Rethinking Causal Action Tokenization with Conditional Annealing in Flow Matching

Chenyu Zhang[1,2*] **Yuhang Cao**[1*] **Daru Du**[1,2] **Yingxi Lu**[1,2] **Jing Shao**[1]
Ruoqu Chen[1,2] **Jiajun Liu**[1,2] **Liu Cao**[1,2] **Yicheng Liu**[1] **Hang Zhao**[1,2] **Mengdi Xu**[1,2†]

[1]IIIS, Tsinghua University [2] Shanghai Qizhi Institute
{cyzhang21@mails, caoyh24@mails, xumd@mail}.tsinghua.edu.cn
https://chenyuzhangx.github.io/CATok/

Abstract

Autoregressive Vision-Language-Action (VLA) models offer a scalable path to robot learning, yet existing action tokenizers treat tokenization as a compression problem, producing representations that are semantically misaligned with the autoregressive backbone. We propose **CATok**, a causal action tokenizer that reframes tokenization as a causally structured generative process. CATok introduces a *conditional annealing mechanism* that extracts action tokens by progressively annealing a flow-matching process: each token is conditioned on all preceding tokens and encodes the residual reconstruction signal at a specific noise level, establishing a coarse-to-fine causal token space whose generative semantics are structurally aligned with autoregressive modeling. A *token-conditioned flow-matching decoder* built on Multimodal Diffusion Transformer (MMDiT) reconstructs continuous action chunks from these discrete tokens with the precision of hybrid diffusion-head architectures. This discrete bottleneck enforces *knowledge insulation* by design, cleanly separating high-level semantic reasoning from low-level motor execution without requiring explicit attention masking. Extensive evaluations across three simulation benchmarks and real-world robotic manipulation tasks demonstrate that CATok consistently surpasses existing tokenization methods in both reconstruction fidelity–compression tradeoff and inference efficiency, while improving VLA task success rate and training efficiency, establishing a high-performance, scalable foundation for purely autoregressive VLA systems.

1 Introduction

Robot foundation models have shown strong generalization across diverse manipulation tasks [1–8]. Vision-Language-Action (VLA) architectures have emerged as a particularly promising paradigm, jointly modeling language instructions, visual observations, and motor commands within a unified policy [1, 2, 5, 6]. Existing VLAs broadly fall into two families: *purely autoregressive models* that predict discretized action tokens using the same next-token objective as language models [1, 2, 9], and *hybrid models* that pair an autoregressive language backbone with a continuous diffusion or flow-matching action head [5, 6]. Hybrid models achieve precise continuous control and avoid the bottleneck of naive action discretization, but they decouple action generation from the autoregressive backbone, limiting the unified token-level modeling and scalable LLM training infrastructure that make purely autoregressive VLAs appealing [10–12].

*Equal contribution. †Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

A central challenge for purely autoregressive VLAs is therefore how to tokenize continuous robot actions. The simplest strategy is per-dimension uniform binning [1, 2, 9], where each scalar action dimension is independently mapped to a discrete bin. This produces long token sequences of length $O(D \times H)$, ignores correlations across action dimensions, and provides no meaningful causal structure within an action chunk. Recent action tokenizers instead compress entire action chunks into short discrete sequences [13–15], but they remain imperfectly matched to autoregressive generation. FAST [13] uses DCT compression and BPE to obtain compact codes, yet its frequency-ordered coefficients are not naturally aligned with left-to-right autoregressive generation, and its variable-length outputs can make decoding brittle [15]. FASTer [14] learns fixed-length RVQ codes with strong reconstruction fidelity, but optimizing a codebook for reconstruction alone does not ensure compatibility with the language-model backbone. OAT [15] introduces a left-to-right ordering through nested dropout, but the resulting token positions lack semantic grounding, since each position has no principled correspondence to information granularity or generative stage.

These limitations point to a deeper issue: *existing action tokenizers mainly encode **what** action should be reconstructed, but not **how** that action should be generated.* In generative modeling, denoising processes naturally expose a hierarchy of abstraction: high-noise steps capture coarse global structure, while low-noise steps refine fine-grained details [16]. If action tokens were aligned with this hierarchy, then a token sequence could provide a causally ordered, coarse-to-fine representation of continuous control. Such a representation would be better matched to autoregressive generation: early tokens would specify global action structure, and later tokens would refine local motor details. However, existing action tokenizers do not explicitly exploit this correspondence between denoising hierarchy and left-to-right token generation.

Inspired by these observations, we introduce CATOK, a causal action tokenizer that aligns action tokenization with the generative structure of autoregressive models. CATOK represents action chunks as compact discrete tokens and reconstructs them using an MMDiT-based flow-matching decoder [17, 18], achieving the precision of diffusion-based action heads [5] while remaining fully compatible with autoregressive VLA training. To structure the token space, we employ a conditional annealing process in which each token is generated conditioned on its predecessors and corresponds to a stage of the denoising process, resulting in a causally ordered coarse-to-fine representation of actions [16] (illustrated in Figure 1). Experiments on three simulation benchmarks show that CATOK achieves the highest success rates across all benchmarks, while delivering a $1.7\times$ faster VLA inference speed and a $3.6\times$ stronger reconstruction fidelity–compression balance than FAST.

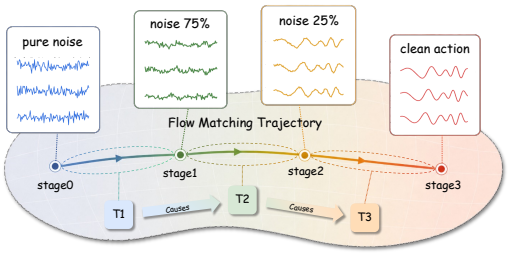


Figure 1: **Flow-Matching to Action Tokens.** CATOK grounds discrete action tokens in the continuous flow matching trajectory. Each token is learned as a stage-wise information increment, imbuing the sequence with a natural coarse-to-fine hierarchy. By construction, these increments follow the temporal evolution of the flow, ensuring a causally-ordered structure that seamlessly aligns with autoregressive generation.

Our contributions are three-fold:

- **Causal action tokenization via conditional annealing.** We formulate action tokenization as a sequential generative process via conditional annealing, producing tokens that follow a coarse-to-fine, causally ordered structure aligned with autoregressive modeling.
- **Token-conditioned flow matching decoder.** We introduce an MMDiT-based [17] flow-matching decoder [18] that reconstructs continuous action chunks from compact discrete tokens, combining the control fidelity of continuous generative action heads with the training compatibility of purely autoregressive VLAs.
- **Strong empirical performance.** CATOK consistently outperforms prior methods such as FAST [13] and OAT [15] across multiple simulation and real-world benchmarks in reconstruction fidelity–compression tradeoff, inference efficiency and VLA task success rate.

2 Related Works

Vision-Language-Action Models. Benefiting from the strong capability of pretrained VLMs [19–21] to understand images and language instructions, VLAs demonstrate remarkable performance and generalization in robotic manipulation tasks [3, 22]. Existing VLA frameworks primarily adopt two paradigms for action integration: discrete tokenization for autoregressive generation [2, 9, 13, 11, 23], and continuous regression via an auxiliary action head conditioned on the VLM's latent representations [5, 24, 6, 25]. Despite the fact that discrete-token autoregressive paradigms incur excessive inference delays, recent works reveal a contrasting boost in training performance. FAST [13] demonstrates that employing a tokenizer that effectively compresses action chunks can significantly enhance training efficiency. Moreover, fine-tuning VLMs via discrete tokens, rather than continuous action heads, has been shown to better preserve the rich semantics of the model [26]. Together, these findings underscore the critical importance of designing an optimal discrete action tokenizer.

Action Tokenization. Inherently, an action chunk is represented as a two-dimensional matrix spanning both temporal and spatial (action) dimensions. To process such structures, researchers have proposed a variety of action tokenization strategies, ranging from rule-based methods to data-driven approaches. Early binning-based action discretization [2, 9, 22] introduces substantial token redundancy, severely compromising both training and inference speeds. More critically, this naive mapping ignores inter-token dependencies, limiting its effectiveness under the standard next-token-prediction framework. Compression-based methods (e.g., DCT [13] and B-splines [27]) were subsequently introduced to compress each action dimension independently along the temporal axis. While these approaches substantially improve the compression ratio—particularly at high control frequencies—and boost training efficiency, they still fail to capture correlations across action dimensions. Most recently, learning-based tokenizers [11, 28, 29] have emerged to model the spatiotemporal dependencies of actions, with a growing focus on the hierarchical organization and underlying structure of the token space. For instance, RVQ-based approaches [14, 30] capture multi-level hierarchies through residual quantization, whereas OAT [15] enforces structural integrity in the token space by applying random token masking in the action decoder.

While promising, these tokenization paradigms leave room for further refinement in balancing high-quality reconstruction with token-level causal structures. A key objective, therefore, is to develop a training formulation that naturally yields a structured latent topology. To this end, we propose a method that transfers the temporal causal structure inherent in Flow Matching [16, 31, 32] into the token space, thereby inducing causality among tokens. This design is empirically validated to be effective for autoregressive modeling.

3 Method

In this section, we first formalize the action tokenization problem in Section 3.1; then, we describe the CATOK architecture in Section 3.2 and detail its training and inference procedures in Section 3.3. Finally, Section 3.4 presents VLA-CATOK, which integrates the proposed tokenizer into an autoregressive VLA backbone.

3.1 Problem Formulation

We study action tokenization for purely autoregressive VLA models. Given a continuous action chunk $\mathcal{A} = (a_t, \ldots, a_{t+H-1}) \in \mathbb{R}^{H \times d_a}$, the goal is to learn an encoder $\mathcal{E}$ and decoder $\mathcal{D}$ such that

$$\mathcal{E}(\mathcal{A}) = C = (c_1, \ldots, c_K), \quad c_k \in \{1, \ldots, |\mathcal{C}|\}, \qquad \mathcal{D}(C) \approx \mathcal{A},$$

where $\mathcal{C}$ is a finite codebook and K is the token sequence length. The tokenizer is trained to minimize reconstruction error:

$$\min_{\mathcal{E},\mathcal{D}} \mathbb{E}_{\mathcal{A}} \big[\|\mathcal{D}(\mathcal{E}(\mathcal{A})) - \mathcal{A}\|^2 \big].$$

Once trained and frozen, $\mathcal{E}$ and $\mathcal{D}$ convert action chunks into discrete tokens C, which a purely autoregressive VLA policy π_θ predicts conditioned on observations $o_t = (I_t, s_t, l)$. The VLA is

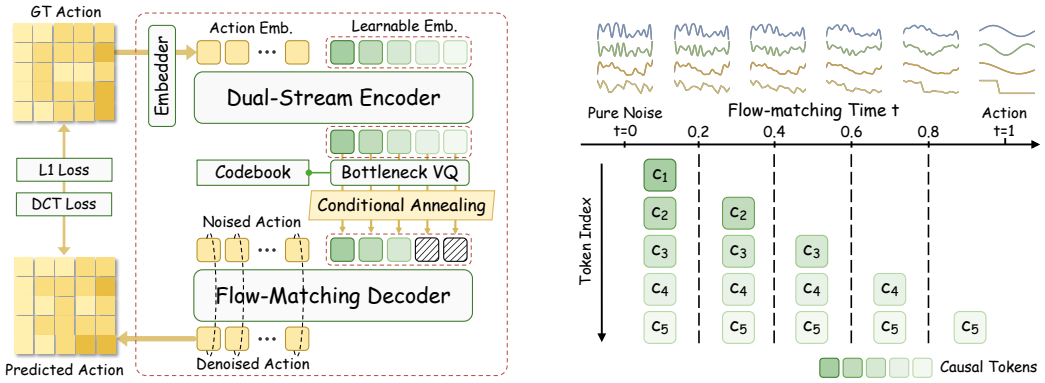


Figure 2: **An Overview of CATOK.** (a) **CATOK Pipeline.** CATOK encodes an action chunk with a *Dual-Stream Encoder* into latent queries, discretizes them via a *Bottleneck VQ*, and reconstructs actions using a *Flow-Matching Decoder with Conditional Annealing*. (b) **Conditional Annealing Mechanism** progressively masks the first $\kappa(t)$ token embeddings according to flow matching timestep t: darker colors indicate tokens earlier in the sequence, and lighter colors indicate later tokens, assigning each token to a different stage of the flow trajectory.

trained with cross-entropy over tokens:

$$\min_{\theta} \mathbb{E}_{(o_t, \mathcal{A})} \Big[-\sum_{k=1}^{K} \log \pi_\theta(c_k \mid o_t, c_{<k}) \Big].$$

At inference, π_θ generates $\hat{C} = (\hat{c}_1, \ldots, \hat{c}_K)$ autoregressively, and the frozen decoder recovers the continuous action $\hat{\mathcal{A}} = \mathcal{D}(\hat{C})$ for execution.

3.2 CATOK Architecture

Our causal action tokenizer, CATOK, encodes action chunks using a **Dual-Stream Action Encoder** that maps them into a compact set of latent query tokens via co-attention. These latent tokens are then discretized through a **Bottleneck Vector Quantizer** into a finite codebook. Finally, a **Flow-Matching Decoder with Conditional Annealing** reconstructs high-fidelity action trajectories while enforcing a coarse-to-fine causal structure over the tokens.

Dual-Stream Action Encoder. We first normalize the raw action chunk using quantile-based normalization to mitigate the impact of outliers and improve numerical stability. Instead of flattening or 1D patchification, we encode $\mathcal{A}_{t:t+H}$ using a 2D convolutional neural network (CNN) with weight normalization [33], which captures local correlations along both temporal and action dimensions. We further add learnable 2D positional embeddings to preserve the spatiotemporal structure, yielding structured embeddings $\mathbf{E}_{\text{act}}$.

In parallel, we initialize K learnable query embeddings $\mathbf{Q}^{(0)} \in \mathbb{R}^{K \times D}$ with independent positional embeddings. We then feed both $\mathbf{E}_{\text{act}}$ and $\mathbf{Q}^{(0)}$ into a symmetric multi-modal transformer inspired by MMDiT [17], where co-attention enables bidirectional interaction between the two token sets (Figure 2(a)). In this process, the action embeddings provide structured context, while the query tokens aggregate information into a compact latent space. After L layers, we retain the updated query stream $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_K) \in \mathbb{R}^{K \times D}$ as the final representation of the action chunk.

Bottleneck Vector Quantizer. To obtain a compact discrete representation, we adopt the *factorized codes* design of ViT-VQGAN [34]. Each latent vector $\mathbf{q}_k \in \mathbb{R}^D$ is first projected into a lower-dimensional space through a linear bottleneck $\psi : \mathbb{R}^D \to \mathbb{R}^{D'}$ with $D' < D$. The projected embedding $\psi(\mathbf{q}_k)$ is then quantized against a learnable codebook $\mathcal{C} = \{\mathbf{e}_1, \ldots, \mathbf{e}_{|\mathcal{C}|}\} \subset \mathbb{R}^{D'}$ by selecting its nearest neighbor indexed c_k in Euclidean distance:

$$c_k = \underset{j \in \{1, \ldots, |\mathcal{C}|\}}{\arg\min} \left\| \mathbf{e}_j - \psi(\mathbf{q}_k) \right\|_2, \tag{1}$$

Finally, a post-projection layer $\phi : \mathbb{R}^{D'} \to \mathbb{R}^{D}$ is applied to reconstruct the quantized embedding in the original latent space, $\hat{\mathbf{q}}_k = \phi(\mathbf{e}_{c_k})$. This factorized design substantially improves codebook utilization by decoupling the codebook dimension from the model's hidden dimension.

Flow-Matching Decoder with Conditional Annealing. Prior work suggests that diffusion/flow timesteps naturally correspond to different levels of abstraction, from coarse global structure to fine-grained details [16]. We exploit this property to ground discrete tokens in specific generative stages via a *conditional annealing* mechanism (as illustrated in Figure 2 (b)).

Let $x_0 \sim \mathcal{N}(0, I)$ and $x_1 = \mathcal{A}$ denote Gaussian noise and the target action chunk respectively. Following rectified flow, we define the interpolation path $x_t = (1-t)x_0 + tx_1$ with target velocity $x_1 - x_0$. Instead of conditioning the decoder on all token embeddings throughout the entire trajectory, we progressively mask the first $\kappa(t)$ tokens according to

$$\kappa(t) = \lfloor tK \rfloor, \quad t \in [0, 1], \tag{2}$$

so that only the active subset $\hat{\mathcal{Q}}_{>\kappa(t)}$ is provided as conditioning at time t. The MMDiT-based velocity network [17] is trained with the flow-matching objective

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,x_0,x_1}\left[\left\| v_\theta(x_t, t, \hat{\mathcal{Q}}_{>\kappa(t)}) - (x_1 - x_0) \right\|_1\right]. \tag{3}$$

This schedule forces different tokens to be useful at different stages of the generation process. At early stages, when x_t is dominated by noise, the active tokens must provide coarse global guidance; as t increases and the trajectory approaches the target action, fewer tokens remain active, encouraging later tokens to specialize in residual, fine-grained corrections. Consequently, each token embedding acts as a discrete information increment between adjacent flow-matching stages, inducing a causally ordered coarse-to-fine hierarchy that is naturally aligned with autoregressive action generation.

3.3 CATOK Training and Inference

Training Scheme. The model is trained end-to-end with a combination of reconstruction and quantization losses:

$$\mathcal{L} = \mathcal{L}_{\text{FM}} + \alpha\mathcal{L}_{\text{smooth}} + \beta\mathcal{L}_{\text{VQ}}, \tag{4}$$

where α, β are hyperparameters that balance the three terms.

Specifically, the reconstruction loss integrates constraints in both the time and frequency domains. It consists of a time-domain L_{FM} loss which together with a frequency-domain L_{smooth} loss computed via the discrete cosine transform (DCT):

$$\mathcal{L}_{\text{smooth}} = \mathbb{E}_{t,x_0,x_1}\left\|\text{DCT}(x_1 - x_0) - \text{DCT}(v_\theta(x_t, t, \hat{\mathcal{Q}}_{>\kappa(t)}))\right\|_1, \tag{5}$$

The DCT term explicitly penalizes spectral discrepancies, thereby promoting smoother and more temporally consistent predictions over long horizons. Complementing the reconstruction objective, the quantization loss adopts the standard commitment formulation:

$$\mathcal{L}_{\text{VQ}} = \sum_{i=1}^{K}\left\|\psi(\mathbf{q}_i) - \text{sg}(\hat{\mathbf{q}}_i)\right\|_2^2, \tag{6}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator [35]. To further stabilize training and mitigate codebook collapse, codebook entries are updated using exponential moving averages (EMA) of the assigned embeddings, with rarely used entries, identified by low EMA usage, periodically reinitialized from randomly sampled embeddings in the current batch [35, 36].

Detokenization. Given action tokens $C = (c_1, \ldots, c_K)$, each c_i is mapped to embedding $\hat{\mathbf{q}}_i \in \mathcal{C}$, yielding $\hat{\mathbf{Q}} = (\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_K)$. The action is then decoded via flow matching by solving an ODE:

$$\mathbf{x}_1 = \mathbf{x}_0 + \int_0^1 v_\theta\Big(\mathbf{x}_t, t \mid \tilde{\mathbf{Q}}_t\Big)\, dt, \qquad \mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{7}$$

where v_θ denotes the velocity field and $\tilde{\mathbf{Q}}_t = (\mathbf{0}, \ldots, \mathbf{0}, \hat{\mathbf{q}}_{\kappa(t)+1}, \ldots, \hat{\mathbf{q}}_K)$ retains only the last $K - \kappa(t)$ embeddings via a conditional annealing scheme.

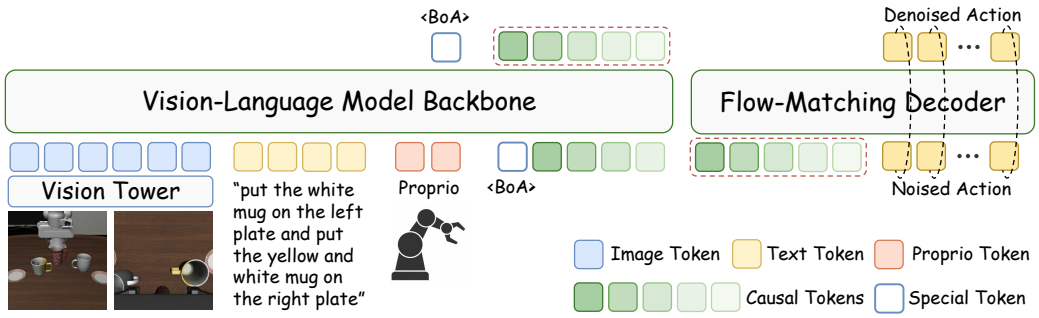


Figure 3: **An Overview of VLA-CATOK**. VLA-CATOK feeds visual, language, and proprioceptive tokens into a VLM backbone to autoregressively predict causal action tokens. These tokens are mapped via the learned codebook and processed by a frozen MMDiT flow-matching decoder to produce the final action in a single denoising step. Discrete tokens serve as the sole interface, naturally insulating the VLM's pretrained knowledge from action-specific gradients.

3.4 VLA-CATOK

To evaluate CATOK in robotic policy learning, we integrate it into a standard autoregressive VLA architecture [2, 13]. As shown in Figure 3, VLA-CATOK takes the current image I_t, language instruction l, and optional proprioceptive state s_t as input, and autoregressively predicts a sequence of action tokens $C = (c_1, \ldots, c_K)$. Each predicted token c_i is mapped to its VQ embedding $\hat{\mathbf{q}}_i$ through the learned codebook, forming $\hat{\mathbf{Q}} = (\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_K)$. A frozen MMDiT flow-matching decoder then converts $\hat{\mathbf{Q}}$ into the final continuous action chunk in a single decoding step.

During VLA training, the CATOK tokenizer and decoder are kept frozen, and only the autoregressive VLA backbone is updated. The backbone is trained with teacher-forced next-token prediction using a cross-entropy loss over CATOK tokens:

$$\mathcal{L}_{AR} = -\sum_{i=1}^{K} \log p_\theta \left(c_i | I_t, s_t, l, c_{<i}\right) \tag{8}$$

Unlike approaches that supervise actions directly and thus propagate action gradients back into the VLM backbone, our framework uses discrete tokens as the interface between the two modules and freezes the pretrained decoder, inherently insulating knowledge between the two modules. This substantially preserves the pretrained knowledge embedded in the VLM, thereby strengthening its instruction-following capability [26].

4 Experiments

To comprehensively evaluate the effectiveness of CATOK, we conduct a series of experiments covering both the tokenizer itself and its integration into downstream VLA policies. Section 4.1 introduces the tokenizer baselines, VLA backbones, and evaluation benchmarks, including both simulation and real-world settings. Section 4.2 examines whether CATOK improves downstream VLA performance and training efficiency. Section 4.3 analyzes whether CATOK produces causally ordered tokens with generative semantics. Finally, Section 4.4 studies the intrinsic properties of the tokenizer, including reconstruction fidelity, compression efficiency, and tokenization latency.

4.1 Experimental Setup

Tokenizer Baselines. We compare CATOK with three other representative action tokenizers for autoregressive policy learning: **BIN** [2], which uniformly discretizes each action dimension into 256 bins; **FAST** [13], which applies DCT-based action compression followed by BPE tokenization; and **OAT** [15], a learning-based VQ-VAE action tokenizer that enforces token-level causal structure through prefix-based masking and decoding. All tokenizers are integrated into the same autoregressive policy framework, and we further analyze their intrinsic token properties in Section 4.4.

Table 1: **Comparison on robotic manipulation simulation benchmarks.** CATOK achieves the best overall performance across all benchmarks, with particularly significant improvements on long-horizon tasks such as Long in LIBERO and *StackGreenCubeOnYellowCube* in SimplerEnv, demonstrating the effectiveness of causal action tokenization for modeling long-range temporal dependencies.

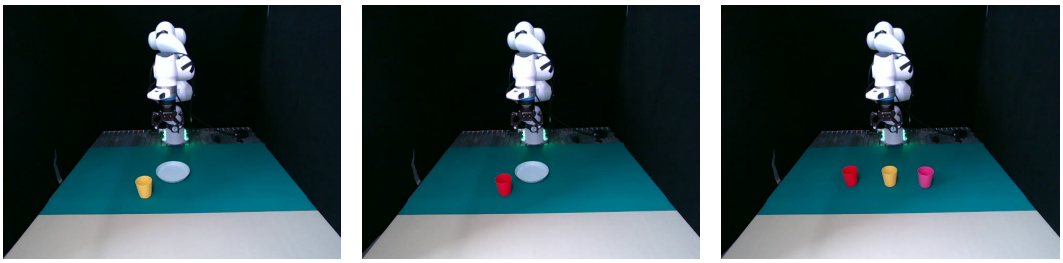
	LIBERO					SimplerEnv					RoboTwin 2.0	
Method	Spatial	Object	Goal	Long	Avg.	Spoon	Carrot	Stack	Eggplant	Avg.	Clean	Randomized
BIN	0.586	0.878	0.680	0.604	0.687	**0.542**	0.333	0.208	**0.708**	0.448	0.213	0.221
FAST	0.960	**0.998**	**0.962**	0.901	0.955	0.500	0.375	0.375	0.625	0.469	0.478	0.478
OAT	0.428	0.876	0.704	0.276	0.571	0.417	0.208	0.167	0.667	0.365	0.229	0.233
CATOK	**0.978**	0.994	0.954	**0.910**	**0.959**	0.458	**0.417**	**0.542**	0.542	**0.490**	**0.489**	**0.531**

Simulation Benchmarks. We evaluate downstream policy performance on three simulation benchmarks: **LIBERO** [37], **SimplerEnv** [38], and **RoboTwin 2.0** [39]. LIBERO consists of four official suites covering diverse long-horizon manipulation tasks. SimplerEnv evaluates zero-shot generalization on real-world-inspired robotic manipulation tasks using the WidowX platform. RoboTwin 2.0 provides large-scale embodied manipulation benchmarks under both clean and randomized environments, covering 50 tasks with substantial scene diversity.

Real-World Benchmark. We further evaluate CATOK on a real Franka manipulator under three task suites, as shown in Fig.4: *Pick-Spatial*, *Pick-Color*, and *Stack-Long*. The tasks are designed to test spatial generalization, color generalization, and long-horizon task completion, respectively.

For training, we collect 362 trajectories for *Pick-Cups* task and 220 trajectories for *Stack-Cups* task. Using Qwen3.5-VL as the shared backbone, we train QwenCATok and QwenFAST with the same training protocol and different action tokenizers. To ensure a comparable reconstruction quality between tokenizers, CATok is pretrained on the collected real-robot data together with InternA1 joint data, yielding reconstruction accuracy comparable to FAST.

Training and Evaluation Protocols. For LIBERO, we fine-tune π_0**-FAST-base** [13] on all four suites and report the average success rate over 40 tasks with 50 rollouts per task. For the more challenging SimplerEnv and RoboTwin 2.0 benchmarks, all methods are trained from scratch using **Qwen3-VL-4B-Instruct** [21] as the shared VLM backbone to ensure fair comparison. In SimplerEnv, policies are trained on Bridge [40] and evaluated zero-shot on the four WidowX tasks following the official protocol. In RoboTwin 2.0, we train on the official clean and randomized datasets, and evaluate on all 50 tasks under both clean and randomized settings with 20 rollouts per task. Additional implementation and evaluation details are provided in the Appendix A.



(a) Pick-Spatial (b) Pick-Color (c) Stack-Long

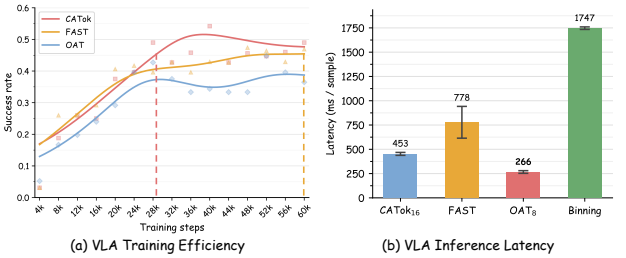
Figure 4: **Real-world scene layouts for the three task suites.** From left to right: *Pick-Spatial*, testing spatial generalization by varying cup and plate positions; *Pick-Color*, testing color generalization with different unseen cup colors; and *Stack-Long*, testing long-horizon manipulation under different spatial layouts.

4.2 Can CATOK Help VLA Training?

CATOK consistently improves VLA success rates across simulation benchmarks. As shown in Table 1, CATOK consistently outperforms all baseline tokenizers across the three simulation bench-

marks. Notably, it exhibits particularly pronounced gains on long-horizon manipulation tasks. In particular, on the *StackGreenCubeOnYellowCube* task in SimplerEnv, CATOK improves the success rate by more than 16.7% relative to the strongest baseline, highlighting its superior ability to model extended sequences of coordinated actions. These results suggest that introducing causal ordering into action tokenization enables autoregressive policies to capture long-range temporal dependencies more effectively, which is crucial for complex long-horizon manipulation tasks.

CATOK improves both VLA training efficiency and inference speed. Figure 5 (a) compares the success rates of CATOK and baseline tokenizers across training steps, highlighting the superior training efficiency of CATOK. CATOK consistently achieves higher success rates, particularly during the later stages of training, indicating both faster convergence and more stable learning dynamics. Notably, CATOK matches the best performance achieved by FAST with only approximately 50% steps, demonstrating substantially improved training efficiency. We attribute this improvement primarily to the causal token structure analyzed in Section 4.3. By organizing action chunks into causally ordered tokens, CATOK naturally aligns the tokenizer with the autoregressive policy objective, enabling more effective stage-wise action modeling rather than learning from weakly structured discrete codes. Consequently, during autoregressive generation, each predicted token provides more informative guidance for subsequent token prediction and trajectory generation, consistently improving both training efficiency and policy performance across diverse tasks. As shown in Figure 5 (b), CATOK also exhibits highly competitive inference efficiency. By jointly considering autoregressive token generation and tokenizer decoding overhead, it achieves near-minimal end-to-end VLA inference latency among all compared methods.



(a) VLA Training Efficiency (b) VLA Inference Latency

Figure 5: (a) Training efficiency on the SimplerEnv benchmark. Curves are Gaussian-smoothed to reduce evaluation stochasticity without changing the relative ranking of different methods. Notably, CATOK reaches the best baseline performance using only around 50% of the training steps. (b) VLA inference latency. By jointly considering autoregressive generation and decoding overhead, CATOK achieves near-minimal end-to-end inference latency while attaining the best success rate among all methods.

Method	Pick-Spatial	Pick-Color	Stack-Long
FAST	0.450	0.300	0.275
CATOK	**0.650**	**0.500**	**0.475**

Table 2: **Real-world manipulation results.** CATOK consistently outperforms FAST across spatial, color, and long-horizon generalization tasks.

CATOK transfers effectively to real-world robots. To further evaluate whether the benefits of causal action tokenization extend beyond simulation, we deploy CATOK on a real-world Franka robot. As shown in Table 2, CATOK consistently outperforms FAST across three task suites covering spatial generalization, color generalization, and long-horizon manipulation. Notably, CATOK achieves a consistent 20 percentage-point improvement over FAST across all three settings, demonstrating that the benefits of causal action tokenization transfer to real-world robotic manipulation.

4.3 Does CATOK Produce Causal Tokens with Generative Semantics?

Causal Predictive Ordering of Action Tokens. We test whether CATOK tokens exhibit a predictive order by training a lightweight action-only autoregressive transformer for each tokenizer and measuring next-token entropy on held-out LIBERO-Bridge Mixture chunks. At each position k, we compute $H_k = -\sum_v p_{k,v} \log p_{k,v}$ from the prediction logits; lower entropy indicates that preceding tokens provide stronger predictive context.

As shown in Figure 6(a), CATOK exhibits a clear decreasing entropy trend when tokens are predicted in their original order, indicating that later tokens become progressively easier to predict as more prefix tokens are observed. When predicted in reverse order, the trend is inverted, demonstrating that the learned token ordering is directional rather than arbitrary. In contrast, FAST remains nearly

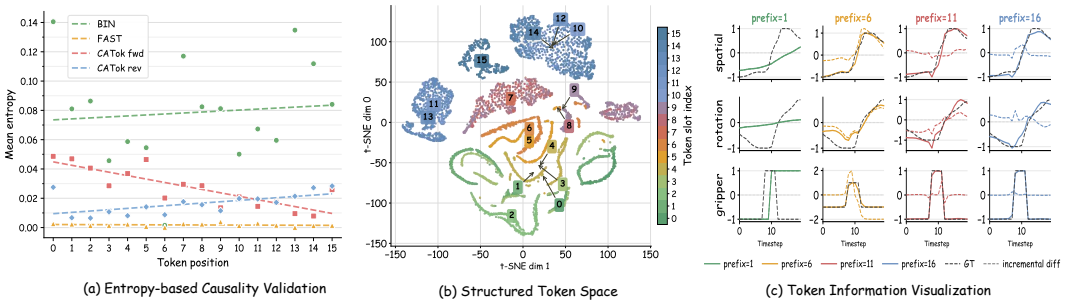


Figure 6: **CATOK Produces Causal Tokens with Generative Semantics**. **(a) Causal Ordering** CATOK 's entropy decreases in the forward order (*CATOK fwd*) and increases in the reverse order (*CATOK rev*), while FAST and Bin show weaker positional structure. **(b) Token embedding geometry.** T-SNE map shows slot-dependent organization, progressing from smooth early-token regions to compact late-token clusters. **(c) Prefix reconstruction.** Increasing token prefixes produce structured trajectory updates, with dashed curves showing incremental contributions.

flat and Bin shows a slight increase, suggesting substantially weaker positional dependency. These results indicate that CATOK learns causally ordered action tokens that are naturally aligned with autoregressive generation.

Stage-wise Generative Semantics. We next examine whether CATOK tokens carry generative semantics, i.e., whether different tokens induce distinct and structured changes in the decoded action chunk. We use two complementary probes: token-space geometry and prefix reconstruction.

Token-space geometry. We visualize VQ embeddings from 32,768 normalized LIBERO action chunks using t-SNE, with embeddings extracted from a frozen CATOK checkpoint (K=16, codebook size 4096, d_{vq}=16). As shown in Figure 6(b), the embeddings are clearly organized by token slot: early slots form smooth elongated regions, middle slots become more diffuse, and late slots form compact, separated clusters. This slot-dependent geometry suggests that CATOK tokens are not exchangeable codebook indices, but occupy distinct generative roles that progress from shared motion factors to specialized residual corrections.

Prefix reconstruction. We reconstruct action chunks from token prefixes with $n \in \{1, 6, 11, 16\}$. As shown in Figure 6(c), adding tokens induces coherent trajectory updates, from structural changes in spatial, rotation, and gripper dimensions to finer residual refinements. This indicates that CATOK tokens serve as meaningful generative increments rather than passive compression codes.

Together, the geometry and reconstruction probes show that CATOK learns a coarse-to-fine hierarchy of generative action tokens. Earlier tokens capture shared global structure, while later tokens contribute increasingly specialized residual refinements. This stage-wise organization complements the causal ordering above and provides a natural inductive bias for autoregressive action generation.

We further validate the causal semantics of individual token positions through native-decoder intervention experiments, including token swapping and single-token removal. These experiments provide direct evidence that different token positions control distinct stages of action generation. Detailed experimental settings and results are provided in Appendix C.1.

4.4 Intrinsic Properties of CATOK

CATOK balances reconstruction fidelity and compression efficiency. Action tokenization inherently requires balancing reconstruction fidelity and compression efficiency. We evaluate reconstruction quality using Valid Reconstruction Rate (VRR) [14] and measure compactness using Compression Rate (CR). Detailed metric definitions are provided in Appendix B.

As shown in Figure 7 (a-c), CATOK achieves near-perfect reconstruction fidelity, with VRR comparable to Binning and FAST and substantially higher than OAT under the same token budget. Meanwhile, it provides significantly better compression than Binning and FAST while remaining comparable to OAT, resulting in the best overall fidelity–compression tradeoff.

Encoding and decoding latency. To evaluate the efficiency of action tokenization during both training and inference, we measure the average encoding and decoding latency during action tokenization.

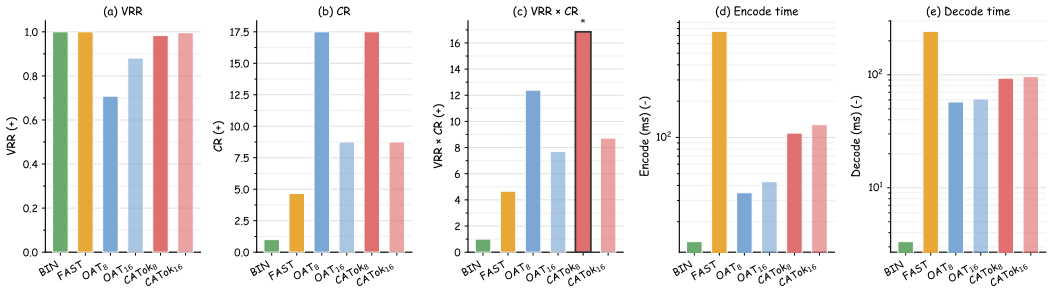


Figure 7: **Intrinsic properties of action tokenizers. (a–c) Fidelity vs. compression efficiency.** VRR measures reconstruction quality; CR measures compression ratio; VRR×CR jointly captures the fidelity–compression tradeoff. CATOK_8 achieves the highest VRR×CR (16.85), outperforming all baselines at the same compression level. **(d–e) Inference efficiency.** Encode and decode latencies are shown on a log scale. The subscript k in OAT_k and CATOK_k denotes the number of tokens.

Detailed measurement protocols are provided in Appendix B. Since CATOK substantially outperforms Binning and OAT in downstream VLA performance and achieves performance comparable to FAST, while consistently obtaining slightly higher success rates overall, FAST serves as the most relevant latency baseline. As shown in Figure 7 (d-e), compared with FAST, CATOK achieves substantially lower encoding and decoding latency while also delivering slightly better downstream VLA performance, demonstrating superior time efficiency for both training and inference.

4.5 Ablations and Analysis

We conduct ablation studies on key components and design choices in CATOK. Specifically, we analyze the effect of the flow-matching head, where the results show that the complete design consistently achieves the best performance. In addition, we study the impact of the number of tokens, a key design choice in CATOK, on downstream VLA performance. Detailed ablation settings and additional results are provided in Appendix C.

5 Conclusion

We introduced CATOK, a causal action tokenizer for purely autoregressive VLA models. By transferring the stage-wise causal structure of flow matching into the token space through conditional annealing, CATOK learns compact action tokens with causal ordering and generative semantics. The resulting token sequence enables autoregressive policies to generate continuous action chunks through a coarse-to-fine factorization, while a frozen MMDiT-based flow-matching decoder preserves high-fidelity action reconstruction. Experiments across multiple simulation benchmarks and real-world robotic manipulation tasks demonstrate that CATOK improves downstream task performance, training efficiency, and the fidelity–compression tradeoff over existing action tokenizers, while maintaining competitive inference efficiency. Our analysis further verifies the causal ordering and generative semantics of the learned token sequence. These results highlight causal and generative action tokenization as a promising approach for scalable purely autoregressive VLA systems.

Limitations. Despite the promising results, our real-world evaluation is currently limited to a single robot platform and a finite set of manipulation tasks. Therefore, the robustness of CATOK across diverse robot embodiments, sensing configurations, environmental conditions, and dynamics remains unexplored. In addition, our experiments are conducted on relatively constrained datasets, and we have not yet investigated the scalability of CATOK on large-scale and diverse cross-embodiment robotic data. Future work will evaluate CATOK across broader real-world embodiments and large-scale robotic datasets to study its scalability and generalization under more diverse real-world settings.

Acknowledgments and Disclosure of Funding

This research was supported by the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, and the Shanghai Qi Zhi Institute, which provided research funding and computational resources. The authors declare no competing interests.

References

[1] Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[2] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[3] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[4] Dibya Ghosh et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[6] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $pi_0.5$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[7] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[8] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. *arXiv preprint arXiv:2512.13030*, 2025.

[9] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[10] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Worldvla: Towards autoregressive action world model, 2025.

[11] Yi Liu, Sukai Wang, Dafeng Wei, Xiaowei Cai, Linqing Zhong, Jiange Yang, Guanghui Ren, Jinyu Zhang, Maoqing Yao, Chuankang Li, et al. Unified embodied vlm reasoning with robotic action via autoregressive discretized pre-training. *arXiv preprint arXiv:2512.24125*, 2025.

[12] Yutong Hu, Jan-Nico Zaech, Nikolay Nikolov, Yuanqi Yao, Sombit Dey, Giuliano Albanese, Renaud Detry, Luc Van Gool, and Danda Paudel. Ar-vla: True autoregressive action expert for vision-language-action models, 2026.

[13] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[14] Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, et al. Faster: Toward efficient autoregressive vision language action modeling via neural action tokenization. *arXiv preprint arXiv:2512.04952*, 2025.

[15] Chaoqi Liu et al. OAT: Ordered action tokenization. *arXiv preprint arXiv:2602.04215*, 2026.

[16] Zhongqi Yue, Jiankun Wang, Qianru Sun, Lei Ji, Eric I-Chao Chang, and Hanwang Zhang. Exploring diffusion time-steps for unsupervised representation learning. *arXiv preprint arXiv:2401.11430*, 2024.

[17] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[18] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[19] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023.

[20] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[21] Qwen Team. Qwen3 technical report, 2025.

[22] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[23] Ankit Goyal, Hugo Hadfield, Xuning Yang, Valts Blukis, and Fabio Ramos. Vla-0: Building state-of-the-art vlas with zero modification. *arXiv preprint arXiv:2510.13054*, 2025.

[24] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[25] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[26] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

[27] Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, et al. Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning. *arXiv preprint arXiv:2506.06072*, 2025.

[28] Zibin Dong, Yicheng Liu, Shiduo Zhang, Baijun Ye, Yifu Yuan, Fei Ni, Jingjing Gong, Xipeng Qiu, Hang Zhao, Yinchuan Li, et al. Actioncodec: What makes for good action tokenizers. *arXiv preprint arXiv:2602.15397*, 2026.

[29] Renming Huang, Chendong Zeng, Wenjing Tang, Jintian Cai, Cewu Lu, and Panpan Cai. Mimic intent, not just trajectories. *arXiv preprint arXiv:2602.08602*, 2026.

[30] Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. *arXiv preprint arXiv:2507.01016*, 2025.

[31] Bohan Wang, Zhongqi Yue, Fengda Zhang, Shuo Chen, Li'an Bi, Junzhe Zhang, Xue Song, Kennard Yanting Chan, Jiachun Pan, Weijia Wu, et al. Selftok: Discrete visual tokens of autoregression, by diffusion, and for reasoning. *arXiv preprint arXiv:2505.07538*, 2025.

[32] Bohan Wang, Mingze Zhou, Zhongqi Yue, Wang Lin, Kaihang Pan, Liyu Jia, Wentao Hu, Wei Zhao, and Hanwang Zhang. Selftok-zero: Reinforcement learning for visual generation via discrete and autoregressive visual tokens. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[33] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

[34] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.

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

[36] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[37] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023.

[38] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation, 2024.

[39] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025.

[40] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

A Implementation Details

Table 3 provides implementation details for the action tokenizer training, VLA training, and evaluation protocols used in our experiments. In particular, for LIBERO and Bridge, we train a tokenizer on the combined datasets with mix ratios of 1.0 and 5.0, respectively, and then train the VLA separately on each dataset. For RoboTwin, we train both the tokenizer and the VLA on the full RoboTwin dataset.

Table 3: Benchmark-specific implementation details.

Setting	**LIBERO**	**Simpler-Bridge**	**RoboTwin 2.0**
Benchmark and embodiment			
Embodiment	Franka	WidowX	Bimanual ALOHA
Action DoF	7	7	14
Training data	Official LIBERO dataset	Bridge dataset	Clean and randomized RoboTwin 2.0 trajectories
Evaluation protocol	4 suites 10 tasks per suite 50 rollouts per task	4 WidowX tasks in SimplerEnv 24 rollouts per task	50 tasks 20 rollouts per task
Action tokenizer configuration			
Action horizon H	20	10	20
Number of action tokens K	16	16	32
Codebook size $\lvert\mathcal{C}\rvert$	4096	4096	4096
Tokenizer training steps	300 000	300 000	200 000
Tokenizer batch size	512	512	512
Tokenizer learning rate	1×10^{-4}	1×10^{-4}	1×10^{-4}
Tokenizer weight decay	1×10^{-2}	1×10^{-2}	1×10^{-2}
Tokenizer warmup steps	1000	1000	1000
VLA training configuration			
VLA backbone	π_0-FAST-base	Qwen3-VL-4B-Instruct	Qwen3-VL-4B-Instruct
VLA training steps	30 000	60 000	40 000
VLA batch size	32	64	32
VLA learning rate	2.5×10^{-5}	1×10^{-5}	1×10^{-5}
VLA weight decay	1×10^{-10}	1×10^{-8}	1×10^{-8}
Number of GPUs	4×H100 GPUs	4×H100 GPUs	4×H100 GPUs

B Evaluation Metrics

In this section, we provide the detailed definitions and computation protocols for the evaluation metrics used to assess the intrinsic properties and efficiency of action tokenizers.

B.1 Valid Reconstruction Rate (VRR)

To evaluate reconstruction fidelity, we adopt the Valid Reconstruction Rate (VRR) [14], defined as the proportion of reconstructed action chunks whose normalized reconstruction error falls below a predefined threshold:

$$\mathrm{VRR} = \frac{N_{\mathrm{valid}}}{N_{\mathrm{total}}}, \quad N_{\mathrm{valid}} = \sum_{i=1}^{N_{\mathrm{total}}} \mathbf{1}\left(\frac{\| A_i^{\mathrm{pred}} - A_i^{\mathrm{gt}} \|_2}{H} < \sigma \right), \tag{9}$$

where $A_i^{\mathrm{pred}}, A_i^{\mathrm{gt}} \in \mathbb{R}^{H \times D}$ denote the reconstructed and ground-truth action chunks for the i-th sample, respectively. Here, H is the prediction horizon and D is the action dimension. The ℓ_2 norm is computed over both temporal and action dimensions, and normalization by H yields the average per-step reconstruction error. σ denotes a predefined threshold, and $\mathbf{1}(\cdot)$ is the indicator function.

Table 4: Controlled ablation of conditional annealing, random masking, and decoder architecture.

Method	Spatial	Object	Goal	Long	Avg.
A - Ours	0.978	0.994	0.954	0.910	0.959
B - Ours w/o conditional annealing	0.920	0.980	0.918	0.834	0.914
C - VQ-VAE + MMDiT + random masking	0.724	0.924	0.756	0.542	0.737
D - VQ-VAE + MMDiT	0.614	0.796	0.296	0.240	0.486

B.2 Compression Rate (CR)

To measure reconstruction efficiency, we use the Compression Rate (CR), defined as the ratio between the size of the original continuous action representation N_{raw} and the compressed discrete token sequence N_{token}:

$$\text{CR} = \frac{N_{\text{raw}}}{N_{\text{token}}}, \quad N_{\text{raw}} = B \times H \times D, \tag{10}$$

where B denotes the batch size, H is the action horizon, and D is the action dimension. A higher CR indicates more compact action representations.

B.3 Encoding and Decoding Latency

To evaluate the efficiency of action tokenization during both training and inference, we measure the average encoding and decoding latency. Since action encoding is performed during training, encoding latency directly affects training efficiency. In contrast, decoding latency primarily impacts inference-time efficiency. Specifically:

- **Encoding time (encode_time)**: the average time required to convert continuous action chunks into discrete token sequences.
- **Decoding time (decode_time)**: the average time required to reconstruct continuous actions from discrete tokens.

All latency measurements are averaged across batches under the same hardware and evaluation settings.

C Ablations and Analysis

In this section, we provide detailed ablation settings, additional experimental results, and further analysis for the key design components and design choices in CATOK.

Conditional Annealing and Architecture Ablations. To isolate the contribution of conditional annealing from the effects of decoder architecture and model capacity, we consider four variants: (A) our full model with conditional annealing, (B) CATOK without conditional annealing, (C) a VQ-VAE with the same MMDiT decoder and comparable parameter count trained with random token masking, and (D) a VQ-VAE with the same MMDiT decoder and comparable parameter count without random masking.

Removing conditional annealing (A $\rightarrow$ B) leads to a clear performance drop from 0.959 to 0.914, demonstrating its importance beyond the underlying flow-matching decoder and transformer architecture. Moreover, both C and D remain substantially below our method despite using the same MMDiT decoder and comparable model capacity, showing that the improvement cannot be explained solely by decoder architecture or model capacity. Notably, C already employs random token masking yet achieves only 0.737, indicating that random masking alone is insufficient to induce the desired causal structure in the token space. Together, these controlled comparisons demonstrate that conditional annealing is a critical factor in organizing token representations into a structured and causally ordered representation.

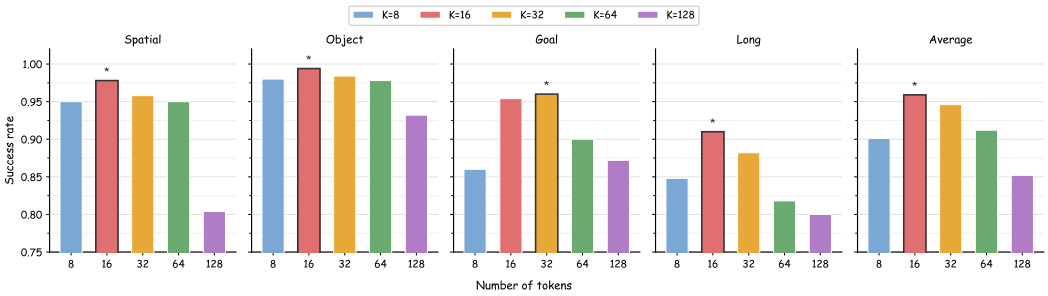


Figure 8: Effect of the number of action tokens on downstream VLA performance on the LIBERO benchmark. Performance improves from 8 to 16 tokens but degrades with larger token budgets, with 16 tokens achieving the best overall performance.

Number of Tokens. We further analyze the impact of the number of action tokens in CATOK on downstream VLA performance. Specifically, we evaluate CATOK with different token budgets $(8, 16, 32, 64,$ and 128 tokens) while keeping all other training and architecture settings fixed. The results on the LIBERO benchmark are shown in Figure 8. Increasing the number of tokens from 8 to 16 consistently improves downstream VLA performance, while further increasing the token budget beyond 16 leads to performance degradation. We attribute this degradation to the increased difficulty of autoregressive modeling over excessively long token sequences, which reduces token efficiency and negatively impacts downstream policy learning. Overall, 16 tokens provide the best balance between representation capacity and autoregressive learning efficiency.

C.1 Causality Validation

To further verify that the learned token ordering corresponds to genuine causal roles in action generation, we perform intervention experiments using the frozen native decoder of CATOK. Specifically, we consider two types of interventions: *donor-swap* and *single-token removal*. Both experiments directly modify individual token positions while keeping the remaining tokens unchanged, allowing us to measure the causal influence of each token on the reconstructed action trajectory.

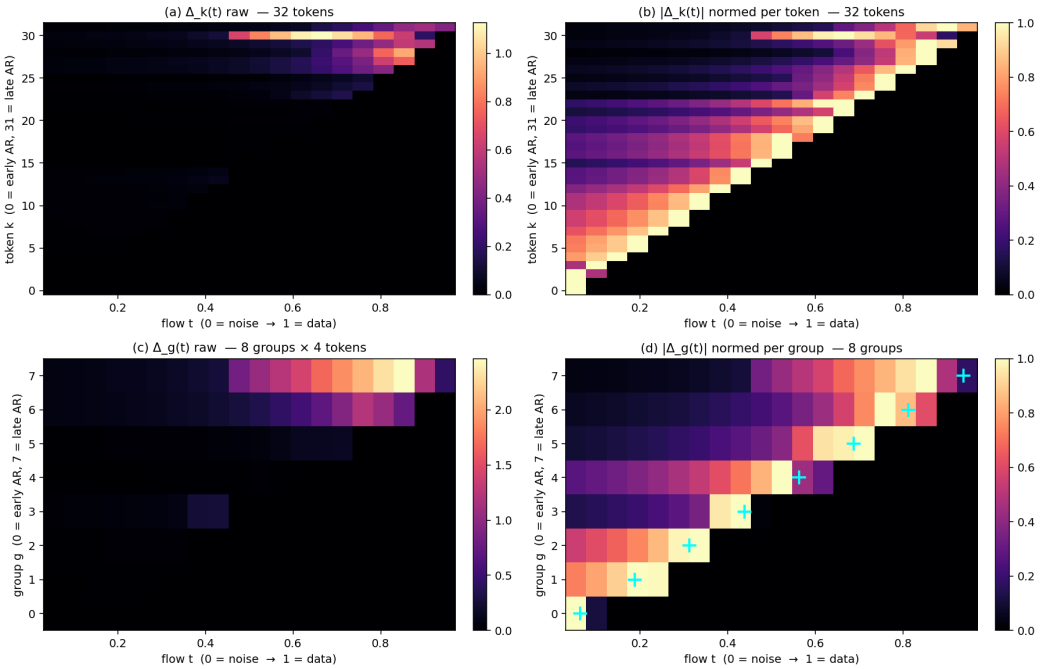
Donor-Swap Intervention. We first evaluate the influence of each token position by replacing the token at position k with the corresponding token from another action chunk, while keeping all other tokens unchanged. We then decode the intervened token sequence using the native flow-matching decoder and measure the resulting deviation from the original reconstruction. By performing this intervention at different token positions and flow-matching time steps, we obtain a position-dependent influence map. If the tokens follow a causal coarse-to-fine ordering, early tokens should primarily affect coarse action structure, while later tokens should exert stronger influence during later stages of generation.

Single-Token Removal. We further perform a single-token removal experiment by replacing the token at position k with a neutral embedding while keeping all other tokens unchanged. The resulting action reconstruction is compared with the reconstruction from the complete token sequence. This experiment isolates the contribution of individual token positions and provides a complementary measure of their generative influence.

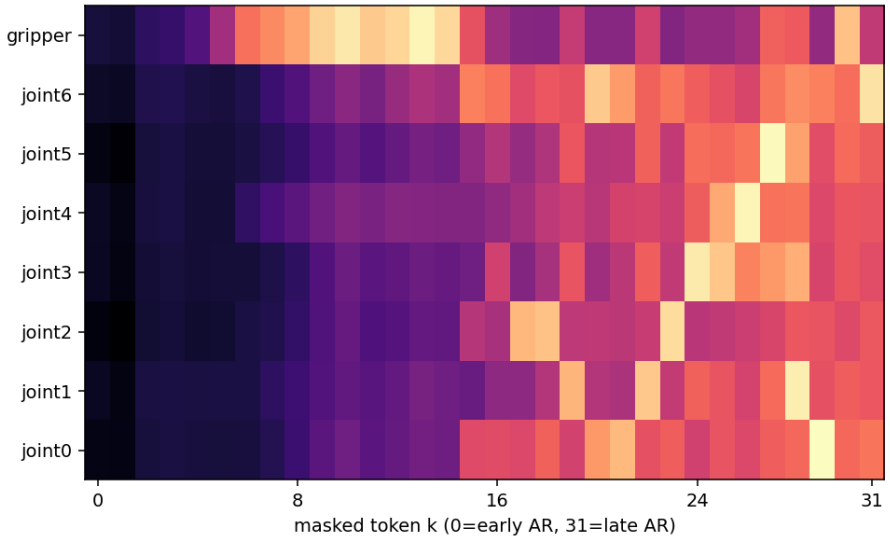
Results. As shown in Figure 9, the intervention effects exhibit a clear position-dependent structure. Early tokens have stronger influence on coarse action components and earlier generation stages, whereas later tokens increasingly affect fine-grained action details. The consistent stage-specific influence observed under both donor-swap and single-token removal interventions provides direct evidence that the token ordering learned by CATOK corresponds to meaningful causal roles rather than merely reflecting an arbitrary positional ordering.

D Full Robotwin 2.0 Benchmark

Table 5 provides the full per-task RoboTwin 2.0 results corresponding to the aggregate scores in Table 1. We report success rates for both clean and randomized evaluation settings across all 50 tasks, with 20 rollouts per task.



(a) Donor-swap influence matrix $\Delta(t)$. Replacing token k with a donor's same-position embedding yields a position- and flow-stage-dependent influence map with triangular support, confirming the conditional-annealing schedule.



(b) Single-token removal. Zeroing token k and decoding the intervened sequence isolates each position's generative contribution; the per-dimension footprint progresses from coarse gripper/task-state information (early tokens) to fine joint/trajectory-detail information (late tokens).

Figure 9: Causality validation via native-decoder interventions. Both donor-swap (a) and single-token removal (b) exhibit a clear position-dependent structure: early tokens have stronger influence on coarse action components and earlier generation stages, whereas later tokens increasingly affect fine-grained action details, confirming that the learned token ordering corresponds to meaningful causal roles rather than an arbitrary positional ordering.

Table 5: Per-task success rates on RoboTwin 2.0.

Simulation Task	BIN		FAST		OAT		CATOK	
	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
Adjust Bottle	80%	95%	90%	100%	100%	100%	90%	100%
Beat Block Hammer	15%	20%	25%	50%	15%	5%	65%	65%
Blocks Ranking RGB	0%	0%	10%	10%	5%	0%	5%	0%
Blocks Ranking Size	0%	0%	0%	5%	5%	0%	5%	10%
Click Alarmclock	65%	60%	90%	75%	60%	70%	65%	80%
Click Bell	95%	85%	80%	80%	75%	50%	95%	95%
Dump Bin Bigbin	30%	30%	95%	75%	15%	20%	65%	70%
Grab Roller	65%	50%	100%	100%	55%	60%	100%	100%
Handover Block	0%	0%	0%	0%	0%	0%	10%	10%
Handover Mic	10%	10%	55%	50%	5%	10%	60%	65%
Hanging Mug	0%	0%	10%	15%	0%	0%	15%	5%
Lift Pot	25%	0%	45%	45%	0%	0%	30%	40%
Move Can Pot	0%	0%	25%	50%	15%	15%	50%	55%
Move Pillbottle Pad	0%	10%	55%	40%	15%	25%	50%	70%
Move Playingcard Away	30%	65%	65%	80%	45%	60%	90%	85%
Move Stapler Pad	0%	5%	10%	10%	5%	0%	15%	10%
Open Laptop	0%	15%	70%	55%	20%	20%	75%	90%
Open Microwave	0%	5%	15%	20%	20%	15%	20%	35%
Pick Diverse Bottles	0%	0%	40%	45%	0%	5%	35%	40%
Pick Dual Bottles	0%	0%	30%	55%	0%	5%	60%	40%
Place A2B Left	40%	30%	85%	80%	40%	40%	75%	80%
Place A2B Right	35%	25%	65%	70%	45%	35%	65%	70%
Place Bread Basket	25%	20%	70%	40%	20%	20%	45%	65%
Place Bread Skillet	40%	35%	50%	50%	15%	10%	45%	60%
Place Burger Fries	5%	10%	80%	70%	25%	40%	80%	80%
Place Can Basket	0%	5%	20%	30%	0%	5%	30%	50%
Place Cans Plasticbox	0%	0%	50%	25%	0%	0%	30%	35%
Place Container Plate	55%	60%	85%	96%	65%	50%	90%	90%
Place Dual Shoes	5%	0%	10%	20%	0%	0%	0%	15%
Place Empty Cup	35%	35%	80%	85%	35%	10%	50%	75%
Place Fan	0%	10%	40%	40%	5%	5%	25%	30%
Place Mouse Pad	5%	10%	30%	30%	5%	5%	30%	55%
Place Object Basket	5%	5%	45%	25%	20%	25%	50%	60%
Place Object Scale	10%	30%	70%	45%	20%	30%	40%	50%
Place Object Stand	35%	45%	80%	90%	35%	50%	75%	85%
Place Phone Stand	25%	0%	65%	55%	10%	0%	40%	75%
Place Shoe	25%	25%	50%	40%	25%	50%	75%	50%
Press Stapler	45%	45%	60%	55%	55%	75%	85%	85%
Put Bottles Dustbin	0%	0%	5%	0%	0%	0%	5%	0%
Put Object Cabinet	0%	0%	25%	35%	0%	0%	45%	40%
Rotate QRcode	10%	10%	30%	35%	0%	5%	60%	35%
Scan Object	0%	10%	20%	25%	0%	5%	25%	20%
Shake Bottle	95%	85%	100%	100%	85%	100%	90%	95%
Shake Bottle Horizontally	90%	95%	100%	95%	90%	85%	100%	85%
Stack Blocks Three	0%	0%	0%	10%	0%	0%	0%	0%
Stack Blocks Two	10%	10%	50%	40%	15%	5%	40%	50%
Stack Bowls Three	0%	0%	10%	10%	0%	0%	20%	15%
Stack Bowls Two	25%	25%	40%	75%	45%	20%	50%	60%
Stamp Seal	5%	15%	25%	40%	15%	20%	40%	40%
Turn Switch	25%	15%	40%	20%	20%	15%	40%	40%
Average	21.3%	22.1%	47.8%	47.82%	22.9%	23.3%	48.9%	53.1%

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions of CATok, including causal action tokenization and the conditional annealing in flow matching while decoding. The claims are supported by the reconstruction and VLA benchmark evaluation results in Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a limitations discussion in Section 4: our real-world evaluation is currently limited to a single robot platform and a finite set of manipulation tasks, and we have not yet studied the scalability of CATOK on large-scale and diverse cross-embodiment robotic data. Future work will evaluate CATOK across broader real-world embodiments and datasets.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present theoretical theorems or proofs. The proposed method is motivated algorithmically and evaluated empirically.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the tokenizer architecture, datasets, evaluation protocols and metrics to obtain the main results. Additional details, hyperparameters, and evaluation settings are provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and documentation on how to run it are publicly available at https://chenyuzhangx.github.io/CATok/.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental section and Appendix A specify the datasets, tokenizer configuration, action horizon, and training steps. These details are sufficient to understand and reproduce the reported comparisons.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: For lightweight experiments such as model inference latency evaluation, we report the uncertainty to avoid randomness in the autoregressive inference process.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide a summary of the compute resources and training details in Appendix A, including the GPU hardware, training configurations, and evaluation protocols used for the reported experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses standard and open-source robot learning data benchmarks, and does not involve private user data or human-subjects experiments. We follow the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work is a methodological study on action tokenization for robot learning and does not introduce a deployed system with direct societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models, scraped datasets, or general-purpose generative systems.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets, models, codebases, and other external assets used in this work are properly credited in the paper and supplementary material through citations and acknowledgments. We also respect and follow the corresponding licenses and terms of use of these assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release code and documentation on how to run it at `https://chenyuzhangx.github.io/CATok/`. Together with the provided configurations, they are sufficient to reproduce our model.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human-subjects experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subjects experiments, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not use LLMs as important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.