide.