BEYOND RECONSTRUCTION: WHAT MATTERS IN ACTION TOKENIZATION FOR ROBOT POLICIES?

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive action-token policies such as vision-language-action models require *action tokenizers* to translate discrete token sequences into precise control actions in continuous space. Many action tokenizers learn the mapping between tokens and actions via a reconstruction objective. However, as we show through extensive analysis, sufficiently accurate action reconstruction is only one part of what makes a downstream robot policy successful. It is also critical that the policy is *able to predict the right tokens* for new observations, and that *unseen policy token predictions still decode into reasonable actions*. These properties are downstream of tokenizer training and are not directly incentivized by a reconstruction objective alone. In this work, we introduce **Predictable and Robust Action Tokenization (ProAct)**, a tokenizer training method that strategically augments reconstruction with the goal of improving downstream predictability and robustness. ProAct is policy-agnostic and uses only action datasets for training. Across the Robomimic, LIBERO, and RoboTwin benchmarks and a diverse set of tokenizer architectures, ProAct improves rollout success by an average of 11.3 percentage points. These improvements also translate to vision-language-action policies and real-world robotic manipulation, yielding average gains of 21.8 and 36.7 percentage points, respectively. These results suggest that effective action tokenization should be designed as a policy interface—balancing fidelity, predictability, and robustness—rather than as a reconstruction problem alone.

1 INTRODUCTION

Many robot policies predict continuous action chunks (Zhao et al., 2023; Chi et al., 2023), whereas autoregressive vision-language models (VLMs) generate discrete token sequences. Action tokenization bridges this gap by encoding action chunks as tokens and decoding them back to continuous control actions. RT-1 (Brohan et al., 2023) discretizes robot actions, while RT-2 (Zitkovich et al., 2023) explicitly connects action tokens to pretrained VLMs and introduces the vision-language-action (VLA) formulation. Action tokenization has since become standard in autoregressive VLAs (Kim et al., 2024; Pertsch et al., 2025; Wang et al., 2025), in part because it preserves the native next-token interface of autoregressive VLMs. Discrete action tokens can represent multiple valid action modes, whereas regression objectives average across modes and yield an invalid intermediate action (Florence et al., 2022; Shafiullah et al., 2022; Lee et al., 2024). Moreover, action tokenization enables temporal compression (Mete et al., 2024; Pertsch et al., 2025) and language-model-style policy post-training (Galaxea Team, 2026).

This growing adoption of action tokenization raises a more specific question: how should continuous actions be tokenized? Independent per-dimension, per-timestep binning (Brohan et al., 2023) ignores temporal redundancy within action chunks and is inefficient (Appendix A); FAST (Pertsch et al., 2025) exploits such temporal structure through frequency-domain compression, but relies on a fixed analytical representation (Dong et al., 2026) and is tied to dataset-specific frequency statistics, making it less transferable across tasks or embodiments. Recent learned tokenizers report gains over these fixed tokenization schemes on selected benchmarks (Dong et al., 2026; Kang et al., 2026; Liu et al., 2026a); however, the principles for designing effective learned action tokenizers remain underexplored. Existing works often choose nominal bitrate, chunk size, and execution horizon empirically, while the properties of the resulting token space remain poorly understood. More concretely, tokenizer training and policy learning involve different aspects of the same discrete token

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.

space: the tokenizer focuses on reconstructing expert actions from encoded tokens, while the downstream autoregressive (AR) policy must predict and compose those tokens from observations. As a result, reconstruction quality alone does not characterize either how predictable the induced token targets are or how policy-generated token sequences decode into actions. These gaps expose a fundamental question: **"what matters in action tokenization for robot policies?"**

Across 22 tokenizer configurations that span key design choices (Lee et al., 2024; Mete et al., 2024; Wang et al., 2025; Liu et al., 2026b; Dong et al., 2026; Liu et al., 2026a; Kang et al., 2026), we find that tokenizer reconstruction error alone is not a reliable indicator of policy success. Specifically, a sufficiently low reconstruction error is necessary for successful control, but further reductions in error do not predict rollout performance. We further analyze how bitrate, chunk size, and the execution horizon affect downstream performance. Beyond reconstruction, our empirical and theoretical results identify an underexplored failure mode: autoregressive policies can produce token sequences that differ from the encoded targets, yet standard training objectives do not directly control the resulting action-space error. This observation motivates **Predictable and Robust Action Tokenization (ProAct)**, a tokenizer-training method with two components. *Token Balance Regularization* makes the encoded targets easier for the downstream policy to predict. *Token Perturbation* improves robustness to token mismatches. Neither requires access to the downstream policy or the observations. Together, we show that ProAct improves downstream rollout performance across nearly all evaluated tokenizer configurations. Ablating each component shows that both interventions can improve rollout performance even when reconstruction and policy-validation metrics do not consistently explain the gains. Decoded-error diagnostics support the intended robustness effect of Token Perturbation. Together, these results suggest that effective action tokenization requires balancing fidelity, predictability, and robustness to the sequences that the downstream policy actually produces.

Our contributions are threefold:

- We conduct a controlled study of 22 learned action tokenizers, characterizing the effects of reconstruction fidelity, nominal bitrate, chunk size, and execution horizon on downstream control. We find that sufficient reconstruction accuracy is necessary for successful rollouts, but that further improvements in reconstruction do not reliably predict better downstream performance.
- We show that AR policies routinely compose complete token sequences not observed in the encoded demonstrations, and develop a one-step error decomposition that distinguishes matched predictions from token mismatches, revealing that standard tokenizer and policy objectives do not control the action-space severity of mismatched predictions.
- Motivated by this analysis, we propose ProAct, which combines two complementary tokenizer-only training methods: Token Perturbation, which improves robustness to token mismatches, and Token Balance Regularization, which makes the encoded targets easier for the downstream policy to predict. We show that ProAct improves average success rate by up to 11.3 percentage points on Robomimic, LIBERO, and RoboTwin, by 21.8 points for a VLA policy on LIBERO, and by 36.7 points on real-world manipulation tasks.

2 Related Work

Imitation Learning and Action Chunking Imitation learning trains policies from expert demonstrations, commonly by behavior cloning (Ross et al., 2011). Action chunking extends this formulation by predicting a sequence of future actions and executing either the full chunk or a prefix before re-planning. It is used across transformer, diffusion, and VLA policies, including ACT (Zhao et al., 2023), Diffusion Policy (Chi et al., 2023), and OpenVLA-OFT (Kim et al., 2025). Alongside these empirical design choices, recent theory analyzes when open-loop chunk execution can mitigate compounding imitation error under stability assumptions (Zhang et al., 2025; Lazzati et al., 2026).

Action-Generation Paradigms in VLA Policies A primary distinction among VLA policies is their output representation. One family predicts continuous action chunks directly through an action head: π_0 (Black et al., 2025), $\pi_{0.5}$ (Physical Intelligence et al., 2025), and GR00T N1 (NVIDIA et al., 2025) use flow-matching or diffusion-based objectives, whereas OpenVLA-OFT uses an L_1 regression objective (Kim et al., 2025). These methods map VLM features directly to continuous actions without an intervening vocabulary. A second family predicts discrete action tokens that must be detokenized before execution; recent examples include VQ-VLA (Wang et al., 2025), FASTer (Liu

et al., 2026b), and G0.5 (Galaxea Team, 2026). This token-based formulation allows action prediction to retain the VLM's native autoregressive next-token interface. The increased adoption of action tokenization for robot policies raises the question that we aim to address in this work: how to best train the action tokenizer that determines the VLM prediction targets.

Discrete Representation Learning and Tokenization Discrete representation learning maps continuous features to tokens using vector quantization (VQ) (van den Oord et al., 2017), residual VQ (RVQ) (Lee et al., 2022), normalized RVQ (NRVQ) (Yu et al., 2022; Kumar et al., 2023), finite scalar quantization (FSQ) (Mentzer et al., 2024), or reconstruction-conditioned codebooks such as QinCo (Huijben et al., 2024; Vallaeys et al., 2025). These methods are used by tokenizers for vision (van den Oord et al., 2017; Yu et al., 2022; Qiu et al., 2025) and audio (Zeghidour et al., 2022).

For robot control, tokenization defines the prediction targets for the downstream policy. Analytical representations include per-dimension binning (Brohan et al., 2023), FAST frequency-domain codes (Pertsch et al., 2025), and BEAST B-spline coefficients (Zhou et al., 2025). Learned methods include VQ-BeT (Lee et al., 2024) and QueST (Mete et al., 2024) for discrete skills, OAT for prefix decodability (Liu et al., 2026a), and VQ-VLA (Wang et al., 2025) and FASTer (Liu et al., 2026b) for autoregressive VLAs. Recent methods further incorporate vocabulary structure (Dong et al., 2026), robot state (Jiang et al., 2026), or multimodal features (Kang et al., 2026). While such auxiliary signals can improve alignment on source tasks, they tie the tokenizer to task- or observation-specific supervision and may require realignment under distribution changes. To avoid this, we instead train tokenizers on action chunks alone, treating auxiliary supervision as complementary. Unlike prior systems that tune reconstruction, bitrate, and temporal horizons individually, we jointly analyze these factors and policy-generated sequences across a controlled tokenizer design space.

3 PRELIMINARIES

Action-token imitation learning first trains and freezes a tokenizer on action chunks, then trains a policy to predict the tokens that correspond to demonstration trajectory actions. For an H-step action chunk $A \in \mathbb{R}^{H \times d_a}$ with action dimension d_a, a learned tokenizer consists of an encoder E, a quantizer $Q = (q, e)$, and a decoder D:

$$\overbrace{A \xrightarrow{E} z_e \xrightarrow{q} C}^{\text{tokenization: } C=\mathcal{T}(A):=q(E(A))} \overbrace{\xrightarrow{e} z_q \xrightarrow{D} \widehat{A}}^{\text{detokenization: } \widehat{A}=\mathcal{D}(C):=D(e(C))}. \tag{1}$$

Here, q maps encoder embeddings to the token sequence C, while e maps C back to quantized embeddings. Across 22 configurations, we pair transformer (TF) (Liu et al., 2026a), Conv2D (C2) (Wang et al., 2025), and convolution–transformer (CT) backbones (Liu et al., 2026b) with VQ (van den Oord et al., 2017), RVQ (Lee et al., 2022), NRVQ (Yu et al., 2022; Kumar et al., 2023), FSQ (Mentzer et al., 2024), and QinCo (Huijben et al., 2024), using shared (S) or level-specific (L) codebooks where applicable. Appendix B.1 defines the complete tokenizer configurations.

Using a frozen tokenizer, each demonstration chunk yields a length-L target token sequence $C = \mathcal{T}(A) = (c_1, \ldots, c_L)$. Given observation O, the policy predicts a sequence $\widehat{C}$ autoregressively according to $p_\psi(C \mid O)$ and detokenizes its prediction as $\widehat{A}^\pi = \mathcal{D}(\widehat{C})$. We train the policy with the teacher-forced negative log-likelihood (NLL)

$$\mathcal{L}_{\text{AR}} := \mathbb{E}_{(O,C)}[-\log p_\psi(C \mid O)] = \mathbb{E}_{(O,C)}\left[\sum_{i=1}^{L} \ell_{\text{NLL},i}\right], \ \ell_{\text{NLL},i} := -\log p_\psi(c_i \mid c_{<i}, O). \tag{2}$$

The robot executes the first $H_{\text{exec}} \leq H$ actions before replanning; the policy and rollout settings remain fixed across tokenizer configurations.

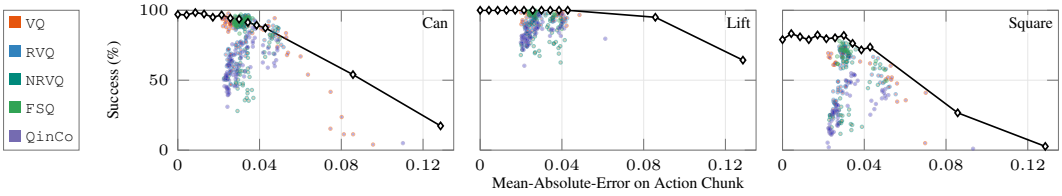


Figure 1: Expected validation reconstruction error $\mathcal{E}_{\text{rec}}$ and FM action-noise tolerance on Robomimic. Each colored point shows a tokenizer's error $\mathcal{E}_{\text{rec}}$ and downstream rollout success. Black curves indicate the performance of an FM policy with an increasing amount of pose noise, revealing task-dependent action-error tolerance.

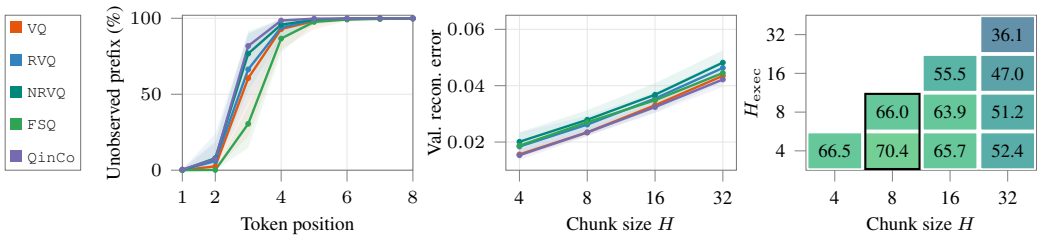
4 What Matters for Action Tokenization?

Understanding how action tokenization affects downstream performance is challenging, since it couples reconstruction, nominal bitrate, action horizon, and how an induced policy generates token sequences. In this section, we investigate these aspects empirically and theoretically.

4.1 Low Reconstruction Error Is Necessary but not Sufficient

A standard reconstruction metric for an expert action chunk A and its reconstruction $\widehat{A}$ is the mean absolute error (MAE) $\ell(A, \widehat{A}) = \frac{1}{Hd_a}\|\widehat{A} - A\|_1$.[1] We estimate the expected reconstruction error $\mathcal{E}_{\text{rec}} = \mathbb{E}[\ell(A, \mathcal{D}(\mathcal{T}(A)))]$ on validation chunks and examine its relationship with downstream rollout success on three Robomimic tasks (Figure 1). High action MAE consistently results in poor policy performance, while low MAE allows for good performance only some of the time: tokenizers with similar low reconstruction error can induce substantially different downstream performance. The inability to rank tokenizers based on reconstruction error is especially clear within the RVQ, NRVQ, and QinCo subset: the Spearman correlations between reconstruction error and rollout success are non-trivially *positive* on Can ($\rho = 0.45$), Lift ($\rho = 0.35$), and Square ($\rho = 0.50$), respectively (Appendix C.1). Thus, once sufficient fidelity is reached, lower reconstruction error is not a reliable indicator of better rollout performance.

To calibrate when action error becomes consequential, we inject controlled noise into action chunks predicted by an observation-conditioned flow-matching (FM) policy, which serves as an expert proxy (Braun et al., 2024). The black curves in Figure 1 show rollout success of the FM policy under increasing action perturbations, with the x-axis reporting the induced action-chunk MAE. The tolerance is task-dependent: Can and Square begin to degrade at relatively small errors, while Lift remains robust to much larger perturbations. This supports viewing low reconstruction error as a task-dependent necessary constraint rather than a metric that can reliably rank tokenizers once they reach a sufficient level of fidelity. We provide more details in Appendix C.1, including the results of a direct isolation experiment in which FM-predicted action chunks are tokenized, reconstructed, and executed, yielding the same conclusion that low reconstruction error is necessary but not sufficient.



(a) Unobserved-prefix rate. (b) Reconstruction versus H. (c) Rollout success (%).

Figure 2: (a) Rollout unobserved-prefix rate and (b) validation reconstruction error versus chunk size, where lines are quantizer-family means and shaded bands are min–max ranges over tokenizer designs; (c) success (%) across horizon pairs on Robomimic. Plots in (a) and (b) share colors.

[1] ℓ_2 is another standard reconstruction objective.

4.2 Nominal Bitrate and Action Chunking

A tokenizer with L levels and K tokens per level has a nominal bitrate (i.e., maximum representational capacity) of $L \log_2 K$ bits. Increasing this capacity generally reduces reconstruction error, but Section 4.1 shows that beyond a sufficient level of reconstruction accuracy, further reductions in error do not imply better downstream control (Figure 11; Appendix C.2). We therefore treat nominal bitrate as a representational capacity constraint, rather than simply making it as large as possible. Unless stated otherwise, we use $K = 128$ and $L = 8$ for our main experiments.

Action chunking adds two design choices for the policy: the chunk size H (i.e., the policy prediction horizon), and the execution horizon H_{exec} (i.e., the open-loop execution length before replanning). Under stability assumptions, open-loop execution can mitigate compounding error, but its benefit diminishes beyond a modest horizon (Zhang et al., 2025). We see the same qualitative behavior experimentally, with moderate horizons providing the best rollout success (Figure 2(c)). The degradation at large H is severe for autoregressive policies, as longer chunks increase reconstruction error (Figure 2(b)) above the level necessary for the task (approximately 0.04 on Robomimic). Increasing K or L could recover some of this lost fidelity, but it also changes the token space and therefore the downstream prediction problem; higher capacity does not imply that longer chunks become easier to learn or better for control (Appendix C.2). Continuous FM policies show the same performance trend in H_{exec} (Appendix C.3), so this temporal trade-off is not specific to tokenization. Thus, simply increasing bitrate cannot eliminate the underlying temporal trade-off. Per Figure 2(c), we primarily use $H = 8$ with $H_{\text{exec}} = 4$ for high success and $H_{\text{exec}} = 8$ for fast inference.

4.3 Beyond Empirical Training Support: A One-Step Error Analysis

While imitation learning incentivizes a policy to reproduce token sequences encoded from the demonstrations, rolling out the policy often produces token sequences that are absent from the set of encoded sequences from the training demonstrations. Figure 2(a) measures the fraction of rollout-generated prefixes that never appear in the encoded demonstrations. Leaving the empirical training support does not necessarily imply control failure, since unseen token sequences may still decode to valid actions (Appendix C.4).

Typically, tokenizers are trained to minimize reconstruction error relative to expert demonstrations, while autoregressive policies are trained to predict the corresponding demonstrated tokens. Neither objective explicitly controls the action-space error when the policy predicts a token sequence that is different from the encoded target. This motivates the following analysis:

Result 1: (informal; Proposition C.1)

There are three types of one-step tokenizer-policy errors. When the policy predicts the encoded target action (**Match**), the error reduces to $\mathcal{E}_{\text{rec}}$ and is directly optimized by the tokenizer reconstruction objective. When it predicts the encoding of another action close to the expert action for the same observation (**Benign Mismatch**), the error is determined by action similarity in addition to tokenizer reconstruction quality. For any other prediction (**Uncontrolled Mismatch**), the action error can be arbitrarily large relative to the reconstruction objective. Teacher forcing encourages exact matches but does not bound the severity of the action-space mismatches.

This distinguishes two properties of token mismatches: how often token mismatches occur and how harmful they are after decoding. Policy training mainly governs mismatch probability, whereas the tokenizer intrinsically shapes both the probability and severity of token mismatches. We make this decomposition formal in Appendix C.5.

5 Learning Robust and Predictable Action Tokenizers

The analysis in Section 4.3 motivates two complementary tokenizer-only interventions in ProAct: *Token Balance Regularization* for target predictability and *Token Perturbation* for mismatch robustness. ProAct applies both during tokenizer training without access to the downstream policy, images, or proprioceptive observations, yet consistently improves downstream performance.

5.1 Token Balance Regularization

Token Balance Regularization (TBR) seeks to **reduce the occurrence of mismatches** by making the tokenizer's discrete targets easier for a downstream policy to predict. When the policy predicts the demonstrated token sequence, the decoded action error is controlled by the optimized tokenizer reconstruction error. However, the encoded target may not be uniquely determined by the observation: demonstrations with similar observations can correspond to different valid token sequences. Under the tokenizer-induced distribution $P(C \mid O)$, a policy loss $\mathcal{L}_{\mathrm{AR}}$ (Equation 2) is lower-bounded by

$$\mathcal{L}_{\mathrm{AR}} = \mathbb{H}(C \mid O) + \mathbb{E}_O[\mathrm{KL}(P(\cdot \mid O) \,\|\, p_\psi(\cdot \mid O))] \geq \mathbb{H}(C \mid O). \tag{3}$$

$\mathbb{H}(C \mid O)$ is the *irreducible policy loss*: a tokenizer-dependent loss component in $\mathcal{L}_{\mathrm{AR}}$ that remains even when an unrestricted policy matches the conditional distribution of token targets.

A tokenizer partitions the action space into regions, with all actions in the same region assigned to the same token; we refer to the interfaces between different regions as *token boundaries*. The *token margin* of an action is the distance to the nearest boundary. Intuitively, given an observation, a larger margin makes nearby actions more likely to stay within the same token region, and avoids assigning them to different tokens. As a result, nearby actions are more likely to share the same token target. We show analytically that in the case of a linear encoder model, balanced feature learning (BFL) can reduce mismatches by inducing larger token margins, as summarized in the following result.

Result 2: (informal; Proposition D.7 and Corollary D.10)

Covariance-aware preconditioning, a type of balanced feature learning, reduces disparities in learning rates across different feature directions. Assuming a linear encoder, this disparity reduction translates directly into larger token margins, which lowers the irreducible policy loss and makes it easier for a downstream policy to predict the right tokens during training. This is because the preconditioning reduces the encoder's sensitivity to perturbations in action-space.

Therefore, we design TBR to promote more balanced feature learning (BFL) across different directions during tokenizer training. TBR relates to methods that regulate feature scales or correlations, such as batch normalization (Ioffe & Szegedy, 2015) and whitening (Huang et al., 2018). We instantiate TBR with a recent BFL method using covariance-aware preconditioning (Martens & Grosse, 2015; Zhang et al., 2026) for optimization updates without changing the training objective. Appendices D.1 and D.2 provide the details of our analysis and implementation.

5.2 Token Perturbation

Token Perturbation (TP) aims to **constrain the severity of token mismatches** by training the tokenizer to reconstruct an action from perturbed versions of its encoded token sequence. Given an action A with encoding $C = \mathcal{T}(A)$, we sample a perturbed sequence $C' \sim \nu(\cdot \mid C)$ and reconstruct the original action using the same training objective $\ell(A, \mathcal{D}(C'))$. We implement this with a straight-through estimator to train the entire tokenizer, so the forward pass uses the perturbed token embedding while gradients still flow through the original encoder latent. Here, $\nu(\cdot \mid C)$ denotes the conditional perturbation distribution. One simple perturbation rule randomly selects token positions and replaces them with nearby quantizer candidates in latent space. Reconstructing from token perturbation encourages robustness to token mismatches. Despite being policy-agnostic proxies rather than actual policy errors, these perturbations substantially improve downstream rollout success. We additionally study spectral-metric requantization for perturbation sampling and MaxUp (Gong et al., 2020); Appendix D.3 provides details and comparisons.

6 Experiments

6.1 Experimental Setting

We evaluate a diverse set of 22 tokenizer configurations, named ⟨`backbone`⟩-⟨`quantizer`⟩-⟨`S`/`L`⟩ (Section 3, Appendix B.1), while holding the downstream autoregressive policy configuration fixed. Our simulation study spans three manipulation benchmarks (Section 6.2)—

Table 1: Simulation success rates (%) at $H_{\text{exec}} = 4 / 8$, averaged within each benchmark. **Bold** denotes better downstream performance for the same execution horizon.

Tokenizer	RoboMimic		LIBERO		RoboTwin	
	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct
TF-VQ-S	73.4 / 80.2	**86.1 / 86.2**	50.6 / 45.4	**56.6 / 53.2**	**73.3 / 70.0**	62.7 / 64.0
TF-VQ-L	75.9 / 71.9	**77.6 / 72.7**	50.0 / 49.4	**58.6 / 52.7**	32.0 / 41.3	**58.7 / 70.0**
TF-RVQ-S	79.2 / 78.4	**84.3 / 86.2**	68.1 / 64.5	**72.1 / 69.5**	**72.0 / 69.3**	64.0 / **69.3**
TF-RVQ-L	59.9 / 55.6	**73.1 / 68.3**	57.5 / 51.8	**67.8 / 60.2**	23.3 / 22.7	**42.7 / 38.7**
CT-RVQ-S	**81.6** / 79.6	80.9 / **85.1**	38.9 / 46.3	**70.2 / 68.2**	61.3 / 64.0	**67.3 / 67.3**
CT-RVQ-L	62.0 / 59.2	**66.9 / 64.8**	48.6 / 46.1	**65.8 / 58.9**	17.3 / 22.7	**33.3 / 34.0**
C2-RVQ-S	**85.1** / 83.2	82.0 / **83.8**	**70.9 / 69.3**	70.1 / 68.0	**48.0 / 54.7**	23.3 / 22.0
C2-RVQ-L	59.2 / 57.9	**65.9 / 61.4**	54.4 / 50.0	**64.7 / 58.4**	**30.7** / 22.7	25.3 / **30.7**
TF-NRVQ-S	80.0 / 80.7	**84.3 / 84.3**	36.2 / 38.7	**41.5 / 47.5**	**62.7** / 56.7	61.3 / **58.0**
TF-NRVQ-L	56.6 / 48.3	**60.6 / 57.1**	55.2 / 49.0	**61.7 / 53.4**	30.0 / 36.0	**41.3 / 39.3**
CT-NRVQ-S	0.0 / 0.0	**81.1 / 86.6**	0.0 / 0.0	**56.7 / 63.7**	0.0 / 0.0	**48.0 / 68.0**
CT-NRVQ-L	0.0 / 0.0	**61.7 / 54.0**	0.0 / 0.0	**52.9 / 44.7**	0.0 / 0.0	**42.0 / 32.0**
C2-NRVQ-S	**80.8 / 82.9**	78.4 / **82.9**	**73.7 / 71.0**	70.7 / 67.5	**56.0** / 63.3	55.3 / **66.0**
C2-NRVQ-L	56.3 / 48.0	**61.4 / 52.0**	54.0 / 46.3	**54.9 / 49.5**	40.7 / **38.7**	**43.3** / 37.3
TF-FSQ-S	74.7 / 74.8	**80.2 / 80.8**	30.7 / 32.2	**61.0 / 58.4**	68.7 / 69.3	**72.7 / 78.0**
TF-FSQ-L	77.7 / 74.0	**81.4 / 83.4**	15.8 / 16.7	**60.4 / 57.1**	**67.3** / 68.7	58.7 / **74.0**
TF-QINCO-S	48.4 / 40.4	**75.8 / 73.8**	**44.9** / 34.4	43.9 / **38.9**	14.7 / 14.0	**52.7 / 48.7**
TF-QINCO-L	44.9 / 32.6	**74.0 / 72.1**	35.2 / 23.8	**37.5 / 33.8**	17.3 / 18.0	**46.0 / 46.0**
CT-QINCO-S	58.3 / 46.8	**70.9 / 63.7**	20.0 / 19.0	**35.4 / 33.1**	18.7 / 19.3	**51.3 / 49.3**
CT-QINCO-L	52.1 / 42.9	**71.2 / 64.1**	28.4 / 20.6	**42.5 / 39.5**	23.3 / 19.3	**47.3 / 44.0**
C2-QINCO-S	50.3 / 41.9	**73.7 / 71.2**	7.3 / 9.4	**38.0 / 35.9**	42.7 / 27.3	**56.7 / 47.3**
C2-QINCO-L	50.6 / 40.0	**60.8 / 52.3**	25.4 / 20.2	**63.6 / 53.6**	20.7 / 15.3	**25.3 / 24.7**

Robomimic (Mandlekar et al., 2022), LIBERO (Liu et al., 2023) and RoboTwin (Mu et al., 2025)—covering both single-arm and bimanual manipulation tasks. Baseline tokenizers are trained with the Reconstruction Only (R.O.) objective, whereas ProAct augments tokenizer training with Token Balance Regularization and Token Perturbation. Additionally, we evaluate our method with a VLA in LIBERO (Section 6.2) and in real-world manipulation domains (Section 6.4). Appendix B.3 and E.5 document the detailed setup, hyperparameters, and evaluation protocol.

6.2 Simulation Results

Table 1 evaluates ProAct across the tokenizer design space. Across Robomimic, LIBERO, and RoboTwin, ProAct improves average success by 10.4 and 11.3 points at $H_{\text{exec}} = 4$ and $H_{\text{exec}} = 8$, respectively[2]. The gains are particularly pronounced for level-specific (L) configurations (+12.3/+13.1 points) and QinCo variants (+20.2/+22.6 points), spanning substantially different quantization designs. Notably, these configurations often achieve lower $\mathcal{E}_{\text{rec}}$ under R.O. (Table 16) despite substantially worse rollout performance, reinforcing that better reconstruction does not necessarily translate to better downstream success. Aggregate improvements extend across all three benchmarks using a common set of hyperparameters for most tokenizer configurations within each benchmark (Table 13; Table 14). Appendix E.1 provides per-task results and additional data-setting and checkpoint comparisons.

Table 2: VLA success rates (%) on LIBERO at $H_{\text{exec}} = 8$.

Tokenizer	R.O.	ProAct
TF-VQ-S	15.5 ± 1.6	$\mathbf{45.4 \pm 2.2}$
TF-VQ-L	25.8 ± 1.9	$\mathbf{48.9 \pm 2.2}$
TF-RVQ-S	$\mathbf{53.8 \pm 2.2}$	48.1 ± 2.2
TF-RVQ-L	39.2 ± 2.1	$\mathbf{53.1 \pm 2.2}$
CT-RVQ-L	22.7 ± 1.8	$\mathbf{29.2 \pm 2.0}$
TF-FSQ-S	11.5 ± 1.4	$\mathbf{48.8 \pm 2.2}$
TF-FSQ-L	32.2 ± 2.0	$\mathbf{55.1 \pm 2.2}$
C2-QINCO-L	9.2 ± 1.3	$\mathbf{55.9 \pm 2.2}$

Table 3: Ablation on Robomimic. We report success rates (%) at $H_{\text{exec}} = 4 / 8$. We show the averaged success by *Task* and *Codebook sharing*.

Subset	R.O.	w/o TBR	w/o TP	ProAct
Task				
Can	76.1 / 70.1	83.5 / 79.2	85.5 / 77.7	**87.0 / 83.7**
Lift	78.9 / 77.1	84.1 / 81.6	**93.5 / 92.9**	84.9 / 84.8
Square	41.1 / 35.7	48.5 / 43.0	42.3 / 37.0	**51.6 / 48.5**
Codebook sharing				
Shared (S)	71.2 / 68.9	78.3 / 76.0	**80.6** / 77.5	79.7 / **79.8**
Level-Specific (L)	59.5 / 53.0	65.8 / 59.8	67.0 / 60.9	**69.3 / 64.8**
Overall	65.4 / 61.0	72.0 / 67.9	73.8 / 69.2	**74.5 / 72.3**

[2]Averages exclude CT-NRVQ-S/L, whose 0% baseline R.O. success rate is an outlier.

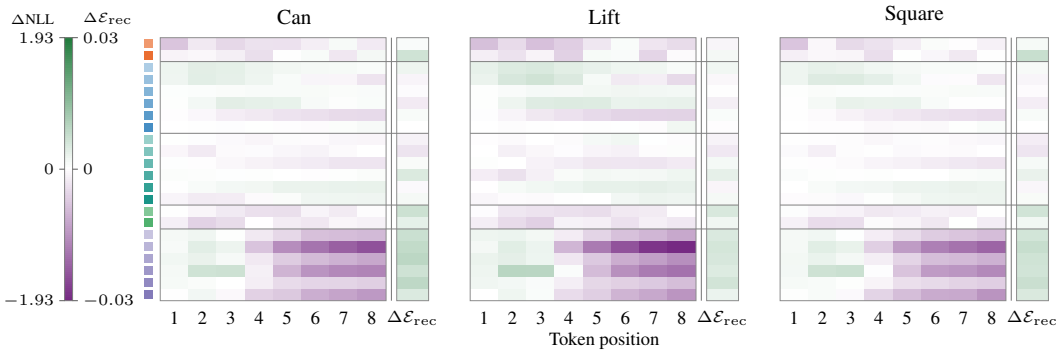


Figure 3: Effect of Token Balance Regularization (TBR). Δ_{NLL} is the difference in normalized NLL between ProAct and w/o TBR at positions 1–8. $\Delta\mathcal{E}_{\text{rec}}$ denotes the corresponding difference in validation reconstruction MAE. Each row is labeled with the same tokenizer color indicator as in Table 1. Purple denotes decreases and green increases.

To test whether these gains transfer to a pretrained vision-language backbone, we fine-tune SmolVLM2-256M (Marafioti et al., 2025) with an autoregressive action expert on LIBERO, with implementation details provided in Appendix Table 12. ProAct improves success rate in 7 of 8 tokenizer pairs, raising the average from 26.2% to 48.1% (Table 2).

6.3 Ablations and Error Analyses

We ablate the two proposed tokenizer-only interventions on Robomimic across the tokenizer design space (Table 3).[3] Both *Token Balance Regularization* (w/o TP) and *Token Perturbation* (w/o TBR) independently improve rollout success, while including both results in the best performance. Appendix E.2 reports the complete task-level results for all tokenizer configurations.

Motivated by the one-step error analysis in Section 4.3, we next examine the specific effect targeted by each intervention: whether TBR makes the resulting token sequences easier for the policy to predict, and whether TP reduces the action-space severity of token mismatches.

Token predictability across positions. Token Balance Regularization is designed to make the tokenizer's discrete targets easier for a downstream policy to predict. To test this effect directly, we measure the teacher-forced per-position NLL (Eqn. 2), normalized by $\log K$, comparing w/o TBR with ProAct (Figure 3). Across tokenizer variants, TBR reduces mean normalized NLL by 12.4% on Can, 10.0% on Lift, and 11.3% on Square, and reduces the within-tokenizer standard deviation of NLL across positions by 18.9–21.4%. These gains are accompanied by a 7.2% average increase in validation MAE relative to w/o TBR across tokenizers and tasks. Figure 3 shows both metrics for each tokenizer. The reduction is most pronounced at later autoregressive positions, yielding a more balanced prediction difficulty across the sequence, as seen in the smaller cross-position NLL standard deviation. Together, these results show that TBR makes the induced token targets easier to predict and more uniform across positions, with a reconstruction trade-off. Appendix E.3 reports aggregate diagnostics and per-tokenizer NLL curves.

Decoded-action robustness. Token Perturbation aims to reduce the action-space impact of token mismatches. For each validation observation with expert action chunk A, we sample M policy-predicted token sequences and decode them into actions $\widehat{A}^{\pi,(1)}, \ldots, \widehat{A}^{\pi,(M)}$. Among samples whose token sequence differs from the expert encoding, we record the Max-M mismatch error $\ell_\pi^{\max}(M) = \max_{i\in[1,M]} \ell(A, \widehat{A}^{\pi,(i)})$. We then define $\Delta_{\text{mismatch}}(M) = \ell^{\max}_{\text{ProAct}}(M) - \ell^{\max}_{\text{w/o TP}}(M)$ and report its average over paired tokenizer variants in Figure 4. As a reference, we also report the paired difference in tokenizer reconstruction error, $\Delta\mathcal{E}_{\text{rec}}$, between ProAct and w/o TP. This paired ablation isolates the effect of TP. As M increases, $\Delta_{\text{mismatch}}(M)$ becomes increasingly negative, showing that TP reduces the largest decoded errors exposed by broader policy sampling, despite a small increase in tokenizer reconstruction error. Since policy-generated sequences rarely exactly match the expert encoding (Appendix C.4), this directly supports the intended effect of TP: reduc-

[3]The aggregate excludes `CT-NRVQ-S/L`, whose 0% baseline R.O. success rate is an outlier.

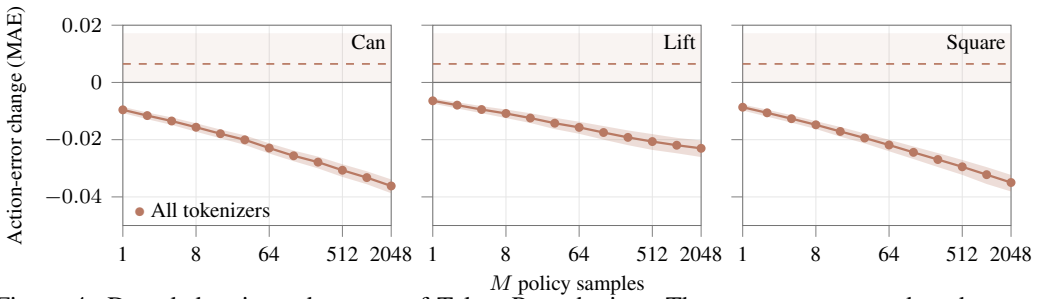


Figure 4: Decoded-action robustness of Token Perturbation. The curves correspond to the mean paired change in Max-M mismatch error between ProAct and w/o TP as the number of sampled policy sequences M increases. Negative values indicate that TP reduces decoded action error; shaded bands denote the 95% bootstrap confidence intervals. The dashed line shows the corresponding change in tokenizer reconstruction error.

ing the action-space severity of token mismatches. Appendix E.4 reports the R.O. versus w/o TBR comparison and discusses the effect of different policy token sampling strategies.

6.4 Real-Robot Experiments

We further evaluate on two real-world UR5 manipulation tasks (Figure 5): *Cube Stacking*, which requires placing a blue triangular block on an orange rectangular block amid cluttered distractors, and *Cup Hanging*, which requires grasping a cup and hanging it onto a cup tree with accurate positioning and alignment. Across both tasks, ProAct improves success for all evaluated tokenizer configurations. The gains are particularly pronounced for the level-specific variants on Cube Stacking, consistent with our simulation results, while Cup Hanging also shows substantial improvements across tokenizers. Table 4 reports the per-task results, and Appendix E.5 provides the full setup and evaluation protocol.

Tokenizer	Cube Stacking		Cup hanging	
	R.O.	ProAct	R.O.	ProAct
TF-FSQ-S	14/30 (46.7)	**17/30 (56.7)**	10/30 (33.3)	**20/30 (66.7)**
TF-FSQ-L	4/30 (13.3)	**18/30 (60.0)**	3/30 (10.0)	**18/30 (60.0)**
TF-RVQ-S	8/30 (26.7)	**19/30 (63.3)**	10/30 (33.3)	**21/30 (70.0)**
TF-RVQ-L	1/30 (3.3)	**19/30 (63.3)**	3/30 (10.0)	**14/30 (46.7)**
CT-RVQ-L	1/30 (3.3)	**12/30 (40.0)**	2/30 (6.7)	**8/30 (26.7)**

Table 4: UR5 results: successes/trials (success %).

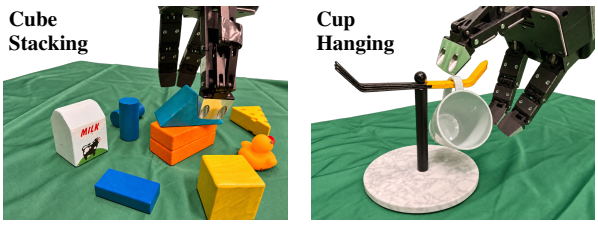


Figure 5: The two real-robot manipulation tasks.

7 Conclusions

Effective action tokenization requires more than accurate reconstruction. Through a controlled study across a broad tokenizer design space, we show that poor reconstruction limits downstream performance, but that among sufficiently accurate tokenizers, rollout success also varies with chunk size, token predictability, and mismatch severity. Additionally, we provide a one-step error analysis that identifies a gap in standard tokenizer and policy objectives: they encourage accurate reconstruction and target prediction, but do not control the severity of token mismatches. To address this gap, we introduce ProAct, which combines *Token Balance Regularization* and *Token Perturbation*, two policy-independent tokenizer training methods that target predictability and robustness, respectively. ProAct improves downstream policy performance across Robomimic, LIBERO, RoboTwin, and real-robot manipulation.

Our experiments focus on autoregressive imitation-learning policies and a limited set of benchmarks, robot embodiments, and training seeds. An important next step is to test whether these findings extend to reinforcement learning and to broader task and embodiment distributions.

AI USE STATEMENT

We used large language models and AI tools to assist with drafting and polishing manuscript sections and detailed results, analyses in the appendices, retrieving and identifying relevant literature in tokenization and robot learning, research experiment implementation, and proving and validating the mathematical claims. This assistance included developing explanatory hypotheses, interpreting experimental results, preparing analysis and visualization scripts, and formatting tables and figures. We did not use AI tools to generate synthetic datasets. We are responsible for the final content of this work, including its text, claims, code, and other artifacts produced with AI assistance.

ETHICS STATEMENT

This work studies action tokenization for robot manipulation. Erroneous robot actions can cause physical harm or damage, and improved performance on the evaluated benchmarks does not imply that a policy is safe to deploy. Deployment requires appropriate human supervision, physical and software safeguards, and validation under the intended operating conditions, including tasks, environments, and embodiments beyond those studied here. The reported improvements should therefore be interpreted within the scope of the evaluated settings rather than as safety guarantees.

REPRODUCIBILITY STATEMENT

We document the tokenizer configurations in Appendix B.1, rollout evaluation and uncertainty estimation in Appendix B.2, and training hyperparameters, checkpoint conventions, and sources of randomness in Appendix B.3. Appendices D.2 and D.3 describe the TBR and TP implementations and their hyperparameters. Detailed simulation results and ablations are reported in Appendices E.1 and E.2, while Appendix E.5 documents the real-robot setup and evaluation protocol.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5144–5151, 2024.

Anthony Brohan et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Zibin Dong, Yicheng Liu, Shiduo Zhang, Baijun Ye, Yifu Yuan, Fei Ni, Jingjing Gong, Xipeng Qiu, Hang Zhao, Yinchuan Li, and Jianye Hao. ActionCodec: What makes for good action tokenizers. *arXiv preprint arXiv:2602.15397*, 2026.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 158–168, 2022.

Galaxea Team. Galaxea G0.5 technical report. Technical Report, 2026.

Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. MaxUp: A simple way to improve generalization of neural network training. *arXiv preprint arXiv:2002.09024*, 2020.

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.

Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 791–800, 2018.

Iris A. M. Huijben, Matthijs Douze, Matthew J. Muckley, Ruud J. G. van Sloun, and Jakob Verbeek. Residual quantization with implicit neural codebooks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 20682–20699, 2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 448–456, 2015.

Tengyue Jiang, Chunpu Xu, Jiayue Kang, and Yao Mu. SA-VLA: State-aware tokenizer for improving vision-language-action models' performance. *arXiv preprint arXiv:2606.30113*, 2026.

Miracle Kang, Lights Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Sylas Chen, Shawn Qin, Yinan Zheng, Jinliang Zheng, Hao Wang, Xianyuan Zhan, and Hang Su. X-Tokenizer: A multimodal action tokenizer for vision-language-action pretraining. *arXiv preprint arXiv:2606.14752*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Rithesh Kumar, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar. High-fidelity audio compression with improved rvqgan. *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 27980–27993, 2023.

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why does action chunking improve behavioral cloning performance in robotic control?, 2026.

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

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 44776–44791, 2023.

Chaoqi Liu, Yue Zhao, Haonan Chen, Xiaoshen Han, Jiawei Gao, Ehsan Adeli, and Yilun Du. Ordered action tokens for visuomotor policy learning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2026a.

Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, Liangtao Zheng, Tao Jiang, Jingjing Gong, Xipeng Qiu, and Hang Zhao. FASTer: Toward efficient autoregressive vision language action modeling via neural action tokenization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026b.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 1678–1690, 2022.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. SmolVLM: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 37, pp. 2408–2417, 2015.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. QueST: Self-supervised skill abstractions for learning continuous control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Kai Qiu, Xiang Li, Jason Kuen, Hao Chen, Xiaohao Xu, Jiuxiang Gu, Yinyi Luo, Bhiksha Raj, Zhe Lin, and Marios Savvides. Robust latent matters: Boosting image generation with sampling error synthesis. *arXiv preprint arXiv:2503.08354*, 2025.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 627–635, 2011.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Théophane Vallaeys, Matthew Muckley, Jakob Verbeek, and Matthijs Douze. QINCo2: Vector compression and search with improved implicit neural codebooks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 11089–11099, 2025.

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 VQ-GAN. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

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

Thomas T. Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and exploratory data collection yield exponential improvements in behavior cloning for continuous control. *arXiv preprint arXiv:2507.09061*, 2025.

Thomas T. Zhang, Alok Shah, Yifei Zhang, Vincent Zhang, Nikolai Matni, and Max Simchowitz. Double preconditioning (DoPr): Optimization for test-time performance, not validation loss. *arXiv preprint arXiv:2606.06418*, 2026.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, Ömer Erdinc YagmOPTurlu, Nils Blank, Moritz Reuss, and Rudolf Lioutikov. BEAST: Efficient tokenization of B-splines encoded action sequences for imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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 *Proceedings of the Conference on Robot Learning (CoRL)*, pp. 2165–2183, 2023.

APPENDICES

A ANALYTICAL TOKENIZATION SCHEMES

Our controlled study focuses on learned action tokenizers rather than fixed analytical schemes. As a representative analytical baseline, we consider per-dimension binning (Bin). Under the same autoregressive policy training protocol used throughout our study, without representation-specific augmentations or decoding heuristics,[4] Bin reaches 67.7% / 68.6% success on Robomimic at $H_{\text{exec}} = 4 \,/\, 8$ (Table 6), and its reconstruction MAE is 0.002533.[5] Bin, however, spends 56 tokens on an $H = 8$ chunk, compared with 8 tokens for the learned tokenizers in our main experiments, and its success remains comparable with that of the learning-based tokenizers trained with the R.O. objective, while using twice the codebook size per level (Table 1). We therefore examine how much of the Bin code a policy actually needs. These experiments motivate our focus on learning-based tokenizers in the controlled study.

Table 5: Configuration of the Bin tokenizer.

Representation	Per-scalar quantization
Vocabulary / bins	256 bins per scalar
Normalization	Per-dim. q01/q99 to $[-1, 1]$
Quantization	Uniform over $[-1, 1]$
Sequence length	56
Learned parameters	None

Table 6: Robomimic success rates (%) of Bin and Prefix-Bin at $H_{\text{exec}} = 4 \,/\, 8$, using the checkpoint with the best teacher-forced validation accuracy and 10 seeds $\times$ 30 episodes per task.

Tokenizer	Tokens	Can	Lift	Square	Avg.
Bin	56	78.7 / 82.0	96.0 / 98.0	28.3 / 25.7	67.7 / 68.6
Prefix-Bin	8	53.7 / 27.3	83.7 / 55.3	15.7 / 1.7	51.0 / 28.1

Table 5 summarizes the Bin configuration. Bin independently quantizes each scalar action into 256 bins, producing a fixed 56-token sequence for an $H = 8$, $d_a = 7$ action chunk.

How much of the Bin code does the policy need? Bin represents an $H = 8$ action chunk as 56 tokens, one for each action dimension and timestep. If most of this sequence is redundant, a policy should not need to predict all of it. We construct *Prefix-Bin*: the policy predicts only the first eight Bin tokens, after which an empirical training-set lookup completes the corresponding 56-token sequence for decoding. Prefix-Bin therefore reduces the policy prediction horizon from 56 to 8 tokens while retaining essentially the same reconstruction fidelity as Bin, with a negligible MAE difference of approximately 3.29×10^{-7}.

This lookup is almost unambiguous on the training corpus: 179,621 distinct eight-token prefixes correspond to 179,622 distinct full Bin sequences. Predicting only these eight tokens, Prefix-Bin retains 51.0% success at $H_{\text{exec}} = 4$, about three quarters of Bin's, with a prediction horizon seven times shorter (Table 6). The gap widens at $H_{\text{exec}} = 8$ (28.1%): the policy's predicted prefixes almost never appear in the training table (10 of 116,977 at rollout), so the suffix is completed from the nearest table entry, and executing more of each chunk executes more of these retrieved actions.

Bin therefore works, but inefficiently: most of its tokens are determined by a short prefix, and a policy that predicts one seventh of them retains three quarters of its success at $H_{\text{exec}} = 4$. A learned tokenizer can instead spend a much shorter code on the structure that varies across action chunks, which is also where TBR and TP act. We therefore focus the controlled study on learned tokenizers.

[4] We intentionally exclude techniques such as Masked Action Augmentation (Goyal et al., 2025) as they may mitigate limitations of the underlying token representation and make it harder to isolate the effect of the tokenizer itself.

[5] This error is substantially smaller than the typical learned-tokenizer validation errors reported in Appendix C.1, Figure 6.

B Additional Preliminaries

B.1 Tokenizer Configuration Index

B.1.1 Tokenizer Backbones

The tokenizer backbone implements the encoder E and decoder D surrounding the discrete bottleneck. We use three variants that differ in how they organize and compress temporal and action-coordinate structure.

Transformer (TF). The encoder embeds each action timestep, adds temporal position embeddings, appends R learned register tokens, and processes the joint sequence with masked self-attention. Each register can attend to the complete action chunk and to preceding registers, and the final register states are projected to $z_e \in \mathbb{R}^{R \times d_z}$. The decoder uses one learned query per output timestep and cross-attends to the quantized registers in a single pass. We set $R = 1$ for residual quantizers and $R = L$ for the parallel VQ and FSQ families. This construction follows the register-token action-autoencoder pattern used in recent learned action tokenizers (Liu et al., 2026a; Dong et al., 2026).

Conv2D (C2). The encoder partitions the action coordinates into semantic groups and divides time into non-overlapping windows, producing a 4×4 or 4×8 patch grid depending on the action space. Convolutional residual blocks and grid self-attention operate at successive resolutions while two downsampling stages compress both grid axes into one global latent vector. The decoder mirrors this hierarchy with two-dimensional upsampling and unpatchifies the reconstructed grid back into an action chunk. Although this backbone includes lightweight self-attention within the grids, its defining structure is hierarchical two-dimensional compression over time and grouped action coordinates, related to the convolutional action-tokenizer design used by VQ-VLA (Wang et al., 2025).

Convolution–Transformer (CT). The hybrid encoder divides time into windows while preserving a separate grid column for every action coordinate, with additional embeddings identifying individual coordinates and semantic action groups. It interleaves convolutional residual blocks with two-dimensional self-attention while downsampling only the temporal axis from four rows to two and then one. Attention pooling across the remaining action-coordinate columns produces one global latent vector; the decoder reverses the temporal hierarchy to reconstruct the action chunk. This design combines the local hierarchy of convolution with attention over the full action grid, following the broad convolution–attention pattern explored by FASTer and QueST (Liu et al., 2026b; Mete et al., 2024).

The C2 and CT backbones emit one latent vector and are therefore paired with the residual quantizer families; TF also supports multiple register latents and is used with all five families.

B.1.2 Quantizer Families

The quantizer $Q = (q, e)$ maps encoder embeddings to discrete symbols through q and maps those symbols back to quantized embeddings through e. Let L be the number of quantization layers, $\ell \in [L]$ index a layer, and K be the number of possible token indices at each layer, with $k \in [K]$ indexing a candidate. The indexing map produces $C = q(z_e) = (c_1, \ldots, c_L) \in [K]^L$, and the embedding map produces $z_q = e(C)$. Thus each tokenizer represents one action chunk using a nominal $L \log_2 K$ bits. VQ and FSQ quantize L parallel encoder latents $z_e = (z_{e,1}, \ldots, z_{e,L})$, whereas RVQ, NRVQ, and QinCo generate L indices sequentially from one latent vector.

Vector quantization (VQ). At layer ℓ, VQ uses a learned codebook $\mathcal{E}_\ell = \{v_{\ell,k}\}_{k=1}^{K}$ and independently assigns latent $z_{e,\ell}$ to its nearest codeword (van den Oord et al., 2017):

$$c_\ell = \arg\min_{k \in [K]} \|z_{e,\ell} - v_{\ell,k}\|_2^2, \qquad z_{q,\ell} = v_{\ell,c_\ell}. \tag{4}$$

Thus, the indices are $C = (c_1, \ldots, c_L)$ and the quantized representation is $z_q = (z_{q,1}, \ldots, z_{q,L})$.

Residual vector quantization (RVQ). RVQ successively quantizes the error left by earlier layers (Lee et al., 2022; Zeghidour et al., 2022). At layer ℓ, let $\mathcal{E}_\ell = \{v_{\ell,k}\}_{k=1}^K$ denote the learned codebook. Starting from $r_0 = z_e$, RVQ applies nearest-neighbor lookup to each residual:

$$\begin{aligned} c_\ell &= \arg\min_{k\in[K]} \|r_{\ell-1} - v_{\ell,k}\|_2^2, \qquad r_\ell = r_{\ell-1} - v_{\ell,c_\ell}, \\ z_q &= \sum_{\ell=1}^{L} v_{\ell,c_\ell}. \end{aligned} \tag{5}$$

Thus, the indices $C = (c_1, \ldots, c_L)$ form an ordered coarse-to-fine sequence, and the quantized representation is $z_q = \sum_{\ell=1}^{L} v_{\ell,c_\ell}$.

Normalized residual vector quantization (NRVQ). NRVQ retains the residual hierarchy of RVQ but performs assignment using normalized code directions and scales the selected direction at each layer by a learned scalar s_ℓ (Yu et al., 2022; Kumar et al., 2023). With $r_0 = z_e$, it computes

$$\begin{aligned} \bar{v}_{\ell,k} &= \frac{v_{\ell,k}}{\|v_{\ell,k}\|_2 + \epsilon}, \qquad c_\ell = \arg\max_{k\in[K]} r_{\ell-1}^\mathsf{T} \bar{v}_{\ell,k}, \\ r_\ell &= r_{\ell-1} - s_\ell \bar{v}_{\ell,c_\ell}, \qquad z_q = \sum_{\ell=1}^{L} s_\ell \bar{v}_{\ell,c_\ell}. \end{aligned} \tag{6}$$

The lookup normalizes the candidate directions but not the incoming residual; normalizing the residual would not change the selected index because its norm is constant across candidates. The learned scalar s_ℓ controls the contribution of layer ℓ to the reconstructed latent. Thus, the indices are $C = (c_1, \ldots, c_L)$ and the quantized representation is $z_q = \sum_{\ell=1}^{L} s_\ell \bar{v}_{\ell,c_\ell}$.

Finite scalar quantization (FSQ). FSQ quantizes each of the L encoder latents using a fixed Cartesian grid (Mentzer et al., 2024). Let ϕ be a bijection from the grid points to $[K]$. At layer ℓ, Π_ℓ^{in} projects $z_{e,\ell}$ to d_f scalar coordinates, applies $\tanh$-based bounding, and rounds each coordinate to its prescribed finite level with a straight-through estimator. The token index and quantized latent are

$$c_\ell = \phi\big(\Pi_\ell^{\text{in}}(z_{e,\ell})\big), \qquad z_{q,\ell} = \Pi_\ell^{\text{out}}\big(\phi^{-1}(c_\ell)\big). \tag{7}$$

Here K is the product of the per-coordinate level counts, and Π_ℓ^{out} maps the selected grid point to the decoder embedding. Thus, the indices are $C = (c_1, \ldots, c_L)$ and the quantized representation is $z_q = (z_{q,1}, \ldots, z_{q,L})$.

Quantization with Implicit Neural Codebooks (QinCo). QinCo extends residual quantization by generating the codebook at each layer conditioned on the quantized representation accumulated from previous layers (Huijben et al., 2024; Vallaeys et al., 2025). Let z_e denote the encoder output, L the number of quantization layers, and K the number of candidate code vectors at each layer. We denote by $z_q^{(\ell)}$ the cumulative quantized representation after layer ℓ, with $z_q^{(0)} = 0$, and define the residual before layer ℓ as $r_{\ell-1} = z_e - z_q^{(\ell-1)}$.

The first layer uses a learned static codebook $\mathcal{E}_1 = \{v_{1,k}\}_{k=1}^K$, where $v_{1,k}$ denotes the k-th code vector. It selects the index $c_1 = \arg\min_{k\in[K]} \|z_e - v_{1,k}\|_2^2$ and initializes the quantized representation as $z_q^{(1)} = v_{1,c_1}$.

For each later layer $\ell \geq 2$, a learnable network $\mathcal{V}_\ell$ maps $z_q^{(\ell-1)}$ to a sample-dependent codebook, after which QinCo selects the code vector closest to the current residual:

$$\begin{aligned} \mathcal{V}_\ell\Big(z_q^{(\ell-1)}\Big) &= \Big\{ v_{\ell,k}\Big(z_q^{(\ell-1)}\Big) \Big\}_{k=1}^K, \\ c_\ell &= \arg\min_{k\in[K]} \Big\| r_{\ell-1} - v_{\ell,k}\Big(z_q^{(\ell-1)}\Big) \Big\|_2^2, \\ z_q^{(\ell)} &= z_q^{(\ell-1)} + v_{\ell,c_\ell}\Big(z_q^{(\ell-1)}\Big). \end{aligned} \tag{8}$$

Here, $v_{\ell,k}(z_q^{(\ell-1)})$ is the k-th code vector generated for layer ℓ under the current prefix $z_q^{(\ell-1)}$, and c_ℓ is the index selected at that layer. In our implementation, each conditional code vector is formed by adding a learned static anchor to the output of a linear generator conditioned on $z_q^{(\ell-1)}$. Therefore, unlike a static codebook, the same token index k can correspond to different code vectors for different prefixes. The resulting token sequence is $C = (c_1, \ldots, c_L)$, and the final quantized representation is $z_q = z_q^{(L)}$.

B.1.3 Shared and Level-Specific Quantization

Let Θ_ℓ denote the family-specific parameter block whose sharing status defines the quantizer variant at layer ℓ. A *shared* quantizer ties these parameters across layers, $\Theta_\ell = \Theta$, whereas a *level-specific* quantizer uses independently parameterized Θ_ℓ. We denote the two variants by the suffixes `S` and `L`, respectively. Both variants produce indices $C \in [K]^L$ and therefore have the same nominal bitrate; they differ only in which quantizer parameters are reused across layers.

VQ. For VQ, Θ_ℓ is the learned codebook $\mathcal{E}_\ell$. Shared VQ uses one codebook $\mathcal{E} = \{v_k\}_{k=1}^K$ for every parallel latent, so the same index k maps to v_k at every layer. Level-specific VQ instead learns an independent codebook $\mathcal{E}_\ell = \{v_{\ell,k}\}_{k=1}^K$ for each layer, so the vector associated with index k can vary with ℓ.

RVQ. For RVQ, Θ_ℓ contains the vectors used to quantize the residual at layer ℓ. Shared RVQ reuses one learned table $\mathcal{E} = \{v_k\}_{k=1}^K$ throughout the residual hierarchy and learns a scalar s_ℓ for each layer. Its assignment and residual update are

$$c_\ell = \arg\min_{k \in [K]} \|r_{\ell-1} - s_\ell v_k\|_2^2, \qquad r_\ell = r_{\ell-1} - s_\ell v_{c_\ell}. \tag{9}$$

Level-specific RVQ instead learns $\mathcal{E}_\ell = \{v_{\ell,k}\}_{k=1}^K$ independently at every layer and applies the RVQ recursion in Equation 5 directly with $v_{\ell,k}$.

NRVQ. For NRVQ, Θ_ℓ is the normalized direction table. Shared NRVQ reuses one table $\{\bar{v}_k\}_{k=1}^K$, selects c_ℓ using these shared directions, and subtracts $s_\ell \bar{v}_{c_\ell}$ from the residual. Level-specific NRVQ instead learns an independent table $\{\bar{v}_{\ell,k}\}_{k=1}^K$ at every layer, selects c_ℓ using the layer-specific directions, and subtracts $s_\ell \bar{v}_{\ell,c_\ell}$. Both variants retain an independent learned scale s_ℓ for each layer; only the normalized directions are shared or learned separately.

FSQ. For FSQ, the Cartesian grid is fixed, so Θ_ℓ consists only of the learned projection pair $(\Pi_\ell^{\text{in}}, \Pi_\ell^{\text{out}})$ surrounding the grid. Shared FSQ ties this pair across layers, $(\Pi_\ell^{\text{in}}, \Pi_\ell^{\text{out}}) = (\Pi^{\text{in}}, \Pi^{\text{out}})$ for all ℓ, whereas level-specific FSQ learns an independent projection pair for each parallel latent.

QinCo. For QinCo, Θ_ℓ is the learnable generator that maps the partial quantized representation to the conditional codebook in Equation 8. For each layer $\ell \geq 2$, let $a_{\ell,k}$ be the learned static anchor for index k. The shared variant ties one generator g across layers, whereas the level-specific variant learns an independent generator g_ℓ:

$$v_{\ell,k}\left(z_q^{(\ell-1)}\right) = \begin{cases} a_{\ell,k} + [g(z_q^{(\ell-1)})]_k, & \text{shared,} \\ a_{\ell,k} + [g_\ell(z_q^{(\ell-1)})]_k, & \text{level-specific.} \end{cases} \tag{10}$$

Both variants retain layer-specific anchors $\{a_{\ell,k}\}_{k=1}^K$, and the first-layer codebook $\mathcal{E}_1$ is unchanged by the sharing choice. Thus, `S` denotes sharing the prefix-to-codebook generator, not using identical realized codebooks at every layer.

Across all five families, the suffixes `S` and `L` therefore identify the family-specific parameter block that is shared or learned independently across layers.

Table 7 lists every tokenizer used in the controlled comparison. The structural identifier and color swatch are held fixed across all main-paper and appendix tables.

Table 7: The 22 tokenizer configurations evaluated in this study. Structural IDs encode the backbone (TF: Transformer; CT: convolution–Transformer; C2: Conv2D), quantizer, and quantization-parameter sharing (S: shared; L: level-specific). The color swatch redundantly indicates the quantizer family.

ID	Backbone	Quantizer	Parameter sharing
TF-VQ-S	Transformer	VQ	Shared
TF-VQ-L	Transformer		Level-specific
TF-RVQ-S	Transformer	RVQ	Shared
TF-RVQ-L	Transformer		Level-specific
CT-RVQ-S	Convolution–Transformer		Shared
CT-RVQ-L	Convolution–Transformer		Level-specific
C2-RVQ-S	Conv2D		Shared
C2-RVQ-L	Conv2D		Level-specific
TF-NRVQ-S	Transformer	NRVQ	Shared
TF-NRVQ-L	Transformer		Level-specific
CT-NRVQ-S	Convolution–Transformer		Shared
CT-NRVQ-L	Convolution–Transformer		Level-specific
C2-NRVQ-S	Conv2D		Shared
C2-NRVQ-L	Conv2D		Level-specific
TF-FSQ-S	Transformer	FSQ	Shared
TF-FSQ-L	Transformer		Level-specific
TF-QINCO-S	Transformer	QinCo	Shared
TF-QINCO-L	Transformer		Level-specific
CT-QINCO-S	Convolution–Transformer		Shared
CT-QINCO-L	Convolution–Transformer		Level-specific
C2-QINCO-S	Conv2D		Shared
C2-QINCO-L	Conv2D		Level-specific

B.2 Rollout Evaluation Protocol

Our primary results use greedy action-token prediction. For an evaluation unit with n binary rollout outcomes, let $y_i \in \{0, 1\}$ denote the outcome of episode i. The reported success rate is

$$\widehat{p} = \frac{1}{n} \sum_{i=1}^{n} y_i. \tag{11}$$

To quantify uncertainty, we pool the n binary episode outcomes and, for each bootstrap replicate $b \in \{1, \ldots, B\}$ with $B = 10{,}000$, draw n outcomes with replacement and compute the corresponding success rate $\widehat{p}^{*(b)}$. We estimate the bootstrap standard error and the 95% interval half-width as

$$\widehat{\mathrm{SE}}_{\mathrm{boot}} = \mathrm{sd}\left(\widehat{p}^{*(1)}, \ldots, \widehat{p}^{*(B)}\right), \qquad \Delta_{95} = 1.96\, \widehat{\mathrm{SE}}_{\mathrm{boot}}. \tag{12}$$

Each individual-task or suite result is reported as $100\widehat{p}\% \pm 100\Delta_{95}$ percentage points. Benchmark-level tables macro-average the corresponding task or suite success rates and omit confidence intervals rather than averaging individual intervals. Execution horizons are evaluated and reported separately.

Robomimic. We use the checkpoint with the best teacher-forced validation accuracy and evaluate each task with 10 rollout seeds, with 30 episodes per seed, for a total of 300 episodes per task and execution horizon. The rollout seeds control environment initialization and are distinct from the policy-training seeds discussed below.

LIBERO. Each of the Spatial, Object, Goal, and LIBERO-10 suites contains 10 tasks. We use the checkpoint with the best teacher-forced validation accuracy. We evaluate each task for 50 episodes using LIBERO's fixed set of 50 initial states. This gives 500 episodes per suite and execution horizon. The benchmark-level cells in Table 1 equally macro-average the four suite success rates.

RoboTwin. We use the checkpoint with the best teacher-forced validation accuracy and evaluate each of the three tasks on 50 random initializations, for a total of 150 episodes per execution horizon. Table 8 summarizes the rollout protocols across the three benchmarks.

Table 8: Primary rollout evaluation protocols across simulation benchmarks. Episode counts apply separately to each execution horizon.

Setting	RoboMimic	LIBERO	RoboTwin
Checkpoint selection	Best teacher-forced validation accuracy		
Execution horizon	$H_{\text{exec}} \in \{4, 8\}$ for primary results; 8 by default		
Policy sampling	Greedy		
Evaluation units	Individual tasks	Spatial, Object, Goal, LIBERO-10; 10 tasks per suite	3 tasks
Episode allocation	10 rollout seeds $\times$ 30 episodes per seed	50 fixed initial states per task	50 random initializations per task
Total episodes	300 per task	500 per suite	150 across all 3 tasks
Episode limit	500 environment steps	500 environment steps for every suite	300 / 500 / 600 environment steps (see note)
Success aggregation	Binary success averaged over 300 episodes per task	Binary success averaged over 500 episodes per suite	Binary success averaged over 50 episodes per task

Note. RoboTwin episode limits: grabbing the roller, 300; placing the burger and fries, 500; placing cans in the plastic box, 600 environment steps.

B.3 Training Hyperparameters and Randomness

Tokenizer training. Table 9 summarizes the tokenizer-training hyperparameters used for the three simulation benchmarks. Our primary goal is to compare downstream policy performance across tokenizer designs and configurations. Because the sweep covers a large number of configurations, we allocate compute to broader design coverage rather than repeated tokenizer-training seeds within each configuration. Each tokenizer configuration is therefore trained once. Consequently, our experiments characterize between-design differences but do not estimate within-design variance arising from tokenizer training; these comparisons should be interpreted accordingly.

Table 9: Tokenizer-training configuration for the three simulation benchmarks. A single recipe is shared by all 22 tokenizer configurations within a benchmark; rows spanning the three columns are additionally identical across benchmarks.

Hyperparameter	RoboMimic	LIBERO	RoboTwin
Data and operating point			
Training tasks	Can, Lift, Square	Spatial, Object, Goal, LIBERO-10	All tasks
Action dimension	7	7	14
Chunk size H	8		
Codebook size K	128		
Quantization levels L	8		
Action normalization	Per-dimension mean/standard deviation		
Optimization			
Optimizer	AdamW, $\beta = (0.9, 0.95)$		
Base learning rate	5×10^{-5}		
Learning-rate schedule	Constant, 100 warmup steps		
Gradient-clipping norm	1.0		
Training budget			
Training epochs	1000		
Batch size	256		
Objective and selection			
Commitment weight	0.25 (VQ, RVQ, NRVQ, QinCo)		
Checkpoint selection	Best validation checkpoint		

Policy training. Table 10 summarizes the policy-training hyperparameters used for the three simulation benchmarks. We estimate policy-training variability using two tokenizer instances, FSQ and

NRVQ, with $K = 128$ and $L = 8$. For each tokenizer, we train the same multi-task autoregressive policy with 10 policy-training seeds and evaluate every trained policy on Can, Lift, and Square using the same bank of 10 rollout seeds and 30 episodes per rollout seed. Each run uses the checkpoint with the best teacher-forced validation accuracy.

Table 10: Autoregressive policy configuration for the three simulation benchmarks. A single recipe is shared by every tokenizer configuration and training objective within a benchmark; rows spanning the three columns are additionally identical across benchmarks.

Hyperparameter	**RoboMimic**	**LIBERO**	**RoboTwin**
ARCHITECTURE			
Model dimension		256	
Transformer layers		4	
Attention heads		4	
Feed-forward dimension	256	1024	1024
Dropout	0.1	0.0	0.0
Observation conditioning		Cross-attention	
Visual backbone		Trainable, 512-dim feature per view	
Camera views	2	2	4
Image resolution		224×224 per view	
Vocabulary size		128	
Action tokens per chunk		8	
INPUTS AND ACTION SPACE			
Proprioception	EEF position, quaternion, joint velocity (14)	EEF position, axis–angle, gripper joints (8)	Per arm: EEF position, 6-D rotation, gripper (20)
Task conditioning	One-hot task identity	Predefined task instruction	One-hot task identity
Language encoder	–	Frozen T5-base, ≤ 24 tokens	–
Action representation		Delta end-effector pose	
OPTIMIZATION			
Optimizer		AdamW	
Base learning rate		1×10^{-4}	
Learning-rate schedule		Constant, warmup over 5% of training	
Weight decay		1×10^{-3}	
Gradient-clipping norm		1.0	
EMA decay		0.999	
TRAINING AND SELECTION			
Training epochs	300	500	500
Batch size		512	
Feature-space blur $\sigma_{\max}$	0.0	2.0	2.0
Augmentation anneal	–	200 epochs	200 epochs
Checkpoint selection		Best teacher-forced validation accuracy	

Table 11 reports the resulting variation in rollout success. The spread is modest on Can and Lift but larger on Square: the across-training-seed standard deviation is 4.3 percentage points for FSQ and 5.8 points for NRVQ, with ranges of 14.3 and 16.7 points, respectively.

This experiment confirms that policy training exhibits non-negligible randomness, with the magnitude of variation differing across tasks and tokenizers. At the same time, the observed variability is not so large as to obscure the broader trends that emerge repeatedly across the full tokenizer sweep. We therefore use this study to contextualize individual pairwise differences, while relying on the large number of comparisons in the sweep to support aggregate conclusions. Because each configuration in the full sweep is trained with a single policy seed, we interpret the sweep primarily through trends that recur across many comparisons rather than the precise magnitude of any individual pairwise difference.

Table 11: Policy-training-seed robustness on RoboMimic. Each row summarizes rollout success over 10 independently trained multi-task policies paired with the same fixed tokenizer. Every policy is evaluated at $H = H_{\text{exec}} = 8$ on the same 300 episodes per task. Values are reported as mean $\pm$ standard deviation across policy-training seeds; all values are percentages.

Tokenizer	Task	Mean $\pm$ SD	Min–max
FSQ	Can	88.7 ± 1.9	86.0–91.3
	Lift	94.1 ± 2.5	88.0–96.3
	Square	51.5 ± 4.3	44.0–58.3
NRVQ	Can	87.9 ± 2.4	84.3–91.3
	Lift	96.4 ± 1.2	94.0–98.0
	Square	54.5 ± 5.8	44.3–61.0

Table 12: Training and evaluation configuration shared by the sixteen VLA policy runs on LIBERO. The runs differ only in the action tokenizer and the resulting token targets.

Hyperparameter	**Value**
ARCHITECTURE AND ACTION REPRESENTATION	
VLM backbone	SmolVLM2-256M-Video-Instruct (pretrained; 228.1M parameters)
Action expert	Llama, 30 layers, hidden dimension 288, feed-forward dimension 1152 (random initialization; 43.2M parameters)
Total / effective trainable parameters	271.4M / 268.0M
Fine-tuning	Full-parameter
Action vocabulary	$K = 128$ shared tokens
Action tokens per chunk	$L = 8$
INPUTS AND PREPROCESSING	
Visual input	Two camera views, tiled horizontally into one image
Source frame / VLM image tile	256×256 per view / 512 pixels
Visual tokens	64 total
Training augmentation	Random crop (0.875), resize to 224×224, and color jitter $(0.2, 0.2, 0.2, 0.05)$
Proprioception	LIBERO 8-D state rendered as text in the prompt
System prompt	Enabled
OPTIMIZATION	
Optimizer	AdamW
Training length	50,000 optimizer steps
Batch Size	132
Learning-rate schedule	100-step linear warmup; cosine decay from 1×10^{-4} to 2×10^{-5}
Weight decay	1×10^{-10}
Gradient-clipping norm	1.0
EMA decay	0.999
EVALUATION	
Checkpoint	Last checkpoint
Execution horizon	$H_{\text{exec}} = 8$
Episodes	500 per suite (50 for each of 10 tasks)

VLA architecture, training, and evaluation. To test whether our tokenizer-side interventions also benefit a policy built on a pretrained vision-language backbone, we fine-tune SmolVLM2-256M (Marafioti et al., 2025) with an ActionCodec-style Llama action expert (Dong et al., 2026) on LIBERO. The action expert autoregressively predicts eight tokens, which the corresponding action tokenizer decodes into an action chunk. We evaluate eight paired tokenizer configurations (Table 2).All sixteen policies use the same VLA architecture, optimization recipe, and evaluation protocol; only the action tokenizer and its resulting token targets differ. The VLA is fine-tuned end to end without LoRA. Although all model parameters are included in optimization, 3.3M parameters in the final VLM decoder layer receive no gradient because the VLM supplies only its keys and values to the action expert.

Table 12 reports the complete shared configuration. Note that we report results from the final training checkpoint after 50k gradient steps. Due to computational constraints, we were not able to train the VLA to convergence. Therefore, our results highlight the strong performance of ProAct in the limited training budget regime.

C Additional Action-Tokenization Analysis

This section gives the measurements and analysis behind Section 4: reconstruction error (Appendix C.1), nominal bitrate (Appendix C.2), chunk size (Appendix C.3), training-support membership (Appendix C.4), and the one-step error decomposition (Appendix C.5).

Conventions. A *single-task* tokenizer–policy pair is trained on demonstrations from one Robomimic task and evaluated on that same task, whereas *multi-task* denotes joint training on a dataset of all tasks. Each subsection states which condition it uses, and Appendix C.5 is analytical and does not depend on the choice. Shaded bands and horizontal spans show min–max ranges over tokenizer designs unless labeled as 95% intervals.

C.1 Reconstruction-Error Details

C.1.1 Does Reconstruction Error Rank Tokenizers?

We ask when reconstruction error ceases to rank downstream performance across tokenizer–policy configurations. The correlation study in Figure 6 retains every valid run in the single-task sweep: 375 for Can, 242 for Lift, and 232 for Square, for 849 tokenizer–policy runs in total. The cohort spans all five quantizer families, multiple tokenizer backbones and training variants, both codebook-sharing choices where applicable, and two nominal rates. With codebook size 128, 123 runs use three levels (21 bits per chunk) and 726 use eight levels (56 bits per chunk). Every row has a unique tokenizer, a matching same-task policy with no recorded training issue, a valid best checkpoint, and an evaluation over 10 rollout seeds and 300 episodes. We therefore apply no filtering based on tokenizer family, bitrate, reconstruction error, or rollout success: high-error and low-success runs are both necessary for characterizing the relationship. Within the RVQ, NRVQ, and QinCo subset discussed in Section 4.1, the taskwise Spearman correlations between validation reconstruction error and rollout success are $\rho = 0.45$, 0.35, and 0.50 for Can, Lift, and Square, respectively.

C.1.2 Calibrating Action-Error Tolerance

We calibrate action-space sensitivity using an observation-conditioned continuous FM policy with $H = H_{\text{exec}} = 8$ for each task separately, treating its clean prediction as the expert action reference for this diagnostic. At every step of each predicted chunk, we independently perturb the three translation channels with $\eta_j^{\text{trans}} \sim \mathcal{U}(-\epsilon_{\text{trans}}, \epsilon_{\text{trans}})$ and the three rotation channels with $\eta_j^{\text{rot}} \sim \mathcal{U}(-\epsilon_{\text{rot}}, \epsilon_{\text{rot}})$. The gripper is left unchanged, and the perturbed action is clipped to its valid range before execution. Viewed as a reconstruction of the clean FM action, the perturbed action incurs a reconstruction-form error whose magnitude we control directly. Since a variable drawn from $\mathcal{U}(-\epsilon, \epsilon)$ has expected absolute magnitude $\epsilon/2$, averaging the six perturbed channels and the unchanged gripper channel gives the following expected seven-dimensional MAE before clipping:

$$\mathbb{E}\left[\frac{\|\eta\|_1}{7}\right] = \frac{3}{14}\left(\epsilon_{\text{trans}} + \epsilon_{\text{rot}}\right). \tag{13}$$

This quantity measures deviation from the clean FM expert in the same raw-action units and with the same MAE form as ℓ; it is not the tokenizer's validation reconstruction error $\mathcal{E}_{\text{rec}}$.

We use two complementary perturbation settings. The *joint-noise* sweep sets $\epsilon_{\text{trans}} = \epsilon_{\text{rot}} = \epsilon$, for which Equation 13 reduces to $3\epsilon/7$, and measures the overall tolerance to pose-action error. The *independent-noise* sweep varies ϵ_{trans} and ϵ_{rot} separately to distinguish translational from rotational sensitivity. We first present the joint-noise calibration across Robomimic, LIBERO, and RoboTwin, followed by the independent-noise sweep on Robomimic.

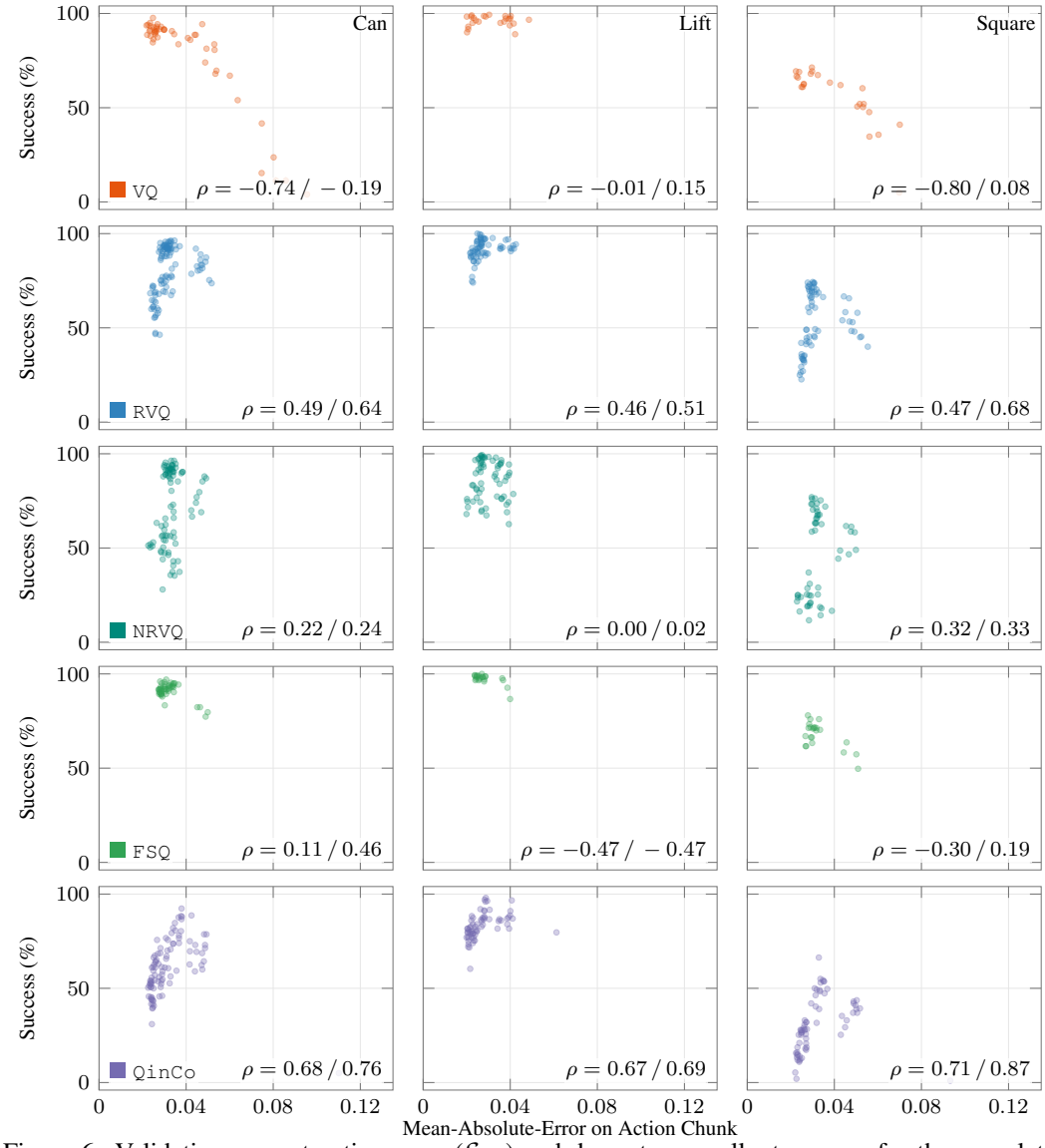


Figure 6: Validation reconstruction error ($\mathcal{E}_{\text{rec}}$) and downstream rollout success for the complete single-task tokenizer–policy sweep, separated by quantizer family (rows: VQ, RVQ, NRVQ, FSQ, QinCo) and task (columns: Can, Lift, Square). Each point is one valid run; all panels share the same axes, and each column sums to the 375/242/232 Can/Lift/Square totals reported above. Bottom-right annotations report Spearman's ρ for all runs / runs with validation MAE < 0.04 (excluding tokenizers with large reconstruction error).

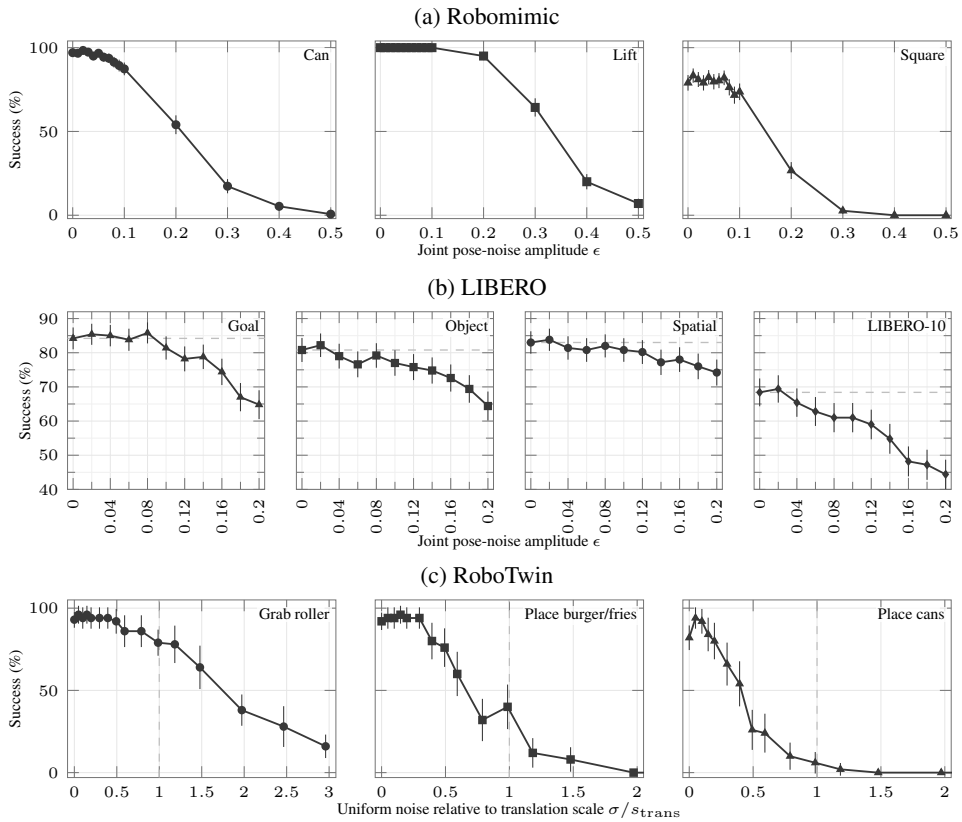


Figure 7: FM response to joint uniform pose-action noise across simulation benchmarks, with one panel per task or suite. Translation and rotation use equal noise amplitudes, while gripper coordinates remain unperturbed. Markers show success and cap-free bars show 95% intervals. (a) Robomimic: 300 episodes per point; the fine sweep covers $\epsilon = 0.01$–0.10 and the coarse sweep $\epsilon = 0.20$–0.50. (b) LIBERO: 500 episodes per point at $\epsilon = 0$–0.20. (c) RoboTwin: $H = 64$ and $H_{\text{exec}} = 8$; both arms receive pose noise from $\mathcal{U}(-\sigma, \sigma)$. The horizontal axis normalizes σ by the translation-channel standard deviation $s_{\text{trans}} = 0.00507$; dashed lines mark $\sigma = s_{\text{trans}}$. Clean references use 100 episodes per task and noisy points use 50, except the three original Grab roller points at $\sigma \in \{0.005, 0.010, 0.015\}$, which use 100.

In the Robomimic joint-noise sweep (Figure 7(a)), success remains near the clean anchors through $\epsilon = 0.07$: 93.7% versus 97.0% on Can, 100% versus 100% on Lift, and 82.0% versus 79.0% on Square. At $\epsilon = 0.30$, success falls to 17.3%, 64.3%, and 2.7%, respectively, confirming that the tolerance scale is task dependent.

The LIBERO joint-noise sweep (Figure 7(b)) shows the same qualitative pattern across four suites: sufficiently large action noise degrades success, but the onset and rate of degradation differ by suite.

RoboTwin uses the same joint-noise design, with perturbations drawn independently from $\mathcal{U}(-\sigma, \sigma)$. Its 14-dimensional dual-arm action contains 12 perturbed pose coordinates and two unchanged gripper coordinates, so the expected pre-clipping MAE is $3\sigma/7$. Because the empirical standard deviations of the translation and rotation coordinates are $s_{\text{trans}} = 0.00507$ and $s_{\text{rot}} = 0.02957$, respectively, we report the noise level relative to the translation scale, σ/s_{trans}.

The fine-grained RoboTwin sweep (Figure 7(c)) reveals a task-dependent plateau followed by a smooth roll-off. At $\sigma/s_{\text{trans}} \approx 0.20$, success is 94%, 94%, and 80% on Grab roller, Place burger/fries, and Place cans, respectively, compared with clean references of 93%, 92%, and 82%. Linear interpolation places the half-success points at $\sigma/s_{\text{trans}} = 1.81$, 0.69, and 0.44, a fourfold range across the three tasks.

Across Robomimic, LIBERO, and RoboTwin, the task-dependent onset and rate of degradation rule out a single universal action-error threshold.

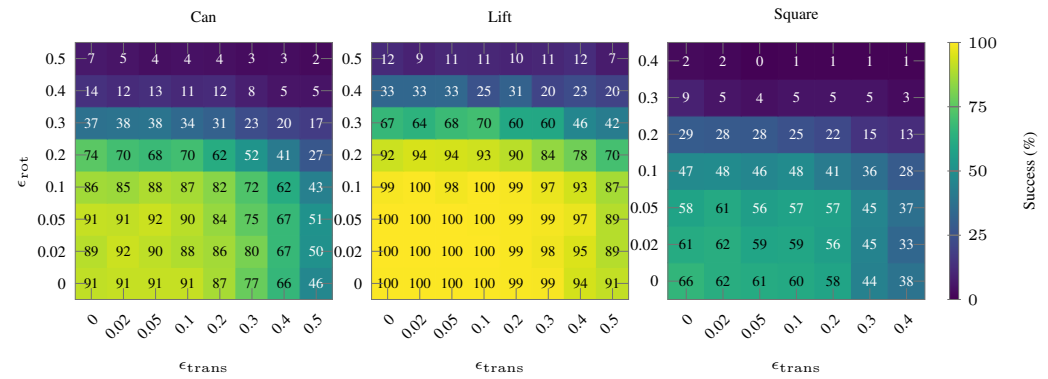


Figure 8: Robomimic FM success under independently varied translation and rotation noise. Cells report success over 300 episodes at $H = H_{\text{exec}} = 8$, with the gripper unperturbed; the SQUARE grid omits $\epsilon = 0.5$. These runs use a different FM policy from Figure 7(a), so sensitivity patterns, but not absolute success levels, are comparable between the two diagnostics.

By varying the two amplitudes separately, Figure 8 isolates their effects within each FM policy. At comparable amplitudes, rotation noise produces the sharper degradation, though the response surface remains strongly task dependent.

C.1.3 Isolating Tokenizer Distortion

The action-noise calibration establishes the magnitude of raw-action error that begins to impair control. We next ask whether reconstruction distortion introduced by the tokenizer alone can reach this consequential regime. To isolate this effect from token prediction, we pass each action chunk A^{FM} produced by the FM policy through a frozen action tokenizer and detokenizer before execution, without training or invoking an AR policy. We measure the resulting distortion using the on-policy reconstruction MAE, $\mathbb{E}[\ell(A^{\text{FM}}, \mathcal{D}(\mathcal{T}(A^{\text{FM}})))]$, including the gripper coordinate.

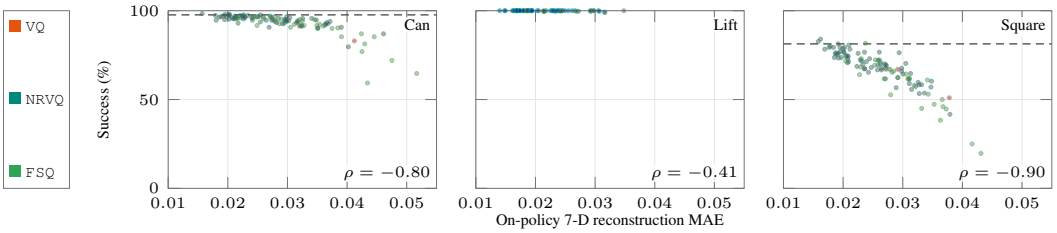


Figure 9: Tokenized-FM success versus on-policy seven-dimensional reconstruction MAE. Each point is one tokenizer condition; color denotes quantizer family. Dashed lines mark the available un-tokenized FM references for Can and Square. Spearman correlations are shown in-panel. All results use $H = H_{\text{exec}} = 8$ and 300 episodes.

Figure 9 reports all 273 tokenized conditions. Reconstruction error is strongly negatively associated with success on Can and Square, and more weakly on Lift, where success remains near ceiling across much of the observed error range. Because token prediction is absent, these results directly characterize how tokenizer-induced reconstruction distortion affects control.

C.1.4 Separating Distortion from Token Prediction

The tokenized-FM policy isolates reconstruction distortion, whereas the tokenizer–AR policy additionally has to predict the token sequence. Pairing the two evaluations on the same tokenizer therefore separates the two error sources. Figure 10 does this for the 63 tokenizers evaluated under both policies: the 9 `TF-FSQ-S` configurations of the bitrate sweep together with its 27 `TF-NRVQ-S` and 27 `TF-NRVQ-L` configurations.

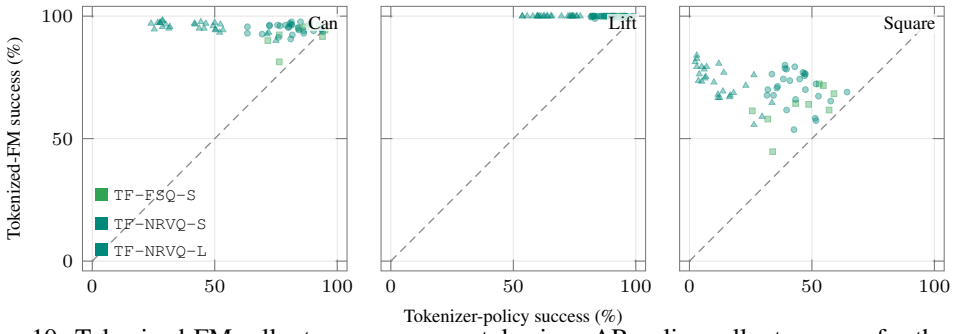


Figure 10: Tokenized-FM rollout success versus tokenizer–AR policy rollout success for the same tokenizer. Each point is one tokenizer configuration: color denotes quantizer family, filled squares mark `TF-FSQ-S`, circles `TF-NRVQ-S`, and triangles `TF-NRVQ-L`. The `TF-NRVQ` points span latent dimensions 32, 64, and 128, and both families span 21–80 bits per chunk. The dashed line marks equal success. Both axes use $H = H_{\text{exec}} = 8$, multi-task training, and 300 episodes per condition; the panels contain the 63 tokenizers for which both evaluations are available.

Removing token prediction shifts nearly every tokenizer above the diagonal, and the two axes span very different ranges: tokenizer–AR success varies widely across these tokenizers, whereas tokenized-FM success stays within a comparatively narrow high band on each task. Reconstruction distortion alone therefore does not account for the spread in tokenizer–AR performance, which instead reflects the difficulty of predicting the token sequence.

Taken together, the four diagnostics support reconstruction as a feasibility condition: sufficiently large distortion degrades control, yet once reconstruction error enters a sufficiently low regime, further reductions do not reliably rank downstream control performance.

C.2 Nominal Bitrate Analysis

Higher nominal bitrate lowers reconstruction error, so it is tempting to maximize it. Section 4.1 showed that reconstruction only needs to be accurate enough, so higher bitrate need not improve control. We test this at fixed chunk size: higher bitrate generally improves reconstruction, while downstream success remains non-monotonic and can even decline.

Nominal bitrate bounds capacity, not encoded information. For an L-level tokenizer with K choices per level, $B_{\text{nom}} = L \log_2 K$ is the log-cardinality of the code space: the most information a chunk's token sequence could carry. Writing $C_{1:L}$ for that sequence, the information it does carry obeys

$$\mathbb{H}(C_{1:L}) \;\le\; L \log_2 K \;=\; B_{\text{nom}},$$

This gap is common in practice, since discrete representation learning substantially under-utilizes a nominal codebook (Mentzer et al., 2024). Enlarging the code space enlarges the set of achievable rate–distortion points, so reconstruction should improve with B_{nom}. Whether that added capacity is used, and whether the code it yields is easier or harder for an autoregressive policy to predict, is not determined by B_{nom}: excess action-token capacity can absorb high-frequency variation that complicates autoregressive learning, whereas compression can strip temporal redundancy and leave more informative prediction targets (Dong et al., 2026; Pertsch et al., 2025). The rest of this subsection measures the effect on reconstruction and on downstream success separately.

Setup. Figure 11 sweeps B_{nom} over the full grid $K \in \{128, 512, 1024\}$ and $L \in \{3, 6, 8\}$, giving nine rates from 21 to 80 bits per chunk, for the Transformer-backbone TF-FSQ and TF-NRVQ tokenizers. The chunk size is fixed at $H = H_{\text{exec}} = 8$ throughout, so bitrate is the only design factor that varies; each point is a single tokenizer configuration paired with a multi-task AR policy and evaluated over 300 rollout episodes per task. We exclude the prefix-only partial-decode runs here, because they change the deployed bitrate without recomputing the tokenizer's offline validation reconstruction error, which would leave the two axes of Figure 11 measured at different rates. The upper panel reports validation reconstruction MSE in raw action units; this is not the MAE $\mathcal{E}_{\text{rec}}$ of Section 4.1, so its magnitudes are not comparable to the tolerance scale calibrated there. Every

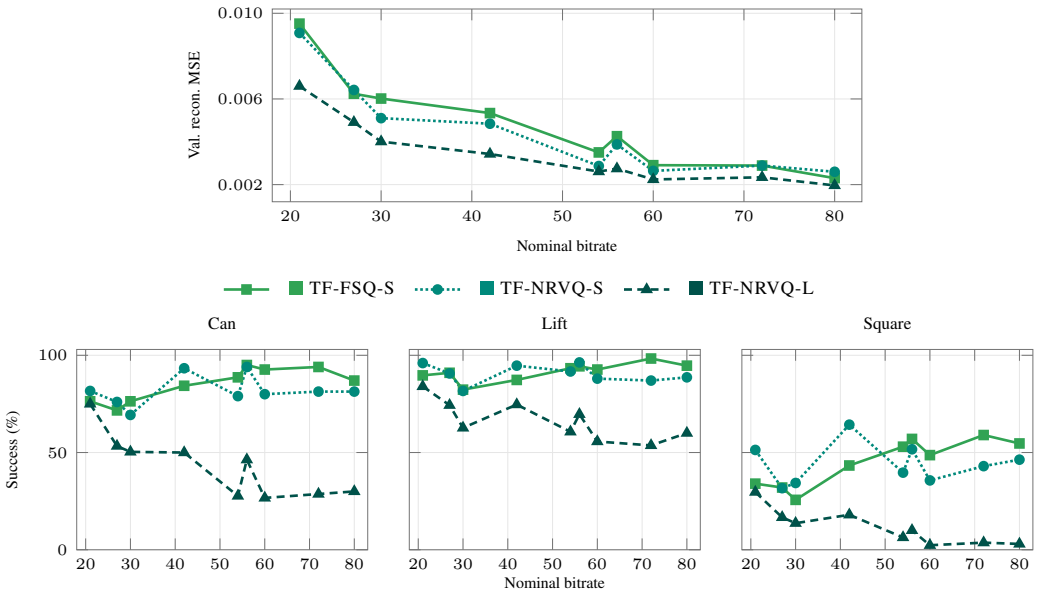


Figure 11: Robomimic full-decode bitrate sweep at fixed $H = H_{\text{exec}} = 8$. The upper panel shows tokenizer validation reconstruction MSE in raw action units, which is a property of the tokenizer and therefore shared across tasks; the lower panels show the corresponding taskwise rollout success over 300 episodes. Nominal bitrate is $L \log_2 K$ for $K \in \{128, 512, 1024\}$ and $L \in \{3, 6, 8\}$. Each point is one tokenizer configuration, so no averaging or design range is involved. All tokenizers use Transformer backbones (TF); S/L denote shared and level-specific codebooks (Appendix B.1.3). TF-FSQ-S has a single configuration per bitrate.

policy in this sweep was trained with horizontal image flipping enabled, so comparisons within the sweep hold, while absolute success levels are not comparable to the other tables in this paper.

Effect on reconstruction. Across TF-FSQ-S, TF-NRVQ-S, and TF-NRVQ-L, validation reconstruction MSE generally decreases as nominal bitrate increases, with only small local reversals between nearby rates. Greater representational capacity therefore usually improves reconstruction fidelity, as expected from rate–distortion behavior.

Effect on downstream success. Downstream success does not follow the same monotonic trend. TF-FSQ-S generally improves with bitrate but peaks before the largest rate, TF-NRVQ-S fluctuates without a consistent direction, and TF-NRVQ-L performs best at the lowest tested rate on every task even though it has the lowest reconstruction error of the three tokenizers at every rate. Thus, additional bitrate can improve reconstruction without reliably improving control, and the downstream effect depends on the tokenizer design.

Interaction with chunk size. Bitrate must also be read relative to the chunk size H, since a fixed B_{nom} allocates only B_{nom}/H bits per action step, or $B_{\text{nom}}/(Hd_a)$ per scalar action dimension. Longer chunks consequently need more capacity to hold reconstruction fidelity fixed, which is what Appendix C.3 measures. Raising the bitrate relieves that representational bottleneck, yet it does not address the separate difficulty of predicting a longer action sequence or executing it open-loop, and the results above show that it does not reliably help even at fixed H. Hence, bitrate and chunk size should not simply be maximized. Bitrate is best treated as a *conditional capacity requirement*: for a chosen H it must be large enough to enter the feasible reconstruction regime of Section 4.1; beyond that point, H and H_{exec} should be chosen by downstream success and inference cost.

C.3 Chunk-Size Details

Chunking introduces two distinct effects. Increasing the prediction horizon H requires the policy to predict farther into the future and, for a tokenizer with fixed capacity, compresses more action steps

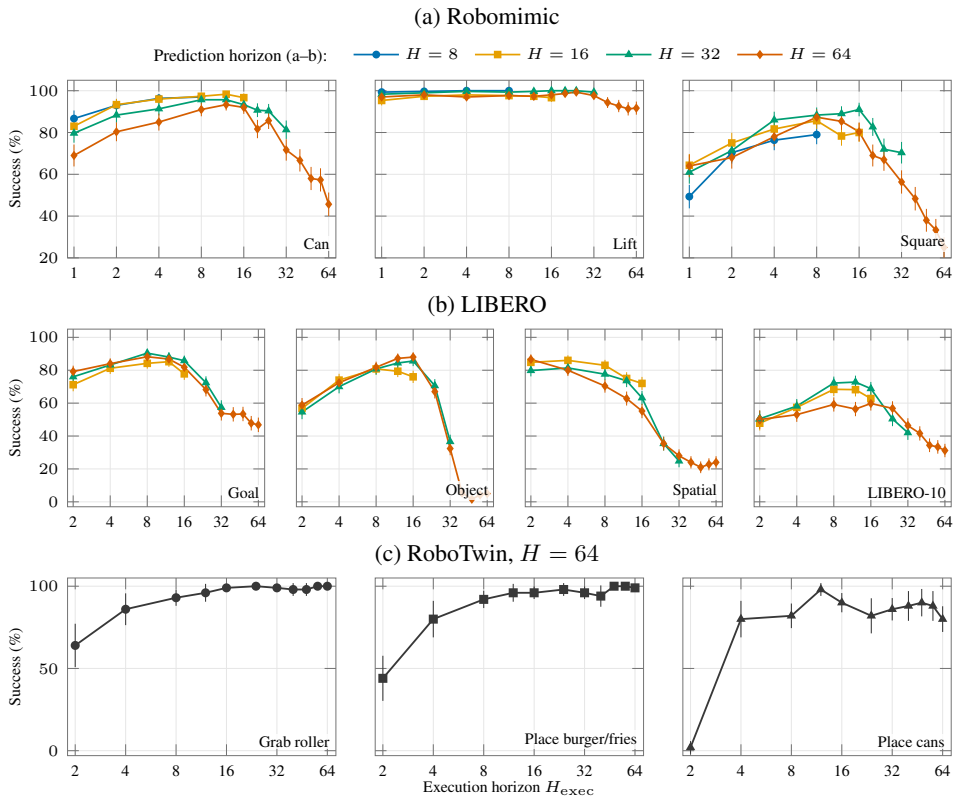


Figure 12: FM success across prediction horizon H and execution horizon H_{exec}, with one panel per task or suite. Bars show 95% bootstrap intervals, and curves terminate at $H_{\text{exec}} = H$. The shared color legend applies to (a–b), which use the last checkpoint. (a) Robomimic: 300 episodes per point. (b) LIBERO: 500 episodes per suite, except the hollow LIBERO-10 points at $(H, H_{\text{exec}}) = (32, 2)$ and $(64, 2)$, with 400 and 300 completed episodes, respectively. (c) RoboTwin: fixed $H = 64$; evaluations at $H_{\text{exec}} \in \{8, 16, 32, 64\}$ use 100 episodes per task, and the remaining horizons use 50. Episode limits are 300, 500, and 600 environment steps for Grab roller, Place burger/fries, and Place cans, respectively.

into the same representation. Increasing the execution horizon H_{exec} instead reduces replanning frequency and increases the duration of open-loop execution.

We separate these effects by first sweeping H and H_{exec} with continuous FM policies, which isolate the prediction- and execution-horizon effects without tokenization (Appendix C.3.1). We then repeat the sweep with the tokenizer–AR policy (Appendix C.3.2) and compare the two settings directly. The results show that H_{exec} exhibits an intermediate optimum in both settings, while increasing H introduces an additional cost for tokenized policies once reconstruction leaves the feasible regime identified in Section 4.1.

C.3.1 Chunk Size with Flow-Matching Policies

Because FM policies predict action chunks without tokenizing them, no horizon effect measured here can be attributed to a tokenizer. All three benchmarks show the same underlying shape in H_{exec}: success is depressed when the policy replans almost every step, rises to an interior optimum, and falls again as execution approaches the full predicted chunk. The swept ranges show different parts of this shape: Robomimic shows both the rise and the fall, LIBERO mainly the fall, and RoboTwin mainly the rise.

Evaluation protocol. We evaluate FM policies on Robomimic, LIBERO, and RoboTwin across the available prediction horizons H and execution horizons H_{exec}, and compare their rollout success using the same evaluation protocol as the tokenized policies.

Robomimic. On Robomimic, increasing H_{exec} from one step consistently helps until a task- and horizon-dependent intermediate range, whereas executing most or all of a long chunk degrades success. After macro-averaging Can, Lift, and Square, the best execution horizon is 8 for $H \in \{8, 16\}$, 12 or 16 for $H = 32$, and 8 or 12 for $H = 64$. For $H = 64$, macro-average success falls from 92.0% at $H_{\text{exec}} \in \{8, 12\}$ to 54.1% when the entire predicted chunk is executed. Success therefore both rises and falls with H_{exec}, but the optimum is close to the short end, so the rise spans only a few steps.

LIBERO. LIBERO exhibits the same broad preference for moderate execution horizons, but the optimum is suite dependent. The four-suite macro-average peaks at $H_{\text{exec}} = 8$ for all three prediction horizons, reaching 79.1%, 80.2%, and 74.9% for $H = 16$, 32, and 64, respectively. Long open-loop execution is especially damaging for Object and Spatial; at $H = 64$, the macro-average declines to 26.8% at $H_{\text{exec}} = 64$. The short-horizon end is the least completely evaluated part of this grid, since both hollow LIBERO-10 points in Figure 12(b) sit at $H_{\text{exec}} = 2$; we therefore do not read a replanning penalty into LIBERO on its own.

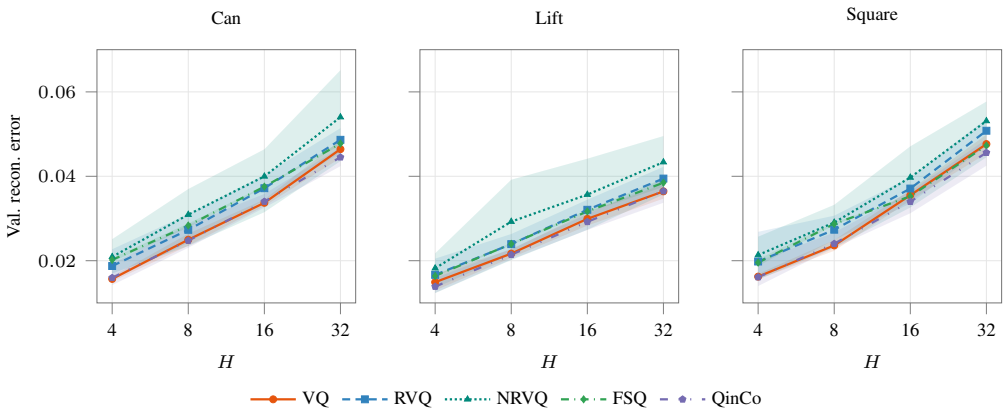
RoboTwin. RoboTwin shows the strongest benefit from increasing the execution horizon. Increasing H_{exec} from 2 to 4 raises success from 64% to 86% on Grab Roller, from 44% to 80% on Place Burger and Fries, and from 2% to 80% on Place Cans. Performance remains high once H_{exec} reaches roughly 12, with the first two tasks near ceiling and Place Cans mostly between 80% and 90%. Thus, as in Robomimic, overly frequent replanning is detrimental, but the effect is substantially stronger on RoboTwin.

Implication for tokenizer policies. The tokenizer-free sweeps show that a large prediction horizon is not inherently harmful when paired with a moderate execution horizon: even at $H = 64$, FM reaches 92.0% on Robomimic and 74.9% on LIBERO. The degradation observed at large H in the tokenized setting therefore cannot be explained by prediction horizon alone.

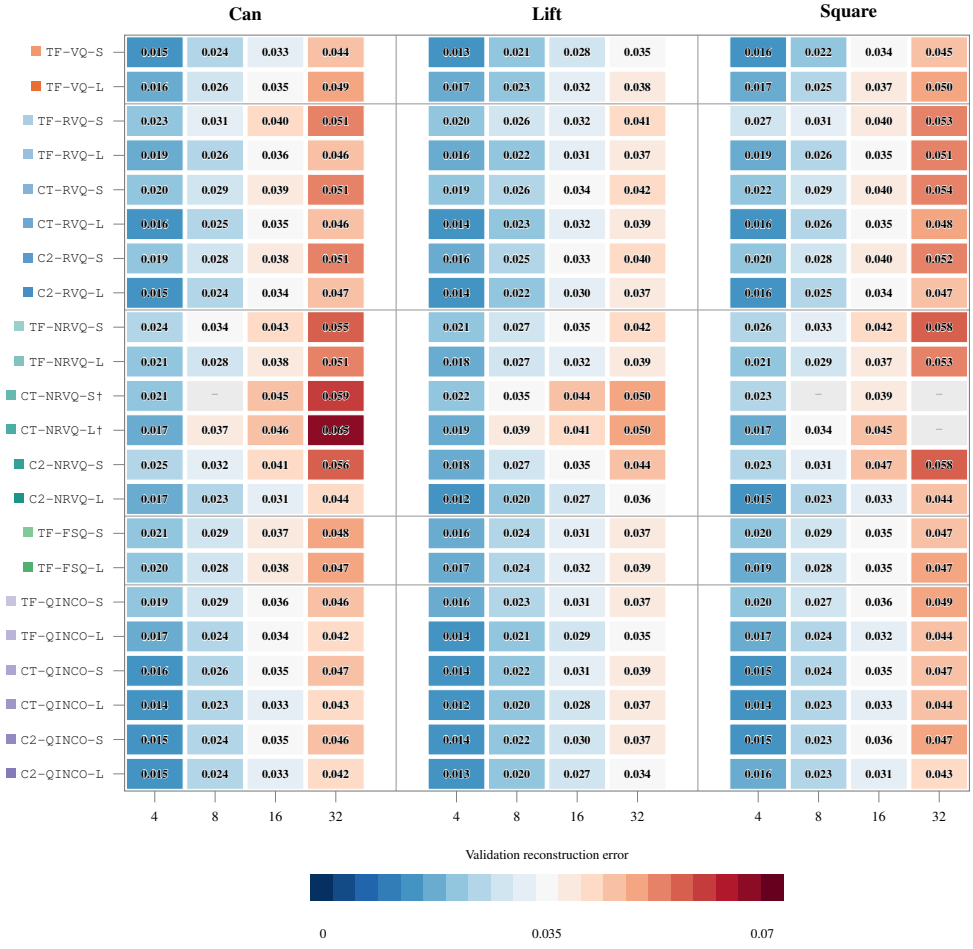
C.3.2 Chunk Size with Tokenizers and Tokenizer–AR Policies

Evaluation protocol. We evaluate all valid (H, H_{exec}) pairs for $H \in \{4, 8, 16, 32\}$ using Reconstruction-only tokenizers and matching single-task policies. To make horizon comparisons insensitive to failed tokenizer runs, each task uses the intersection of configurations valid at every H: 21 for Can, 22 for Lift, and 20 for Square. All success values use the best policy checkpoint, greedy decoding, and 300 episodes per tokenizer–task condition. Reconstruction error in this subsubsection is always the validation mean of ℓ between decoded and target chunks in raw action space rather than latent or code space. Two features of this protocol differ from the FM sweeps above, which report one policy per value and use the last checkpoint: every value here averages 20–22 tokenizer configurations, and checkpoints are selected by validation accuracy.

Reconstruction difficulty at larger H. We first examine whether increasing H makes action chunks harder to reconstruct under a fixed token budget. This tokenizer-specific effect could explain any additional degradation beyond the trends already observed in the tokenizer-free sweeps. Figure 13 reports task-specific family summaries together with the validation reconstruction error of every tokenizer configuration. Family curves average the configurations available for each task, with min–max ranges across designs.

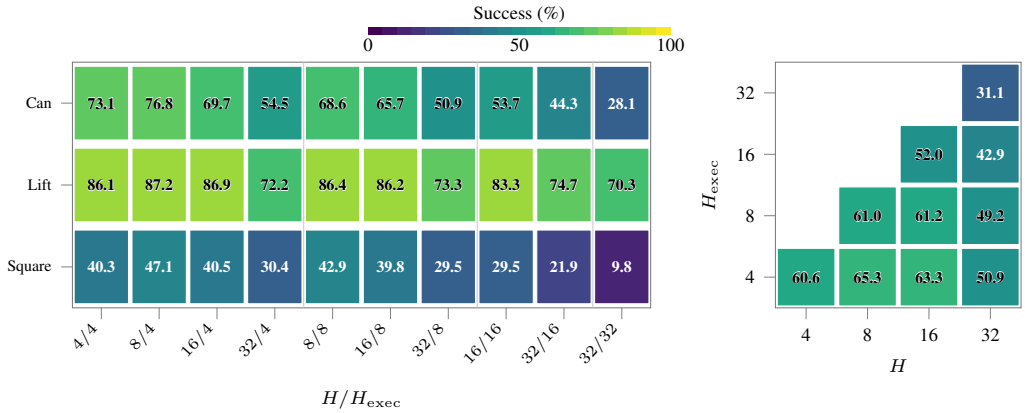


(a) Task-specific family means and tokenizer-design ranges.



(b) Tokenizer-level validation reconstruction errors.

Figure 13: Detailed validation-reconstruction behavior as the chunk size increases. **(a)** Color identifies quantizer family, with a distinct line style and marker shape per family, and shaded regions show min–max variation over valid tokenizer designs. **(b)** Rows follow the tokenizer ordering in Appendix B.1; all task blocks share a blue-to-red scale from 0 to 0.07, with red denoting larger validation reconstruction error. A dagger marks configurations excluded from the globally matched main-text aggregation, and gray "–" cells denote above-threshold tokenizer runs whose validation errors are unavailable.



(a) Matching single-task policies. (b) Multi-task policies.

Figure 14: Detailed downstream success across action and execution horizons. **(a)** Each row averages the fixed valid tokenizer set for one task. Columns are grouped by increasing execution horizon H_{exec} and, within each group, by increasing chunk size H; labels are shown as H/H_{exec} and the vertical rules separate the H_{exec} groups. **(b)** Each cell averages the 20 configurations valid at every multi-task horizon and then macro-averages Can, Lift, and Square. Both panels use a common Viridis scale from 0 to 100% success; numbers inside cells are percentages.

Validation reconstruction error rises systematically with chunk size. Across the multi-task tokenizer set, the mean increases from 0.0168 at $H = 4$ to 0.0247, 0.0328, and 0.0437 at $H = 8, 16, 32$, respectively. The single-task family curves show the same broad trend despite variation across tokenizer designs and tasks. Only the largest chunk size leaves the regime that Section 4.1 identifies as feasible: $H = 32$ is the first horizon whose mean validation reconstruction error, 0.0437, exceeds the $\approx$0.04 Robomimic tolerance, whereas $H \in \{4, 8, 16\}$ all remain inside it.

Downstream effects of H and H_{exec}. Downstream success turns over at the same chunk size. Figure 14 separates the aggregate in Figure 2(c) by task and adds the corresponding multi-task sweep.

Increasing the prediction horizon is not uniformly beneficial. Across both execution horizons, performance is relatively stable from $H = 8$ to $H = 16$ but drops substantially at $H = 32$: mean success decreases from 63.3% to 50.9% at $H_{\text{exec}} = 4$, and from 61.2% to 49.2% at $H_{\text{exec}} = 8$. This degradation coincides with reconstruction error leaving the low-error regime identified in Section 4.1, suggesting that longer prediction horizons become harmful once they make the action chunk sufficiently difficult to reconstruct.

C.4 Training-Support Membership Diagnostics

We ask two questions: how often the policy's token sequences leave the empirical training support, and whether restricting decoding to observed sequences helps.

Setup. Every diagnostic in this subsection uses the *single-task* condition at $H = H_{\text{exec}} = 8$: each of the 22 tokenizer configurations is paired with a policy trained and evaluated on the same Robomimic task, giving 66 tokenizer–task conditions, and the offline and online analyses below draw on the identical cohort. Let $\mathcal{C}_{\text{train}} = \{\mathcal{T}(A) : A \in \mathcal{A}_{\text{train}}\}$ be the finite set of encoded training chunks, which we call the empirical training support; note that $\mathcal{A}_{\text{train}}$ is the training set of action chunks and $\mathcal{C}_{\text{train}}$ its encoding, so membership below always refers to the latter. To measure frequency, we record whether each generated prefix occurs among prefixes of the same length in $\mathcal{C}_{\text{train}}$, and whether the complete unconstrained greedy sequence matches the encoded target. To measure influence, we construct a trie from $\mathcal{C}_{\text{train}}$ and constrain greedy decoding according to

$$\widehat{c}_i = \arg \max_{\substack{v:\, \exists C \in \mathcal{C}_{\text{train}} \\ C_{1:i} = (\widehat{C}_{<i}, v)}} \text{logit}_i(v). \tag{14}$$

Every selected token therefore preserves a prefix encoded from training, and the complete generated sequence belongs to $\mathcal{C}_{\text{train}}$. Comparing constrained against unconstrained decoding under otherwise identical conditions isolates the effect of training-support membership.

C.4.1 Offline Analysis

Token mismatch frequency. Two conditions in the cohort used for the family-level validation diagnostics are degenerate: their tokenizers collapse the training set onto only two distinct token sequences, so the policy reproduces the encoded target trivially and rollout success is zero under every decoding rule. Excluding those two, the exact target-sequence match rate is at most 3.8%, so at least 96.2% of greedy predictions differ from their encoded targets. For every non-collapsed tokenizer, nearly all greedy predictions thus differ from their encoded targets. Figure 15(a) shows where they emerge within the autoregressive sequence, and Figure 15(b) resolves the exact-match rate by quantizer family and task.

Action-reconstruction error. Figure 15(c) reports how constraining decoding to the training support changes the validation action error, measured between the decoded prediction and the encoded target chunk. Training-support-constrained decoding increases validation action MAE in 64 of 66 tokenizer–task conditions and ties in the remaining 2. In the normalized raw-action L_1 units of ℓ, the task-average constrained-minus-unconstrained MAE is 0.0302 on Can, 0.0227 on Lift, and 0.0335 on Square. Restricting predictions to observed sequences worsens action accuracy in almost every condition. The two ties are the degenerate conditions above, whose greedy predictions already lie in $\mathcal{C}_{\text{train}}$, so the constraint is vacuous there and acts on every condition in which it can act.

C.4.2 Online Analysis

Figure 16 repeats both measurements during rollout, where the policy conditions on its own executed actions rather than on held-out expert chunks. Its prefix curves are grouped by quantizer family for each task; shaded regions are within-family min–max ranges across tokenizer designs. The forest panels retain all 22 tokenizer configurations and their paired intervals.

Token mismatch frequency. Figure 16(a) reports the task-specific rollout unobserved-prefix curves, which track how often the generated prefix is absent from $\mathcal{C}_{\text{train}}$ at each position of the sequence. As in the offline analysis, departures from the training support are not confined to a single quantizer family.

Rollout success. Constrained decoding reduces rollout success in 53 of 66 tokenizer–task conditions, improves it in 11, and ties in 2. The mean constrained-minus-unconstrained changes are -19.7, -14.7, and -17.6 percentage points on Can, Lift, and Square, respectively. The intervention therefore degrades both offline action accuracy and closed-loop performance for most configurations. The two ties are again the degenerate conditions, whose rollout success is zero under both decoding rules, so the constraint acts on every condition in which it can act.

Together, the offline and online diagnostics show that training-support membership is neither necessary nor sufficient for a good prediction: novel token compositions are common, and forcing predictions onto observed sequences lowers offline action accuracy and, in most conditions, rollout success.

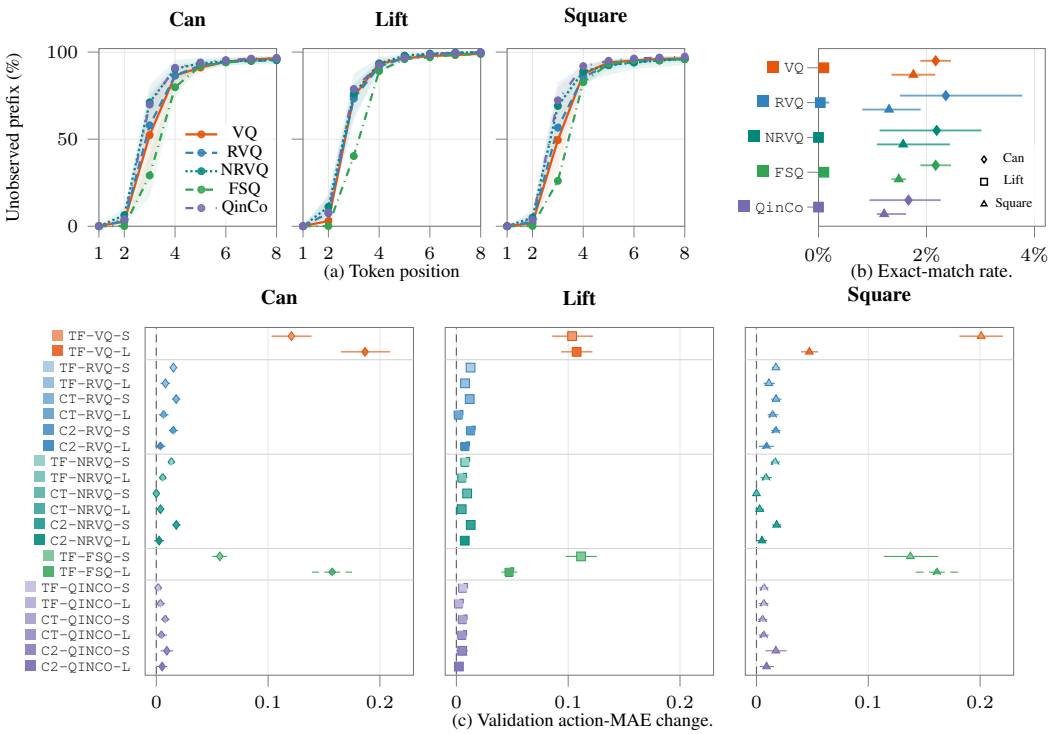


Figure 15: Held-out validation training-support diagnostics. (a) Accumulated unobserved-prefix rates: curves and bands show family means and min–max design ranges. (b) Unconstrained greedy exact-match rates: family/task means and min–max ranges over the same cohort. (c) Training-support-constrained minus unconstrained normalized raw-action L_1/MAE ℓ for every tokenizer with paired 95% intervals; positive values indicate worse prediction. Color and line style identify family; diamonds, squares, and triangles identify Can, Lift, and Square. Tokenizer colors and order match Appendix B.1.

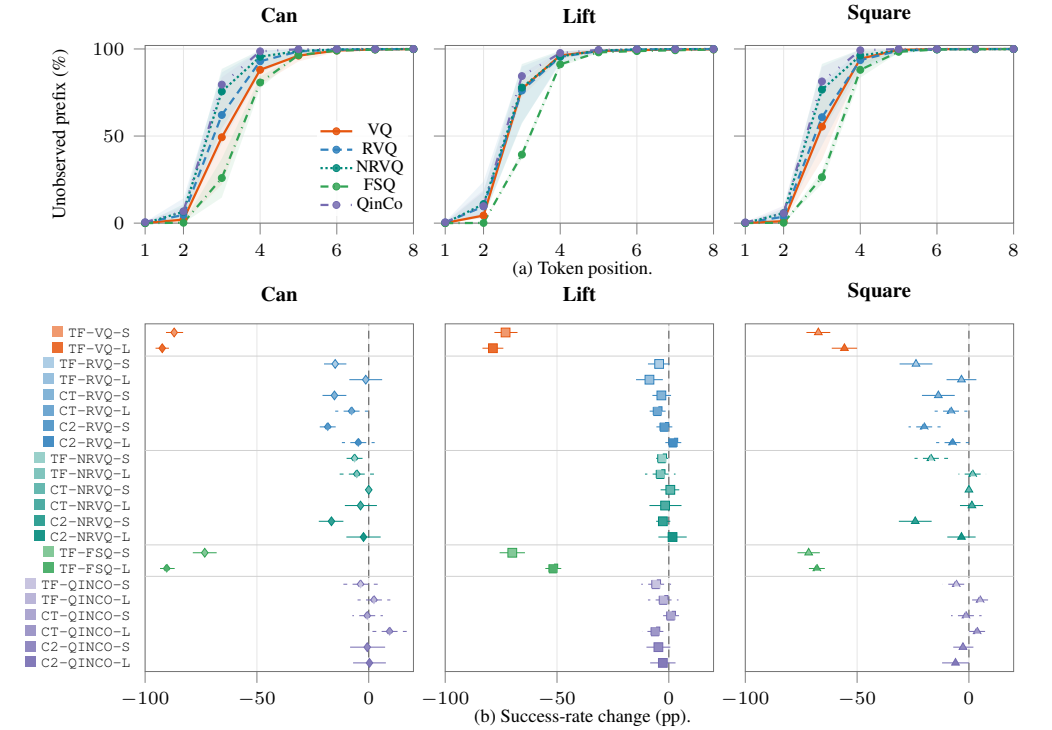


Figure 16: Rollout training-support diagnostics using greedy policies at the checkpoint with the best teacher-forced validation accuracy, $H_{\text{exec}} = 8$, and 300 episodes per condition. (a) Accumulated unobserved-prefix rates: family means and min–max design ranges. (b) Constrained minus unconstrained success for all 22 tokenizers with paired 95% intervals. Family styles, task markers, and tokenizer colors/order follow Figure 15.

C.5 One-Step Error Decomposition

We study the expected action-chunk error of a tokenized autoregressive policy,

$$J_E = \mathbb{E}_{(O,A)\sim p_{\mathrm{data}}} \mathbb{E}_{\widehat{C}\sim p_\psi^{\mathrm{dec}}(\cdot|O)} \left[\|\mathcal{D}(\widehat{C}) - A\| \right]. \tag{15}$$

Here (O, A) is an observation–action pair from the offline expert data distribution p_{data}: A is the expert action chunk recorded for observation O. We have access only to a fixed dataset of such pairs, with no additional expert queries. The deployed policy generates $\widehat{C}$ from O, independently of the recorded A conditional on O, and decodes it into an action chunk. Our objective is to decompose J_E and identify which contributions are controlled by reconstruction and teacher-forced training.

Training objectives and decoding rules. We use the tokenization map $\mathcal{T}$, detokenization map $\mathcal{D}$, and autoregressive model p_ψ from Section 3. An action chunk has shape $H \times d_a$, and its target encoding is $C^\star = \mathcal{T}(A) = (c_1^\star, \ldots, c_L^\star)$. Errors are measured by the MAE of Section 4.1: we write $\|X\| := \|X\|_1/(Hd_a)$ for the flattened chunk, so that $\ell(A, \widehat{A}) = \|\widehat{A} - A\|$. Let $p_{\mathcal{A}}$ be the action marginal of p_{data} and define the reconstruction error $r(A) = \|\mathcal{D}(\mathcal{T}(A)) - A\|$. The reconstruction objective and teacher-forced negative log-likelihood are

$$\begin{aligned} \mathcal{L}_{\mathrm{tok}} &= \mathbb{E}_{A\sim p_{\mathcal{A}}}[r(A)], \\ \mathcal{L}_{\mathrm{AR}} &= -\mathbb{E}_{(O,A)\sim p_{\mathrm{data}}}[\log p_\psi(C^\star \mid O)] \\ &= -\mathbb{E}_{(O,A)\sim p_{\mathrm{data}}} \left[\sum_{i=1}^{L} \log p_\psi(c_i^\star \mid c_{<i}^\star, O) \right]. \end{aligned} \tag{16}$$

All logarithms are natural. The reconstruction objective and J_E use the same action marginal; the analysis also applies when p_{data} is the empirical distribution of the recorded dataset. Quantizer-specific auxiliary losses do not enter the argument.

The model distribution p_ψ and deployed distribution p_ψ^{dec} need not coincide. We consider three decoding rules: *sampling*, for which $p_\psi^{\mathrm{dec}} = p_\psi$; *mode-seeking*, which returns the token-wise greedy sequence or an exact sequence mode, including beam search when it returns that mode; and *truncated sampling*, which restricts each conditional distribution to a set containing its most probable token and renormalizes, as in top-k or nucleus sampling. For mode-seeking, p_ψ^{dec} is the point mass at the returned sequence. The decomposition below holds for any deployed distribution; these three rules determine the constants in the mismatch-probability bound.

Error decomposition. For each recorded pair, define

$$\mathsf{M} = \{\widehat{C} = C^\star\}, \qquad p_{\mathrm{m}}(O, A) = p_\psi^{\mathrm{dec}}(C^\star \mid O), \qquad \bar{p} = \mathbb{E}[p_{\mathrm{m}}(O, A)]. \tag{17}$$

Unless specified otherwise, expectations and probabilities are over $(O, A) \sim p_{\mathrm{data}}$ and the deployed token draw. Write $\overline{\mathcal{E}}_= = \mathbb{E}[\mathbf{1}_{\mathsf{M}}\|\mathcal{D}(\widehat{C}) - A\|]$ for the matched contribution and $M_{\neq} = \mathbb{E}[\|\mathcal{D}(\widehat{C}) - A\| \mid \mathsf{M}^c]$ for the conditional mismatch error.

Proposition C.1 (One-step error decomposition)**.** *The expected decoded error satisfies*

$$J_E = \overline{\mathcal{E}}_= + (1 - \bar{p})M_{\neq}, \qquad \overline{\mathcal{E}}_= \le \mathcal{L}_{\mathrm{tok}}. \tag{18}$$

For the three decoding rules above,

$$1 - \bar{p} \le c_{\mathrm{dec}}\mathcal{L}_{\mathrm{AR}}, \qquad c_{\mathrm{dec}} = \begin{cases} 1, & \textit{sampling}, \\ 1/\log 2, & \textit{mode-seeking}, \\ 1 + 1/(2\log 2), & \textit{truncated sampling}. \end{cases} \tag{19}$$

Consequently,

$$J_E \le \mathcal{L}_{\mathrm{tok}} + c_{\mathrm{dec}}\mathcal{L}_{\mathrm{AR}}M_{\neq}. \tag{20}$$

Proof. On M, the decoder receives $C^\star$ and its error is exactly $r(A)$. Splitting the expectation over M and M^c gives the identity in Equation 18. Moreover, $0 \le p_{\mathrm{m}}(O, A) \le 1$ gives

$$\overline{\mathcal{E}}_= = \mathbb{E}[p_{\mathrm{m}}(O, A)r(A)] \le \mathbb{E}_{A\sim p_{\mathcal{A}}}[r(A)] = \mathcal{L}_{\mathrm{tok}}. \tag{21}$$

The probability bound is established in Lemma C.2 below. Substituting both bounds into the decomposition proves Equation 20. If there are no mismatches, their contribution is zero. □

Mismatch probability. Teacher forcing controls the model likelihood of the target sequence. The following lemma transfers that control to the deployed match probability.

Lemma C.2 (Match probability under decoding)**.** *Equation 19 holds for each of the three decoding rules. For sampling, the sharper bound is* $1 - \bar{p} \le 1 - e^{-\mathcal{L}_{\mathrm{AR}}}$.

Proof. For a fixed pair (O, A), write $s = p_\psi(C^\star \mid O)$. Under sampling, $p_{\mathrm{m}} = s$, so Jensen's inequality yields

$$\bar{p} = \mathbb{E}[s] \ge \exp(\mathbb{E}[\log s]) = e^{-\mathcal{L}_{\mathrm{AR}}} \ge 1 - \mathcal{L}_{\mathrm{AR}}. \tag{22}$$

For the other rules, observe that $s > 1/2$ implies that every target-prefix conditional probability exceeds $1/2$, since their product is s and each factor is at most one. Thus every target token is the unique most probable token at its prefix, and $C^\star$ is also the unique sequence mode. Markov's inequality gives

$$\Pr(s \le 1/2) = \Pr(-\log s \ge \log 2) \le \frac{\mathcal{L}_{\mathrm{AR}}}{\log 2}. \tag{23}$$

Under mode-seeking, $s > 1/2$ therefore implies $p_{\mathrm{m}} = 1$, giving $1 - \bar{p} \le \Pr(s \le 1/2) \le \mathcal{L}_{\mathrm{AR}}/\log 2$.

Under truncated sampling, $s > 1/2$ ensures that every target token is retained along its prefix. Renormalization can only increase its conditional probability, so $p_{\mathrm{m}} \ge s$ on this event. Consequently,

$$\begin{aligned} \bar{p} &\ge \mathbb{E}[s\mathbf{1}_{\{s>1/2\}}] \ge \mathbb{E}[s] - \tfrac{1}{2}\Pr(s \le 1/2) \\ &\ge e^{-\mathcal{L}_{\mathrm{AR}}} - \frac{\mathcal{L}_{\mathrm{AR}}}{2\log 2} \ge 1 - \left(1 + \frac{1}{2\log 2}\right)\mathcal{L}_{\mathrm{AR}}. \end{aligned} \tag{24}$$

This proves all three cases. □

Mismatch severity. To examine $M_{\neq}$, define the encoded action support and the action preimage of a sequence:

$$\begin{aligned} \mathcal{C}_{\mathrm{seen}} &= \{C : \Pr_{A \sim p_{\mathcal{A}}}(\mathcal{T}(A) = C) > 0\}, \\ \mathcal{F}(C) &= \{A' \in \operatorname{supp} p_{\mathcal{A}} : \mathcal{T}(A') = C\}. \end{aligned} \tag{25}$$

For the empirical data distribution, $\mathcal{C}_{\mathrm{seen}}$ is precisely the training support $\mathcal{C}_{\mathrm{train}}$. The mismatch event splits into

$$\begin{aligned} \mathsf{B}_{\mathrm{in}} &= \{\widehat{C} \ne C^\star,\ \widehat{C} \in \mathcal{C}_{\mathrm{seen}}\}, \\ \mathsf{B}_{\mathrm{out}} &= \{\widehat{C} \notin \mathcal{C}_{\mathrm{seen}}\}. \end{aligned} \tag{26}$$

Writing $\alpha = \Pr(\mathsf{B}_{\mathrm{in}} \mid \mathsf{M}^c)$ and $M_{\mathrm{in}}, M_{\mathrm{out}}$ for their conditional decoded errors gives

$$M_{\neq} = \alpha M_{\mathrm{in}} + (1 - \alpha) M_{\mathrm{out}}. \tag{27}$$

When $\Pr(\mathsf{M}^c) = 0$, set $\alpha = 0$ and omit the mismatch terms.

In-support mismatches. For $\widehat{C} \in \mathcal{C}_{\mathrm{seen}}$, choose $A' \in \mathcal{F}(\widehat{C})$. Then $\mathcal{D}(\widehat{C}) = \mathcal{D}(\mathcal{T}(A'))$, and the triangle inequality gives

$$\begin{aligned} \|\mathcal{D}(\widehat{C}) - A\| &\le b(\widehat{C}, A), \\ b(C, A) &:= \inf_{A' \in \mathcal{F}(C)} [r(A') + \|A' - A\|] \le R_{\mathrm{rec}} + D_{\mathcal{A}}, \end{aligned} \tag{28}$$

where

$$R_{\mathrm{rec}} = \sup_{A \in \operatorname{supp} p_{\mathcal{A}}} r(A), \qquad D_{\mathcal{A}} = \sup_{A, A' \in \operatorname{supp} p_{\mathcal{A}}} \|A' - A\|.$$

The last inequality follows because each $C \in \mathcal{C}_{\mathrm{seen}}$ has a preimage in the support, where $r(A') \le R_{\mathrm{rec}}$. Taking the conditional expectation yields

$$M_{\mathrm{in}} \le \mathbb{E}[b(\widehat{C}, A) \mid \mathsf{B}_{\mathrm{in}}] \le R_{\mathrm{rec}} + D_{\mathcal{A}}.$$

The preimage bound depends on how close a corresponding action A' is to the current target A; the diameter bound can be much looser. Moreover, reconstruction training controls the average $r(A)$,

rather than R_{rec}. Without a lower bound on the probability of individual actions, a small average does not imply a small worst-case error; it gives only the tail bound

$$\Pr_{A\sim p_{\mathcal{A}}}(r(A) > \varepsilon) \le \mathcal{L}_{\mathrm{tok}}/\varepsilon.$$

Off-support mismatches. On $\mathsf{B}_{\mathrm{out}}$, the predicted sequence has zero probability under the encoded action distribution. The reconstruction objective therefore places no direct constraint on its decoded action, while the teacher-forced objective does not evaluate $\mathcal{D}$ at all. To make this precise, suppose $\Pr(\mathsf{B}_{\mathrm{out}}) > 0$ and impose no additional range or regularity constraint on the detokenization map. For a vector u with $\|u\| = 1$ and $R > 0$, define

$$\mathcal{D}_R(C) = \begin{cases} \mathcal{D}(C), & C \in \mathcal{C}_{\mathrm{seen}}, \\ \mathcal{D}(C) + Ru, & C \notin \mathcal{C}_{\mathrm{seen}}. \end{cases} \tag{29}$$

This leaves $\mathcal{L}_{\mathrm{tok}}$, $\mathcal{L}_{\mathrm{AR}}$, and the deployed token distribution unchanged. For every off-support prediction, the reverse triangle inequality gives $\|\mathcal{D}_R(\widehat{C}) - A\| \ge R - \|\mathcal{D}(\widehat{C}) - A\|$, so its conditional expected error grows without bound as $R \to \infty$. Thus no finite upper bound on M_{out} follows from the two loss values alone. A bounded output range or an additional regularity condition can supply a separate bound.

This argument applies to any decoding rule that assigns positive probability to $\mathsf{B}_{\mathrm{out}}$. With finite-logit softmax sampling, that probability is positive whenever $\mathcal{C}_{\mathrm{seen}}$ is a proper subset of the token space. Mode-seeking or truncation may avoid the event, but do not otherwise change the absence of decoder supervision on it.

Combined bound. Combining Proposition C.1 with the two mismatch branches gives

$$J_E \le \mathcal{L}_{\mathrm{tok}} + c_{\mathrm{dec}}\mathcal{L}_{\mathrm{AR}}\left[\alpha(R_{\mathrm{rec}} + D_{\mathcal{A}}) + (1-\alpha)M_{\mathrm{out}}\right]. \tag{30}$$

Reconstruction controls the matched contribution, while teacher forcing controls mismatch frequency. The remaining error depends on what mismatched sequences decode to: even an in-support sequence may represent an action far from the current target, and off-support decoder outputs are not constrained by either objective. The in-support bound also requires worst-case reconstruction and action-range information beyond the average loss. Changing the decoding rule preserves this decomposition, although it can change both branch probabilities and conditional errors. These statements concern one-step error under the offline expert data distribution.

Benign and uncontrolled mismatches. Motivated by the action-noise tolerance observed in Appendix C.1.2, assume each observation–action pair (O, A) admits a local region $\mathcal{R}(O, A)$ around A whose actions preserve acceptable execution at O. Let $\rho \ge 0$ be a local tolerance radius bound such that $\|a - A\| \le \rho$ for every $a \in \mathcal{R}(O, A)$. An in-support token mismatch is *benign* when it admits a reconstruction target $A' \in \mathcal{F}(\widehat{C}) \cap \mathcal{R}(O, A)$ with $r(A') \le R_{\mathrm{rec}}$. Since $\mathcal{D}(\widehat{C}) = \mathcal{D}(\mathcal{T}(A'))$, the triangle inequality then gives

$$\|\mathcal{D}(\widehat{C}) - A\| \le r(A') + \|A' - A\| \le R_{\mathrm{rec}} + \rho.$$

Here, benign refers to the local compatibility of the reconstruction target; the decoded action additionally incurs reconstruction error. Retain $\alpha = \Pr(\mathsf{B}_{\mathrm{in}} \mid \mathsf{M}^c)$ from Equation 27 and let β be the probability of a benign in-support mismatch conditional on M^c, so that $0 \le \beta \le \alpha$. The remaining in-support mismatches and all off-support mismatches form the *uncontrolled* branch: they lack this local compatibility guarantee, which does not imply execution failure. Splitting the in-support contribution in Equation 30 gives

$$J_E \le \underbrace{\mathcal{L}_{\mathrm{tok}}}_{\text{matched}} + c_{\mathrm{dec}}\mathcal{L}_{\mathrm{AR}}\Big[\underbrace{\beta(R_{\mathrm{rec}} + \rho)}_{\text{benign}} + \overbrace{\underbrace{(\alpha-\beta)(R_{\mathrm{rec}} + D_{\mathcal{A}})}_{\text{uncontrolled}} + \overbrace{(1-\alpha)M_{\mathrm{out}}}^{\text{off-support}}}^{\text{in-support}}\Big]. \tag{31}$$

The benign branch defined above identifies a sufficient condition for local compatibility, rather than exhaustively characterizing all acceptable predictions. The empirical training support records the final tokenizer's encodings of the training chunks and need not include every sequence encountered during tokenizer training as token assignments evolve. Such additional exposure may allow

sequences outside the empirical support to decode to locally acceptable actions. These predictions therefore remain in the off-support contribution M_{out}, whose value can be small in practice. However, this, in general, provides no guarantee on the action error, but suggests possible mechanisms that explain the phenomena that we study in Section 4.3 and Appendix C.4.

Remark: connection to trajectory-level error. Our analysis isolates the sources of action-chunk prediction error under expert observations. For how prediction errors accumulate during execution, we refer readers to Zhang et al. (2025), who relate squared prediction error on expert demonstrations to trajectory-level error under assumptions on system stability, the policy class, and chunked execution. Their results provide a complementary perspective on when accurate action-chunk prediction can translate into accurate trajectories.

D Additional Method Details

D.1 Token Balance Regularization: Balanced Feature Learning and Irreducible Policy Loss

This section motivates TBR through the geometry of the token targets and the principle of balanced feature learning (BFL). We first analyze the irreducible loss of the teacher-forced loss in policy learning (Step 1) and bound it using token margins (Step 2). Because tokenization must also support action reconstruction (Step 3), we then examine a linear model in which noise-subspace sensitivity reduces margins without changing the tokenizer's token encoding (Step 4). Under mild assumptions, we show that reconstruction weakly penalizes this sensitivity (Step 5), and that activation preconditioning, as one concrete BFL mechanism, removes it more rapidly than gradient descent and reduces irreducible policy log-loss under a controlled comparison (Step 6).

D.1.1 Irreducible Policy Loss and Token Geometry

Step 1: Irreducible policy log-loss. Let $(O, A) \sim p_{\mathrm{data}}$ be an observation–action pair as in Appendix C.5. Let τ be a deterministic token-assignment map, $C = \tau(A)$ the resulting token target, and $p_\psi(C \mid O)$ the policy distribution parameterized by ψ. Probabilities and expectations are over (O, A), $\mathbb{H}$ denotes (conditional) entropy with natural logarithms, and J_E, $\mathcal{L}_{\mathrm{tok}}$, $\mathcal{L}_{\mathrm{AR}}$, $M_{\neq}$, c_{dec} are as in Proposition C.1. That proposition gives

$$J_E \le \mathcal{L}_{\mathrm{tok}} + c_{\mathrm{dec}}\, \mathcal{L}_{\mathrm{AR}}\, M_{\neq}, \tag{32}$$

where reconstruction training controls $\mathcal{L}_{\mathrm{tok}}$ and Token Perturbation targets $M_{\neq}$. We next isolate the irreducible component of $\mathcal{L}_{\mathrm{AR}} := \mathbb{E}_{(O,C)}[-\log p_\psi(C \mid O)]$, which is the sequence-level negative log-likelihood. If the training loss is averaged over the L token positions, the entropy and KL terms below should also be divided by L.

Proposition D.1 (Irreducible part of the teacher-forced loss)**.** *Let $P(C \mid O)$ denote the true conditional distribution of C and* KL *the Kullback–Leibler divergence. For every policy p_ψ,*

$$\mathcal{L}_{\mathrm{AR}} = \underbrace{\mathbb{H}(C \mid O)}_{\textit{tokenizer and data}} + \underbrace{\mathbb{E}_O\, \mathrm{KL}\big(P(C \mid O) \,\big\|\, p_\psi(C \mid O)\big)}_{\textit{policy}} \ge \mathbb{H}(C \mid O), \tag{33}$$

with equality if and only if $p_\psi(\cdot \mid O) = P(\cdot \mid O)$ almost surely.

Proof. Taking expectations in $-\log p_\psi(C \mid O) = -\log P(C \mid O) + \log(P(C \mid O)/p_\psi(C \mid O))$ gives Equation 33. The KL divergence is nonnegative and vanishes exactly when the two conditional distributions agree almost surely. □

Thus $\mathbb{H}(C \mid O)$ is the *irreducible log-loss* shared by all policies trained on the same token targets. Changing the tokenizer can directly affect this loss; here we focus on the first term in Equation 33—the conditional token entropy.

Step 2: Token margins bound the irreducible loss. Consider first a single token $c = \tau(A) \in \{1, \dots, K\}$, where $K \ge 2$ is the codebook size. For a sequence $C = (c_1, \dots, c_L)$ of L tokens, let $c_{<i} = (c_1, \dots, c_{i-1})$ denote the preceding tokens. The chain rule $\mathbb{H}(C \mid O) = \sum_{i=1}^{L} \mathbb{H}(c_i \mid O, c_{<i})$ reduces the sequence setting to the single-token case by replacing O with $(O, c_{<i})$. For each observation, choose a measurable reference action $g(O)$; the bounds below hold for any function g of O and the conditional mean $g(O) := \mathbb{E}[A \mid O]$ is one natural choice. This reference serves as a geometric witness for a token label. We measure the margins from the reference action, around which the actions recorded at O form a cloud, and call it the *center* (Figure 17). We write A for the random action chunk and lowercase a for a fixed action chunk, following the usual convention for a random variable and its values. Define

$$\begin{aligned}
\mathcal{C}(O) &= \{a : \tau(a) = \tau(g(O))\} && \text{(cell of the center)},\\
m(O) &= \inf\{\|a - g(O)\|_2 : a \notin \mathcal{C}(O)\} && \text{(margin)},\\
\mathbb{B}(O) &= \{a : \|a - g(O)\|_2 < m(O)\} && \text{(margin ball)},\\
p_{\neq}(O) &= \Pr[A \notin \mathcal{C}(O) \mid O] && \text{(label inconsistency)}.
\end{aligned} \tag{34}$$

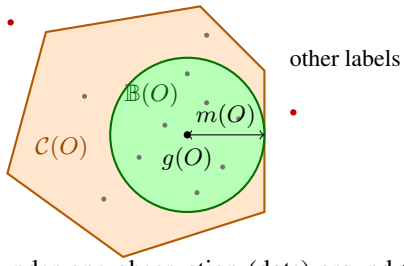


Figure 17: Actions recorded under one observation (dots) around their center $g(O)$. The margin ball $\mathbb{B}(O)$ lies in the cell $\mathcal{C}(O)$ of the center; samples outside $\mathcal{C}(O)$ (red) carry another label and lie outside $\mathbb{B}(O)$.

Thus $\mathcal{C}(O)$ is the token cell containing the reference action $g(O)$, and $m(O)$ is the distance from $g(O)$ to the nearest cell boundary. The label inconsistency $p_{\neq}(O)$ is the probability that a demonstration action A recorded at O receives a different token from its center $g(O)$. When $0 < p_{\neq}(O) < 1$, demonstrations of the same observation disagree on the token, so no policy can predict it from O alone. Let $h_2(p) = -p\log p - (1-p)\log(1-p)$ denote binary entropy.

Proposition D.2 (Irreducible loss and label inconsistency)**.** *Let $\mathbb{H}^{\neq}(O) := \mathbb{H}(c \mid O, A \notin \mathcal{C}(O))$ be the entropy among labels outside the center cell. Then*

$$\mathbb{H}(c \mid O) = \mathbb{E}\big[\underbrace{h_2(p_{\neq}(O))}_{\textit{whether}} + \underbrace{p_{\neq}(O)\,\mathbb{H}^{\neq}(O)}_{\textit{where}}\big] \;\le\; \mathbb{E}\big[h_2(p_{\neq}(O)) + p_{\neq}(O)\log(K-1)\big]. \tag{35}$$

Equality holds when, given $A \notin \mathcal{C}(O)$, the label is uniform on the other $K-1$ values, and always when $K = 2$.

Proof. Fix $O = o$ and let $I = \mathbf{1}\{A \notin \mathcal{C}(o)\} = \mathbf{1}\{\tau(A) \neq \tau(g(o))\} = \mathbf{1}\{c \neq \tau(g(o))\}$, which is a deterministic function of c. Because I is Bernoulli with mean $p_{\neq}(o)$ and $c = \tau(g(o))$ when $I = 0$, the chain rule gives $\mathbb{H}(c \mid o) = \mathbb{H}(I \mid o) + \mathbb{H}(c \mid I, o) = h_2(p_{\neq}(o)) + p_{\neq}(o)\mathbb{H}^{\neq}(o)$. Finally, $\mathbb{H}^{\neq}(o) \le \log(K-1)$, with equality for a uniform distribution over the other labels; when $K = 2$, equality always holds. □

The *whether* term $h_2(p_{\neq}(O))$ is the uncertainty about whether a demonstration keeps the token of its center. The *where* term $p_{\neq}(O)\mathbb{H}^{\neq}(O)$ is the uncertainty about which of the other $K-1$ tokens it receives when it does not, and it is at most $p_{\neq}(O)\log(K-1)$. The margin controls the exit probability $p_{\neq}(O)$.

Proposition D.3 (Margin bound)**.** *For every O, $\mathbb{B}(O) \subseteq \mathcal{C}(O)$, and hence*

$$p_{\neq}(O) \le \Pr\big[\|A - g(O)\|_2 \ge m(O) \mid O\big], \tag{36}$$

with equality when $\mathcal{C}(O)$ is a ball centered at $g(O)$.

Proof. By the definition of $m(O)$, every a with $\|a - g(O)\|_2 < m(O)$ belongs to $\mathcal{C}(O)$. Consequently, $\{A \notin \mathcal{C}(O)\} \subseteq \{\|A - g(O)\|_2 \ge m(O)\}$. Taking conditional probabilities gives Equation 36. □

Let $F_K(p) := h_2(p) + p\log(K-1)$ and define its nondecreasing envelope by $\overline{F}_K(q) := \max_{0\le p\le q} F_K(p) = F_K\big(\min\{q, 1-\frac{1}{K}\}\big), 0 \le q \le 1$. Writing $q(O) = \Pr\big[\|A - g(O)\|_2 \ge m(O) \mid O\big]$, we have $\mathbb{H}(c \mid O) \le \mathbb{E}\big[F_K(p_{\neq}(O))\big] \le \mathbb{E}\big[\overline{F}_K(q(O))\big]$. For a fixed conditional action distribution and reference map g, this upper bound is nonincreasing as the margins grow. This implies that the smaller token margin results in a larger upper bound on the token entropies.

Step 3: Predictability and reconstruction. Tokenization should balance predictability and reconstruction. The margin bound suggests that greater separation between conditional action variation and token boundaries can improve predictability, while a sufficiently low reconstruction error is required (Section 4.1). A trivial case is that a tokenizer can have constant token encoding for all

actions, resulting in zero token entropies while action information is totally lost. We therefore fix K and the encoded content, and ask whether reducing sensitivity in orthogonal directions with respect to the token margin boundary can improve token predictability while preserving reconstruction.

D.1.2 A Linear Model Motivating Balanced Feature Learning

Assumptions and scope. We now list the explicit distributional and optimization assumptions. **(A1) Conditional-center subspace model.** Let $A = g(O) + \xi$, where $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$ is independent of O and $\sigma > 0$ is the noise scale. Let $\mathcal{S} \subsetneq \mathbb{R}^d$ be the smallest subspace containing $g(O)$ almost surely, and let $\mathcal{S}^\perp$ denote its orthogonal complement. The low-dimensional assumption concerns the conditional centers, not the noisy actions: when second moments exist, $\mathbb{E}[AA^\top] = \mathbb{E}[g(O)g(O)^\top] + \sigma^2 I_d \succ 0$. Let $P_\mathcal{S}$ and $P_\perp$ be the orthogonal projections onto $\mathcal{S}$ and $\mathcal{S}^\perp$, respectively.

(A2) Binary linear tokenizer. Let $c = \mathbf{1}\{w^\top A > \beta\}$, where $w \neq 0$ is a row of the first encoder layer, $\beta \in \mathbb{R}$ is its threshold, and $\mathbf{1}\{\cdot\}$ is the indicator function. This binary case permits an exact entropy calculation; the general bounds in Steps 1–3 already apply to arbitrary finite codebooks.

Step 4: Noise-subspace sensitivity reduces token margins. The subspace $\mathcal{S}$ is a property of the data alone: along $\mathcal{S}$ the centers vary, and along $\mathcal{S}^\perp$, the *pure-noise directions*, actions carry only residual noise. Under (A2), $m_w(a) := |w^\top a - \beta|/\|w\|_2$ is the distance from a to the boundary, so $m(O) = m_w(g(O))$. Let Φ be the standard normal distribution function.

Proposition D.4 (Exact irreducible loss)**.** *Under (A1)–(A2),*

$$\Pr[c = 1 \mid O] = \Phi\Big(\frac{w^\top g(O) - \beta}{\sigma\|w\|_2}\Big), \qquad \mathbb{H}(c \mid O) = \mathbb{E}\Big[h_2\Big(\Phi\Big(\frac{m_w(g(O))}{\sigma}\Big)\Big)\Big], \tag{37}$$

and the integrand is strictly decreasing in $m_w(g(O))/\sigma$.

Proof. $w^\top A = w^\top g(O) + w^\top \xi$ with $w^\top \xi \sim \mathcal{N}(0, \sigma^2\|w\|_2^2)$ independent of O, which gives the first identity. Since $K = 2$, Proposition D.2 holds with equality, $\mathbb{H}(c \mid O) = \mathbb{E}[h_2(\Pr[c = 1 \mid O])]$, and $h_2(\Phi(x)) = h_2(\Phi(-x))$ is strictly decreasing in $|x|$. □

Decompose $w = w_\mathcal{S} + u$ with $w_\mathcal{S} \in \mathcal{S}$, the *signal-subspace* component, and $u \in \mathcal{S}^\perp$, the *noise-subspace* component.

Proposition D.5 (Removing *noise-subspace* sensitivity increases margins)**.** *Assume (A1)–(A2) and $w_\mathcal{S} \neq 0$, and let α be the angle between w and $w_\mathcal{S}$. Then*

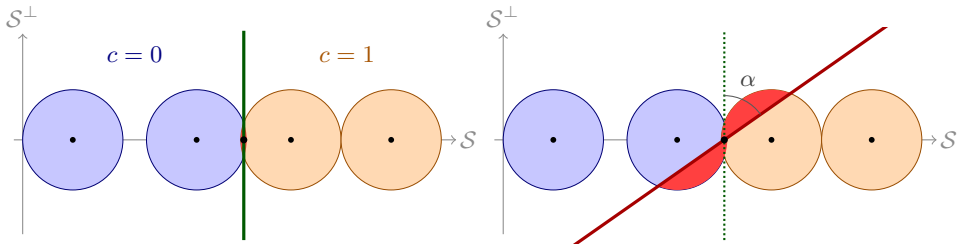
$$m_w(g(O)) = \frac{m_{w_\mathcal{S}}(g(O))}{\kappa}, \qquad \kappa := \sqrt{1 + \frac{\|u\|_2^2}{\|w_\mathcal{S}\|_2^2}} = \frac{1}{\cos\alpha} \geq 1. \tag{38}$$

Moreover, $\mathbf{1}\{w^\top g(O) > \beta\} = \mathbf{1}\{w_\mathcal{S}^\top g(O) > \beta\}$, so every center keeps its label, and $\mathbb{H}(c \mid O) = \mathbb{E}\big[h_2\big(\Phi\big(m_{w_\mathcal{S}}(g(O))/(\sigma\kappa)\big)\big)\big]$ is nondecreasing in $\|u\|_2$, strictly so if $\Pr[m_{w_\mathcal{S}}(g(O)) > 0] > 0$.

Proof. Since $g(O) \in \mathcal{S}$ and $u \perp \mathcal{S}$, $w^\top g(O) = w_\mathcal{S}^\top g(O)$; since $w_\mathcal{S} \perp u$, $\|w\|_2 = \kappa\|w_\mathcal{S}\|_2$ and $\cos\alpha = \|w_\mathcal{S}\|_2/\|w\|_2$. The last claim follows from Proposition D.4. □

The pair $(w_\mathcal{S}, \beta)$ determines where the boundary cuts $\mathcal{S}$, and hence what the token encodes. The component u changes sensitivity in $\mathcal{S}^\perp$ while leaving the token encoding for the reference actions unchanged (Figure 18). Holding $(w_\mathcal{S}, \beta)$ fixed therefore isolates the effect of noise-subspace sensitivity on token predictability.

Thus removing the noise-subspace component multiplies each center margin by κ and reduces conditional token entropy under the condition of Proposition D.5.



(a) $u = 0$ (b) $u \neq 0$

Figure 18: The linear example in two dimensions, where $\mathcal{S}$ is a line. Discs are the action clouds of four observations. (a) $u = 0$. (b) $u \neq 0$ tilts the boundary by α about its cut with $\mathcal{S}$: no center changes side, every margin shrinks by $\kappa = 1/\cos\alpha$, and more samples (red) receive a label different from that of their center.

Step 5: Reconstruction weakly penalizes the wasted weight. Step 4 identified a component of the encoder weight that reduces token margins without changing the labels of the action-cloud centers. We now specify a reconstruction-guided model of encoder training, in which we separate the choice of target codes from the optimization that fits the encoder to those codes. The resulting quadratic objective is exact for this regression problem.

(A3) Fixed-target code regression. Fix a finite set of code embeddings $z_1, \dots, z_K \in \mathbb{R}^r$ and a fixed decoder $D_\star$ whose reconstruction points satisfy

$$r_j := D_\star(z_j) \in \mathcal{S}, \qquad j = 1, \dots, K. \tag{39}$$

Choose the target assignment and target embedding as

$$C^\dagger(A) := \arg\min_j \|P_{\mathcal{S}} A - r_j\|_2^2, \qquad Z^\dagger(A) := z_{C^\dagger(A)}, \tag{40}$$

During the optimization comparison, the decoder, embeddings, and target mapping $A \mapsto Z^\dagger(A)$ are fixed. The linear encoder is trained by

$$\mathcal{L}_{\text{code}}(W) := \frac{1}{2}\mathbb{E}\|WA - Z^\dagger(A)\|_2^2, \qquad W_\star := \arg\min_W \mathcal{L}_{\text{code}}(W). \tag{41}$$

For the binary example, scalar target embeddings ($r = 1$) suffice; the matrix notation also permits the feature-wise optimization calculation below. The range condition in Equation 39 is motivated by optimal reconstruction for assignments based only on $P_{\mathcal{S}}A$. Indeed, under (A1), $P_\perp A$ is zero-mean and independent of $P_{\mathcal{S}}A$. For any fixed target assignment, the squared-error-optimal reconstruction minimizer gives $r_j \in \mathcal{S}$, which incurs no loss of optimality within this class of assignments. We therefore model encoder training by regressing WA onto the corresponding embedding $Z^\dagger(A)$, using code regression as a quadratic surrogate for action reconstruction.

Lemma D.6 (Regression geometry and the noise subspace)**.** *Under (A1) and (A3), $C^\dagger$ minimizes squared action-reconstruction error among assignments to the fixed reconstruction points $r_1, \dots, r_K$, and*

$$\mathbb{E}\big[Z^\dagger(P_\perp A)^\top\big] = 0. \tag{42}$$

Moreover, with $\Sigma = \mathbb{E}[AA^\top]$,

$$\Sigma = \mathbb{E}[g(O)g(O)^\top] + \sigma^2 I, \qquad \Sigma P_\perp = \sigma^2 P_\perp. \tag{43}$$

The eigenvalues on $\mathcal{S}$ exceed σ^2, whereas those on $\mathcal{S}^\perp$ equal σ^2. The unique regression optimum satisfies

$$W_\star = \mathbb{E}[Z^\dagger A^\top]\Sigma^{-1}, \qquad W_\star P_\perp = 0, \tag{44}$$

and the objective admits the exact decomposition

$$\mathcal{L}_{\text{code}}(W) = \frac{1}{2}\mathbb{E}\|(W - W_\star)A\|_2^2 + \mathcal{L}_{\text{code}}(W_\star), \qquad G := \nabla_W \mathcal{L}_{\text{code}}(W) = (W - W_\star)\Sigma. \tag{45}$$

Proof. For every action a and reconstruction point $r_j \in \mathcal{S}$, orthogonality gives

$$\|a - r_j\|_2^2 = \|P_{\mathcal{S}} a - r_j\|_2^2 + \|P_\perp a\|_2^2.$$

The second term is independent of j, so the assignment in Equation 40 is pointwise optimal for the fixed decoder and depends only on $P_{\mathcal{S}} A$. Under (A1), $P_\perp A = P_\perp \xi$ is zero-mean and independent of $P_{\mathcal{S}} A = g(O) + P_{\mathcal{S}} \xi$. Thus $Z^\dagger$ is independent of $P_\perp A$, which proves Equation 42.

Independence of ξ and O gives the expression for Σ. Since $\mathbb{E}[g(O)g(O)^\top]$ has range $\mathcal{S}$, its restriction to $\mathcal{S}$ is positive definite, and it vanishes on $\mathcal{S}^\perp$. This proves the spectral statements. In particular, $\Sigma \succ 0$.

The least-squares normal equation is

$$\mathbb{E}[(W_\star A - Z^\dagger)A^\top] = 0.$$

It yields the stated formula for $W_\star$. Since $\Sigma^{-1} P_\perp = \sigma^{-2} P_\perp$, we also obtain

$$W_\star P_\perp = \sigma^{-2} \mathbb{E}[Z^\dagger (P_\perp A)^\top] = 0.$$

Finally, expand $WA - Z^\dagger = (W - W_\star)A + (W_\star A - Z^\dagger)$. The normal equation makes the expected cross term zero, giving Equation 45. Differentiation gives the gradient. □

For two distinct reconstruction points $r_0, r_1 \in \mathcal{S}$, with ties assigned to code 0, the target assignment is itself a linear binary token:

$$C^\dagger(A) = \mathbf{1}\left\{ (r_1 - r_0)^\top A > \frac{\|r_1\|_2^2 - \|r_0\|_2^2}{2} \right\}. \tag{46}$$

Thus no pure-noise direction is needed for an optimal assignment to these reconstruction points. The least-squares encoder need not interpolate the target embeddings.

Proposition D.7 (Different sensitivities of code loss and conditional entropy)**.** *Assume (A1)–(A3). For one encoder row and its target coordinate $Z_\ell^\dagger$, write*

$$\ell_{\text{code}}(w) := \frac{1}{2}\mathbb{E}(w^\top A - Z_\ell^\dagger)^2, \qquad w = w_{\mathcal{S}} + u, \quad 0 \neq w_{\mathcal{S}} \in \mathcal{S}, \quad u \in \mathcal{S}^\perp.$$

Then

$$\ell_{\text{code}}(w) - \ell_{\text{code}}(w_{\mathcal{S}}) = \frac{1}{2}\sigma^2 \|u\|_2^2. \tag{47}$$

For the token $c_w = \mathbf{1}\{w^\top A > \beta\}$, let

$$\kappa = \sqrt{1 + \frac{\|u\|_2^2}{\|w_{\mathcal{S}}\|_2^2}}.$$

Suppose the signed center distance $s(O) = (w_{\mathcal{S}}^\top g(O) - \beta)/\|w_{\mathcal{S}}\|_2$ has a density f that is continuous and positive at zero. Holding $w_{\mathcal{S}}$, u, β, and the distribution of $g(O)$ fixed as $\sigma \to 0$, we have

$$\mathbb{H}(c_w \mid O) = C_0 \sigma\kappa + o(\sigma), \qquad C_0 = 2f(0)\int_0^\infty h_2(\Phi(y))\, dy > 0, \tag{48}$$

where C_0 is independent of u. Consequently,

$$\mathbb{H}(c_w \mid O) - \mathbb{H}(c_{w_{\mathcal{S}}} \mid O) = C_0 \sigma(\kappa - 1) + o(\sigma). \tag{49}$$

Proof. Let $w_\star$ be the corresponding row of $W_\star$. Lemma D.6 gives $w_\star \in \mathcal{S}$ and

$$\ell_{\text{code}}(w) = \frac{1}{2}(w - w_\star)^\top \Sigma (w - w_\star) + \text{const}.$$

In the decomposition $w - w_\star = (w_{\mathcal{S}} - w_\star) + u$, the cross term vanishes because $\Sigma u = \sigma^2 u$ and $u \perp \mathcal{S}$. This proves Equation 47.

The margin and entropy formulas of Step 4 give

$$\mathbb{H}(c_w \mid O) = \int_{\mathbb{R}} h_2\left(\Phi\left(\frac{|s|}{\sigma\kappa}\right)\right) f(s)\, ds = \sigma\kappa \int_{\mathbb{R}} h_2(\Phi(|y|)) f(\sigma\kappa y)\, dy.$$

To justify the limit, first restrict to $|s| < \delta$, where continuity bounds f and dominated convergence applies after the change of variables. On $|s| \geq \delta$, the integral divided by $\sigma\kappa$ is bounded by

$$\frac{h_2(\Phi(\delta/(\sigma\kappa)))}{\sigma\kappa} \longrightarrow 0.$$

Here the convergence follows from Gaussian tail decay. The limiting constant is therefore C_0 as stated above. Applying the same result with $u = 0$ and subtracting proves Equation 49. □

Implication of Step 5. For fixed nonzero noise-subspace weight, its excess code-regression loss is of order σ^2, whereas its excess conditional entropy is of order σ under the stated boundary-density condition. Encoders with the same signal-subspace component and threshold can therefore have nearly equal code-fitting losses but different token predictability.

Step 6: Balanced feature learning in the regression model. The fixed-target regression problem above has the exact gradient $G = (W - W_\star)\Sigma$. We now compare full-population gradient updates with fixed input second moment and fixed targets. Write

$$\Sigma = \sum_k \gamma_k v_k v_k^\top, \qquad \gamma_k > 0,$$

with orthonormal eigenvectors v_k, and define the feature error along v_k by

$$\epsilon_k(W) := \|(W - W_\star)v_k\|_2, \qquad \mathcal{L}_{\text{code}}(W) = \frac{1}{2}\sum_k \gamma_k \epsilon_k(W)^2 + \mathcal{L}_{\text{code}}(W_\star). \tag{50}$$

The objective weights each feature error by the energy of its input direction. More precisely, the *noise-subspace* weight matrix $U := WP_\perp$ satisfies

$$U = (W - W_\star)P_\perp, \qquad \|U\|_F^2 = \sum_{k:\,\gamma_k = \sigma^2} \epsilon_k(W)^2, \tag{51}$$

by Lemma D.6. Thus the *noise-subspace* weight component is the regression error on the lowest-energy input subspace. The gradient-descent and activation-preconditioning contraction calculations depend only on $G = (W - W_\star)\Sigma$, and hence apply directly to this model.

Definition D.8 (Learning-speed balance in the regression model)**.** *If an update rule maps weights W to updated weights W^+ and satisfies $\epsilon_k(W^+) = \theta_k\, \epsilon_k(W)$ with $\theta_k \in [0, 1)$, its* learning speed *along v_k is $1 - \theta_k$ and its* balance *is*

$$\text{bal} := \frac{\min_k(1 - \theta_k)}{\max_k(1 - \theta_k)} \in (0, 1]. \tag{52}$$

The rule is balanced *if* $\text{bal} = 1$.

Token Balance Regularization (TBR) refers to the procedure of applying balanced feature learning (BFL) to reduce disparity among directional learning speeds. Activation preconditioning is a concrete BFL mechanism analyzed below.

Lemma D.9 (Activation preconditioning improves balance)**.** *Assume (A1) and (A3). Gradient descent, $W^+ = W - \eta G$, and activation preconditioning with ridge parameter $\rho > 0$, $W^+ = W - \eta\, G(\Sigma + \rho I)^{-1}$, where I is the identity matrix, satisfy the following relations; the superscripts* GD *and* AP *denote the two update rules:*

$$\theta_k^{\text{GD}} = 1 - \eta\gamma_k, \quad \text{bal}^{\text{GD}} = \frac{\gamma_{\min}}{\gamma_{\max}}; \qquad \theta_k^{\text{AP}} = 1 - \frac{\eta\gamma_k}{\gamma_k + \rho}, \quad \text{bal}^{\text{AP}} = \frac{\gamma_{\min}}{\gamma_{\max}} \cdot \frac{\gamma_{\max} + \rho}{\gamma_{\min} + \rho} \geq \text{bal}^{\text{GD}}, \tag{53}$$

and $\text{bal}^{\text{AP}} \to 1$ *as* $\rho \to 0$.

Proof. Under (A3), $W^+ - W_\star = (W - W_\star)(I - \eta\Sigma)$ for gradient descent and $W^+ - W_\star = (W - W_\star)\big(I - \eta\Sigma(\Sigma + \rho I)^{-1}\big)$ for activation preconditioning; applying both sides to v_k gives θ_k. Both speeds increase with γ_k, which gives the balances. □

To compare contraction rates rather than balance ratios alone, match the fastest-direction speed to $\tilde{\eta} \in (0,1)$, i.e. $W^+ = W - \frac{\tilde{\eta}}{\gamma_{\max}} G$ for GD and $W^+ = W - \frac{\tilde{\eta}(\gamma_{\max}+\rho)}{\gamma_{\max}} G(\Sigma + \rho I)^{-1}$ for AP. Then, for every k,

$$\frac{1-\theta_k^{\mathrm{AP}}}{1-\theta_k^{\mathrm{GD}}} = \frac{\gamma_{\max}+\rho}{\gamma_k+\rho} \geq 1. \tag{54}$$

The factor Σ in G makes the gradient-descent speed proportional to input energy; activation preconditioning compensates for this dependence through $(\Sigma + \rho I)^{-1}$.

Corollary D.10 (A balanced update reduces irreducible loss in the linear model)**.** *Assume (A1)–(A3), $\sigma^2 < \gamma_{\max}$. Let $w_\star \neq 0$ be the transpose of the corresponding row of $W_\star$, and fix β. Initialize both update rules at $w(0) = w_\star + u(0)$ with $u(0) \in \mathcal{S}^\perp \setminus \{0\}$, and match their fastest-direction speed to $\tilde{\eta} \in (0,1)$. For an integer $T \geq 1$, the weight after T steps is $w(T) = w_\star + u(T)$ for both rules, with*

$$\|u^{\mathrm{GD}}(T)\|_2 = \Big(1 - \frac{\tilde{\eta}\,\sigma^2}{\gamma_{\max}}\Big)^T \|u(0)\|_2, \qquad \|u^{\mathrm{AP}}(T)\|_2 = \Big(1 - \frac{\tilde{\eta}\,\sigma^2}{\sigma^2+\rho} \cdot \frac{\gamma_{\max}+\rho}{\gamma_{\max}}\Big)^T \|u(0)\|_2, \tag{55}$$

so that $\|u^{\mathrm{AP}}(T)\|_2 < \|u^{\mathrm{GD}}(T)\|_2$ and, if $\Pr[m_{w_\star}(g(O)) > 0] > 0$,

$$\mathbb{H}\big(c_T^{\mathrm{AP}} \mid O\big) < \mathbb{H}\big(c_T^{\mathrm{GD}} \mid O\big) \qquad \textit{for every } T \geq 1, \tag{56}$$

where c_T^{GD} and c_T^{AP} are the tokens defined by $w(T)$ under the two rules.

Proof. By Lemma D.6, $\mathcal{S}^\perp$ is an eigenspace of Σ with eigenvalue σ^2, so both updates preserve $w_{\mathcal{S}} = w_\star$ and contract u by the factors in Lemma D.9 with $\gamma_k = \sigma^2$. Equation 54 and $\sigma^2 < \gamma_{\max}$ imply strictly faster contraction under activation preconditioning. Proposition D.5 then gives the entropy ordering under its stated strictness condition. □

Equation 56 directly compares the irreducible losses of two tokenizers with the same reconstruction target $w_\star$. For $\rho \ll \gamma_{\max}$, gradient descent removes the wasted weight on the time scale $\gamma_{\max}/\sigma^2$ and activation preconditioning on the time scale $1 + \rho/\sigma^2$. Combining Propositions D.1 and D.5 with Equation 32 gives, in the example,

$$J_E \leq \mathcal{L}_{\mathrm{tok}} + c_{\mathrm{dec}}\, M_{\neq}\Big[\mathbb{E}\, h_2\Big(\Phi\Big(\frac{m_{w_\star}(g(O))}{\sigma\,\kappa_T}\Big)\Big) + \mathbb{E}_O\,\mathrm{KL}\big(P(C \mid O) \,\big\|\, p_\psi(C \mid O)\big)\Big], \tag{57}$$

where $\kappa_T^2 = 1 + \theta^{2T}\|u(0)\|_2^2/\|w_\star\|_2^2$ and θ is the contraction factor of the training rule on $\mathcal{S}^\perp$. Reconstruction training controls $\mathcal{L}_{\mathrm{tok}}$, Token Perturbation targets $M_{\neq}$, policy training controls the divergence, and TBR acts through θ.

Extension beyond a binary token. The general entropy and margin bounds already apply to any finite codebook. For a linear multiclass tokenizer $\tau(a) = \arg\max_j (w_j^\top a + b_j)$, removing the noise-subspace component of every w_j preserves all scores on $\mathcal{S}$. At a center with a unique winning label j, its cell margin is

$$\min_{k \neq j} \frac{(w_j - w_k)^\top g(O) + b_j - b_k}{\|w_j - w_k\|_2}.$$

Projecting all weights onto $\mathcal{S}$ leaves each numerator fixed and cannot increase any denominator, so it cannot decrease this margin, thus giving a nonincreasing entropy upper bound.

A local extension to nonlinear decision boundaries. The results in Section D.1.1 apply unchanged to nonlinear tokenizers. For a local geometric extension, consider a binary decision $c_\vartheta(a) = \mathbf{1}\{f_\vartheta(a) > \beta\}$ and a reference action $\mu = g(o)$, where f_ϑ is twice continuously differentiable. Define the score gap and input gradient

$$\Delta := |f_\vartheta(\mu) - \beta| > 0, \qquad v := \nabla_a f_\vartheta(\mu).$$

The linearized boundary has distance $\Delta/\|v\|_2$ when $v \neq 0$, giving a first-order version of the geometry in Proposition D.5.

Discussions for multimodal conditional actions. The general bounds do not assume unimodality. A mode variable M can instead be introduced and reference actions defined conditionally on (O, M). For a finite-valued M,

$$\mathbb{H}(C \mid O) = \mathbb{H}(C \mid O, M) + I(C; M \mid O) \leq \mathbb{H}(C \mid O, M) + \mathbb{H}(M \mid O).$$

Modewise margin bounds control the first term and therefore also give an upper bound on the total conditional entropy. Reducing within-mode uncertainty does not alone establish a decrease in $\mathbb{H}(C \mid O)$, because $I(C; M \mid O)$ can change with the tokenizer.

Summary. Token margins provide a way to bound irreducible policy log-loss. In the linear fixed-target model, TBR through activation preconditioning implements BFL by accelerating the removal of noise-subspace sensitivity, while preserving the clean-subspace partition in the controlled comparison. This increases margins and strictly reduces both conditional token entropy and Bayes binary token-classification error. These results identify one mechanism through which the BFL principle used by TBR can improve token predictability; the empirical analyses in Section 6.3 and Appendix E.3 address its behavior in the full tokenizer.

D.2 Token Balance Regularization: Implementation

Activation preconditioning in tokenizer training. We instantiate TBR with the activation-preconditioning component of DoPr (Zhang et al., 2026), applied to the tokenizer's own optimization and to nothing else: the downstream policy is trained afterwards on the frozen tokenizer, and the forward architecture, parameter count, and checkpoint format are unchanged. Consider an eligible layer with weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ acting on an input activation $x \in \mathbb{R}^{d_{\text{in}}}$ as $x \mapsto Wx$, and let $x_1, \ldots, x_n$ collect every activation that layer receives during one optimizer step. We estimate the uncentered activation second moment and its trace-normalized ridge as

$$\widehat{\Sigma} = \frac{1}{n} \sum_{t=1}^{n} x_t x_t^\top, \qquad \rho = \lambda \, \frac{\operatorname{tr} \widehat{\Sigma}}{d_{\text{in}}}, \qquad \widehat{\Sigma}_\rho = \widehat{\Sigma} + \rho I_{d_{\text{in}}}, \tag{58}$$

with damping $\lambda > 0$. Normalizing the ridge by $\operatorname{tr} \widehat{\Sigma}/d_{\text{in}}$ makes ρ track each layer's activation scale, so a single λ transfers across layers and across tokenizer backbones. Writing $G = \partial \mathcal{L}/\partial W$ for the raw weight gradient of the tokenizer objective, the preconditioned gradient is obtained by right-multiplication:

$$\widetilde{G} = G \, \widehat{\Sigma}_\rho^{-1}, \tag{59}$$

and the optimizer consumes $\widetilde{G}$ in place of G. Because $G = \sum_t \delta_t x_t^\top$ for output-side gradients δ_t, Equation 59 divides out the input-covariance factor that scales updates with activation energy, equalizing progress across activation directions instead of letting a few high-variance features dominate. This is the input-side factor of the layerwise curvature approximation used by K-FAC (Martens & Grosse, 2015). We apply it inside an AdamW step, and we leave the codebook embeddings, biases, and normalization parameters to AdamW alone. Within a training step the order is backward pass, activation preconditioning, global gradient-norm clipping, then the AdamW update.

Eligible layers and activation statistics. Preconditioning covers the linear, multi-head attention, and convolutional layers of both the tokenizer backbone and the quantizer. For a linear layer, x is its input and d_{in} its input width, pooled over all chunk positions in the minibatch. For packed same-dimension multi-head attention, the query, key, and value inputs receive three separate factors, and the corresponding row blocks of the packed input-projection gradient are preconditioned independently; the output projection is excluded because it is reached through a functional path that does not expose its input. For a convolution with a single group, x is an extracted patch and $d_{\text{in}} = C_{\text{in}} k_h k_w$, which treats the convolution as a linear map on patches. Statistics are re-estimated from scratch at every optimizer step rather than accumulated in a running average, so $\widehat{\Sigma}$ always describes the current minibatch.

Wide layers and numerical fallbacks. When $d_{\text{in}} \leq d_{\max}$ we form $\widehat{\Sigma}_\rho$ explicitly and apply Equation 59 through its Cholesky factor. For wider layers, and for every convolution, we instead use a

rank-one-plus-diagonal surrogate built from the activation mean and variance,

$$\mu = \frac{1}{n}\sum_{t=1}^{n} x_t, \qquad v = \frac{1}{n}\sum_{t=1}^{n} x_t \odot x_t - \mu \odot \mu, \qquad \widehat{\Sigma} \approx \operatorname{diag}(v) + \mu\mu^\top, \tag{60}$$

whose damped inverse follows from the Sherman–Morrison identity with $D = \operatorname{diag}(v + \rho)$,

$$\widehat{\Sigma}_\rho^{-1} = D^{-1} - \frac{D^{-1}\mu\mu^\top D^{-1}}{1 + \mu^\top D^{-1}\mu}. \tag{61}$$

This costs $O(d_{\text{in}})$ memory and time per layer instead of $O(d_{\text{in}}^2)$ and $O(d_{\text{in}}^3)$. Two guards keep the step well defined. A low-rank activation batch can give $\widehat{\Sigma}$ small negative eigenvalues in single precision and make the Cholesky factorization fail; we then symmetrize, take an eigendecomposition, clamp the eigenvalues at zero, and add ρ. If any eligible gradient is non-finite, preconditioning is skipped for that step and the raw gradients are passed to clipping unchanged. Table 13 lists the damping λ by benchmark and quantizer family and the maximum full-covariance dimension $d_{\max}$.

Table 13: Token Balance Regularization hyperparameters used for TBR-enabled tokenizer training in each simulation benchmark. The LIBERO VQ values apply to both global and per-level configurations from the replacement runs.

Hyperparameter	Symbol	RoboMimic	LIBERO	RoboTwin
Damping, VQ	λ	10^{-1}	10^{-1}	10^{-1}
Damping, RVQ/NRVQ/FSQ/QinCo	λ	10^{-4}	10^{-4}	10^{-3}
Maximum full-covariance dimension	$d_{\max}$	512	512	512

D.3 Token Perturbation: Implementation and Alternative Sampling Studies

Perturbation sampling and training. During tokenizer training, TP perturbs the clean encoding $C = \mathcal{T}(A)$ using a conditional distribution $\nu(\cdot \mid C)$ to reconstruct A. Write $z_q = e(C)$, where e is the token-embedding map.

$$C' \sim \nu(\cdot \mid C), \qquad z_q^{\text{TP}} = z_q + \text{sg}(e(C') - z_q)\,, \tag{62}$$

where sg denotes stop-gradient. Forward decoding uses $e(C')$; the backward pass sends the identity gradient through clean z_q. For each batch, we perturb each sample with Bernoulli probability β. Unperturbed samples retain clean decoder inputs. The reconstruction loss is evaluated on the mixed batch against the original actions. Codebook and commitment auxiliaries use the clean quantization pass and are not recomputed from C'; no separate adversarial or degradation loss is added.

Native-metric neighborhood sampling. Our main experiments use a sampler adapted from RobustTok (Qiu et al., 2025). For each perturbed sample, $\nu_{\alpha,\delta}(\cdot \mid C)$ independently masks token positions with probability α. Each masked token is sampled uniformly from the δ nearest candidates under Euclidean distance in the quantizer's latent space. For **VQ**, candidates are vectors in the applicable shared or level-specific codebook, ranked by distance to the encoder latent embedding. For **FSQ**, candidates are enumerated packed grid codes projected into the corresponding latent embedding. For **RVQ** and **NRVQ**, candidates are the level's code vectors and scaled normalized code vectors, respectively, ranked against its clean residual. **QinCo** likewise uses the clean residual, with a static codebook at level zero and prefix-conditioned codebooks thereafter. All residuals and conditioned codebooks use the clean prefix, even when earlier sampled tokens change; decoding uses the complete perturbed sequence. Both α and δ are annealed linearly during training; Table 14 lists the benchmark-specific settings.

Table 14: Token Perturbation hyperparameters used in each simulation benchmark. The same setting is shared by every tokenizer configuration within a benchmark.

Hyperparameter	Symbol	RoboMimic	LIBERO	RoboTwin
Row selection probability	β	0.1	0.1	0.1
Mask probability	α	$1.0 \to 0.5$	$1.0 \to 0.5$	$0.5 \to 0.125$
Neighborhood size	δ	$15 \to 8$	$15 \to 8$	$4 \to 1$

Alternative perturbation and selection rules. We study two alternatives to test whether more coordinated perturbations or harder training examples improve robustness. *Spectral-metric perturbation* changes how distances to candidate codes are measured: it randomly rotates the quantizer's spectral basis and assigns positive weights to the resulting directions. Candidate distance is then a weighted sum of squared differences along these directions, so a code that is farther away under ordinary Euclidean distance can become the nearest choice. The sampler requantizes every position using this modified distance. The random rotation and weighting are shared across levels, applied within each level's own basis; residual and conditioned quantizers update their state using the newly selected prefix. This construction is intended to coordinate changes across the sequence. *MaxUp* draws eight complete sequences from the native-metric neighborhood sampler and trains on the candidate with the largest reconstruction MSE. Following the worst-augmentation principle of MaxUp (Gong et al., 2020), this tests whether emphasizing difficult perturbations improves robustness beyond single-draw sampling.

Comparison results. We compare the alternatives on Robomimic Can. Here, TP denotes the native-metric sampling, and Alternative means either *Spectral-metric perturbation* or applying *MaxUp* on TP. We report best and last checkpoints separately at $H_{\text{exec}} \in \{4, 8\}$.

Table 15: Controlled Token Perturbation studies on RoboMimic Can. Success rates and deltas are percentage points, averaged over the tokenizer configurations in each study. Δ_{R} compares the alternative with Reconstruction-only, while Δ_{TP} compares it with Token Perturbation. W/L/T counts compare the alternative against the corresponding reference configuration by configuration.

Ckpt.	H_{exec}	Recon.	TP	Alternative	Δ_{R}	W/L/T	Δ_{TP}	W/L/T
Spectral-metric perturbation (8 configurations)								
Best	4	77.08	82.46	74.37	-2.71	4/4/0	-8.08	3/4/1
Best	8	68.12	78.88	70.17	$+2.04$	4/4/0	-8.71	2/5/1
Last	4	75.79	82.17	73.58	-2.21	5/3/0	-8.58	4/4/0
Last	8	68.38	76.79	69.54	$+1.17$	6/2/0	-7.25	3/5/0
MaxUp $K = 8$ *(10 Transformer configurations)*								
Best	4	81.13	86.03	83.30	$+2.17$	6/4/0	-2.73	4/6/0
Best	8	74.70	83.07	79.17	$+4.47$	7/3/0	-3.90	4/6/0
Last	4	80.53	86.27	84.00	$+3.47$	6/4/0	-2.27	5/4/1
Last	8	73.23	79.80	78.43	$+5.20$	7/3/0	-1.37	5/5/0

The performance of Spectral-metric perturbation is in general mixed relative to Token Perturbation. MaxUp remains above Reconstruction-only on average, yet is 1.37–3.90 points below the matched single-draw treatment.

Discussion. One possible explanation for the spectral variant's inconsistent gains is that changing the latent-space metric can therefore produce perturbations with different action-space consequences across quantizers. In residual and prefix-conditioned quantizers, an early replacement changes subsequent residuals or codebooks, potentially propagating its effect through the sequence. Codebook geometry may also matter: random rotations and reweighting can mix dominant and low-variance directions, changing neighborhood rankings in ways that may poorly reflect meaningful action differences. Therefore, it could disrupt useful structure rather than plausible token errors.

For MaxUp, selecting the largest reconstruction error among eight candidates emphasizes the high-error tail of the perturbation distribution. Neither candidate generation nor selection uses downstream-policy errors, so the hardest reconstruction candidate may not represent a likely policy mistake. Training on such candidates may overemphasize rare, severe changes and create a stronger trade-off between reconstruction fidelity and robustness. A severe perturbation may also produce a sequence representing a substantially different action, for which the original action is no longer an appropriate reconstruction target. Training the decoder to map such sequences back to the original action could blur distinctions between actions and compromise reconstruction fidelity. This could be particularly relevant when independently sampled replacements interact across token levels, although the mixed per-tokenizer outcomes do not establish a uniform family-level effect. These explanations remain hypotheses: distinguishing them would require examining codebook spectra, perturbation severity, and whether sampled sequences preserve action similarity and resemble policy-produced mismatches.

E Additional Experimental Results

E.1 Primary Comparison: Task-Level Results and Robustness Checks

This subsection expands Table 1. Appendix E.1.1 reports validation reconstruction errors, Appendix E.1.2 reports each task or suite separately, and Appendix E.1.3 checks on Robomimic whether the comparison holds with the last checkpoint and with single-task policies.

E.1.1 Validation Reconstruction Error

Table 16 reports tokenizer reconstruction MAE $\mathcal{E}_{\text{rec}}$ on each benchmark's multi-task validation data for reconstruction-only training and ProAct. These errors measure reconstruction of the input action chunks in raw action units and are independent of the policy execution horizon. We report each benchmark separately because their action spaces differ.

Table 16: Multi-task validation reconstruction MAE $\mathcal{E}_{\text{rec}}$ in raw action units (↓). R.O. denotes reconstruction-only training. **Bold** marks the lower value within each benchmark and tokenizer pair; ties at the displayed precision are both bold.

Tokenizer	RoboMimic		LIBERO		RoboTwin	
	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct
TF-VQ-S	**0.02256**	0.02558	**0.01334**	0.02327	**0.00080**	0.00088
TF-VQ-L	**0.02329**	0.04603	**0.01567**	0.03743	**0.00077**	0.00087
TF-RVQ-S	**0.02896**	0.03099	**0.01941**	0.02477	**0.00085**	0.00110
TF-RVQ-L	**0.02420**	0.02746	**0.01552**	0.02101	**0.00071**	0.00103
CT-RVQ-S	**0.02690**	0.02761	0.02052	**0.01990**	**0.00068**	0.00091
CT-RVQ-L	**0.02271**	0.02560	**0.01466**	0.02155	**0.00057**	0.00075
C2-RVQ-S	**0.02657**	0.02893	**0.02025**	0.02321	**0.00090**	0.00170
C2-RVQ-L	**0.02197**	0.02866	**0.01403**	0.02429	**0.00060**	0.00075
TF-NRVQ-S	0.03058	**0.02984**	**0.01954**	0.02385	**0.00125**	0.00157
TF-NRVQ-L	**0.02422**	0.02585	**0.01488**	0.01870	**0.00098**	0.00102
CT-NRVQ-S	0.26266	**0.02977**	0.28457	**0.02093**	0.06441	**0.00136**
CT-NRVQ-L	0.26257	**0.03199**	0.28403	**0.02634**	0.06438	**0.00102**
C2-NRVQ-S	0.03149	**0.02789**	**0.01820**	0.02096	**0.00098**	0.00111
C2-NRVQ-L	**0.02250**	0.02873	**0.01504**	0.02218	**0.00068**	0.00100
TF-FSQ-S	**0.02688**	0.03371	**0.01937**	0.02836	**0.00092**	0.00156
TF-FSQ-L	**0.02601**	0.02891	**0.01896**	0.02381	**0.00085**	0.00143
TF-QINCO-S	**0.02475**	0.03577	**0.01555**	0.03242	**0.00081**	0.00114
TF-QINCO-L	**0.02281**	0.03327	**0.01395**	0.02837	**0.00061**	0.00099
CT-QINCO-S	**0.02306**	0.03659	**0.01430**	0.03215	**0.00066**	0.00123
CT-QINCO-L	**0.02129**	0.03386	**0.01402**	0.02898	**0.00059**	0.00075
C2-QINCO-S	**0.02150**	0.03745	**0.01455**	0.03347	**0.00077**	0.00138
C2-QINCO-L	**0.02177**	0.03232	**0.01408**	0.03146	**0.00049**	0.00079

E.1.2 TASK- AND SUITE-LEVEL RESULTS

Robomimic. Table 17 reports Can, Lift, and Square separately for the multi-task setting at the checkpoint with the best teacher-forced validation accuracy. It compares Reconstruction-only with ProAct at both execution horizons and retains the confidence interval computed from each task's 300 binary rollout outcomes.

Table 17: Task-level multi-task RoboMimic success (%) at the checkpoint with the best teacher-forced validation accuracy. Each tokenizer occupies two lines for $H_{\text{exec}} = 4$ and 8 success rate $\pm$ 95% confidence-interval half-width. R.O. denotes Reconstruction Only; **bold** marks the better objective per task and horizon.

Tokenizer	Can		Lift		Square	
	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct
TF-VQ-S	4: 86.7 ± 3.8	4: **95.3 ± 2.4**	4: 82.3 ± 4.3	4: **99.0 ± 1.1**	4: 51.3 ± 5.7	4: **64.0 ± 5.5**
	8: 91.0 ± 3.2	8: **95.3 ± 2.4**	8: 85.7 ± 4.0	8: **100.0 ± 0.0**	8: **64.0 ± 5.4**	8: 63.3 ± 5.4
TF-VQ-L	4: 87.7 ± 3.7	4: **90.7 ± 3.3**	4: 93.7 ± 2.7	4: **98.7 ± 1.3**	4: **46.3 ± 5.7**	4: 43.3 ± 5.6
	8: 86.0 ± 3.9	8: **86.3 ± 3.9**	8: 87.0 ± 3.8	8: **98.3 ± 1.4**	8: **42.7 ± 5.6**	8: 33.3 ± 5.3
TF-RVQ-S	4: 86.0 ± 3.9	4: **92.0 ± 3.1**	4: 91.3 ± 3.2	4: **92.0 ± 3.1**	4: 60.3 ± 5.5	4: **69.0 ± 5.3**
	8: 90.7 ± 3.3	8: **94.0 ± 2.7**	8: 90.3 ± 3.3	8: **94.3 ± 2.6**	8: 54.3 ± 5.7	8: **70.3 ± 5.1**
TF-RVQ-L	4: 73.0 ± 5.1	4: **86.7 ± 3.9**	4: 70.7 ± 5.2	4: **79.3 ± 4.6**	4: 36.0 ± 5.4	4: **53.3 ± 5.7**
	8: 63.7 ± 5.4	8: **80.7 ± 4.5**	8: 77.0 ± 4.7	8: **81.0 ± 4.4**	8: 26.0 ± 4.9	8: **43.3 ± 5.6**
CT-RVQ-S	4: 91.0 ± 3.3	4: **95.0 ± 2.5**	4: **92.0 ± 3.1**	4: 88.7 ± 3.6	4: **61.7 ± 5.5**	4: 59.0 ± 5.6
	8: 91.0 ± 3.3	8: **96.0 ± 2.2**	8: 92.3 ± 3.0	8: **95.0 ± 2.4**	8: 55.3 ± 5.6	8: **64.3 ± 5.5**
CT-RVQ-L	4: 75.3 ± 4.8	4: **80.0 ± 4.5**	4: 74.7 ± 4.9	4: **80.0 ± 4.5**	4: 36.0 ± 5.4	4: **40.7 ± 5.5**
	8: 69.0 ± 5.2	8: **77.7 ± 4.7**	8: 76.0 ± 4.9	8: **80.3 ± 4.5**	8: 32.7 ± 5.3	8: **36.3 ± 5.4**
C2-RVQ-S	4: 94.0 ± 2.7	4: **97.7 ± 1.7**	4: **97.3 ± 1.8**	4: 90.7 ± 3.3	4: **64.0 ± 5.5**	4: 57.7 ± 5.6
	8: 92.0 ± 3.1	8: **94.3 ± 2.6**	8: **96.0 ± 2.2**	8: 93.0 ± 2.9	8: 61.7 ± 5.5	8: **64.0 ± 5.5**
C2-RVQ-L	4: 71.7 ± 5.1	4: **80.0 ± 4.5**	4: 70.7 ± 5.2	4: **73.7 ± 5.0**	4: 35.3 ± 5.4	4: **44.0 ± 5.6**
	8: 66.0 ± 5.3	8: **74.0 ± 5.0**	8: **75.3 ± 4.9**	8: 74.7 ± 4.9	8: 32.3 ± 5.3	8: **35.7 ± 5.5**
TF-NRVQ-S	4: 89.3 ± 3.5	4: **94.3 ± 2.6**	4: **89.0 ± 3.5**	4: 88.7 ± 3.6	4: 61.7 ± 5.5	4: **70.0 ± 5.2**
	8: 90.0 ± 3.4	8: **95.7 ± 2.3**	8: **88.3 ± 3.6**	8: 88.0 ± 3.6	8: 63.7 ± 5.5	8: **69.3 ± 5.2**
TF-NRVQ-L	4: 70.3 ± 5.1	4: **73.0 ± 5.0**	4: 65.3 ± 5.3	4: **71.7 ± 5.1**	4: 34.0 ± 5.3	4: **37.0 ± 5.5**
	8: 60.0 ± 5.5	8: **65.3 ± 5.4**	8: 64.0 ± 5.4	8: **75.0 ± 4.9**	8: 21.0 ± 4.6	8: **31.0 ± 5.2**
CT-NRVQ-S	4: 0.0 ± 0.0	4: **91.0 ± 3.2**	4: 0.0 ± 0.0	4: **88.3 ± 3.6**	4: 0.0 ± 0.0	4: **64.0 ± 5.5**
	8: 0.0 ± 0.0	8: **92.3 ± 3.0**	8: 0.0 ± 0.0	8: **96.7 ± 2.0**	8: 0.0 ± 0.0	8: **70.7 ± 5.1**
CT-NRVQ-L	4: 0.0 ± 0.0	4: **73.0 ± 5.0**	4: 0.0 ± 0.0	4: **79.3 ± 4.6**	4: 0.0 ± 0.0	4: **32.7 ± 5.3**
	8: 0.0 ± 0.0	8: **59.0 ± 5.5**	8: 0.0 ± 0.0	8: **79.0 ± 4.6**	8: 0.0 ± 0.0	8: **24.0 ± 4.8**
C2-NRVQ-S	4: **91.7 ± 3.1**	4: 89.3 ± 3.5	4: **90.0 ± 3.4**	4: 85.0 ± 4.0	4: 60.7 ± 5.5	4: **61.0 ± 5.6**
	8: 92.3 ± 3.0	8: **93.3 ± 2.8**	8: **95.3 ± 2.4**	8: 87.7 ± 3.7	8: 61.0 ± 5.5	8: **67.7 ± 5.3**
C2-NRVQ-L	4: 68.7 ± 5.3	4: **76.7 ± 4.8**	4: 71.7 ± 5.1	4: **72.7 ± 5.1**	4: 28.7 ± 5.1	4: **35.0 ± 5.4**
	8: 54.0 ± 5.7	8: **62.3 ± 5.5**	8: 68.7 ± 5.2	8: **69.0 ± 5.2**	8: 21.3 ± 4.6	8: **24.7 ± 4.9**
TF-FSQ-S	4: **91.0 ± 3.2**	4: 89.0 ± 3.5	4: 83.3 ± 4.2	4: **91.7 ± 3.1**	4: 49.7 ± 5.6	4: **60.0 ± 5.5**
	8: 89.7 ± 3.5	8: **92.3 ± 3.0**	8: 85.0 ± 4.0	8: **93.0 ± 2.9**	8: 49.7 ± 5.6	8: **57.0 ± 5.6**
TF-FSQ-L	4: 91.0 ± 3.2	4: **94.0 ± 2.7**	4: 88.7 ± 3.6	4: **91.0 ± 3.3**	4: 53.3 ± 5.7	4: **59.3 ± 5.6**
	8: 87.3 ± 3.8	8: **92.7 ± 3.0**	8: 84.7 ± 4.1	8: **94.0 ± 2.7**	8: 50.0 ± 5.7	8: **63.7 ± 5.4**
TF-QINCO-S	4: 55.3 ± 5.7	4: **89.7 ± 3.4**	4: 62.3 ± 5.5	4: **83.7 ± 4.2**	4: 27.7 ± 5.0	4: **54.0 ± 5.6**
	8: 47.3 ± 5.6	8: **86.3 ± 3.9**	8: 61.3 ± 5.5	8: **84.0 ± 4.2**	8: 12.7 ± 3.8	8: **51.0 ± 5.7**
TF-QINCO-L	4: 51.0 ± 5.6	4: **86.3 ± 3.9**	4: 64.7 ± 5.5	4: **82.7 ± 4.3**	4: 19.0 ± 4.4	4: **53.0 ± 5.6**
	8: 32.7 ± 5.4	8: **85.7 ± 4.0**	8: 56.3 ± 5.6	8: **82.0 ± 4.3**	8: 8.7 ± 3.1	8: **48.7 ± 5.6**
CT-QINCO-S	4: 67.0 ± 5.3	4: **86.0 ± 4.0**	4: 74.0 ± 4.9	4: **83.7 ± 4.2**	4: 34.0 ± 5.5	4: **43.0 ± 5.7**
	8: 56.3 ± 5.7	8: **77.3 ± 4.7**	8: 65.3 ± 5.5	8: **77.0 ± 4.8**	8: 18.7 ± 4.4	8: **36.7 ± 5.5**
CT-QINCO-L	4: 59.0 ± 5.6	4: **79.7 ± 4.6**	4: 73.7 ± 5.0	4: **85.7 ± 4.0**	4: 23.7 ± 4.8	4: **48.3 ± 5.7**
	8: 45.0 ± 5.6	8: **74.7 ± 4.9**	8: 66.3 ± 5.3	8: **76.7 ± 4.8**	8: 17.3 ± 4.3	8: **41.0 ± 5.7**
C2-QINCO-S	4: 59.7 ± 5.5	4: **91.3 ± 3.2**	4: 70.0 ± 5.2	4: **79.3 ± 4.5**	4: 21.3 ± 4.7	4: **50.3 ± 5.6**
	8: 50.0 ± 5.7	8: **87.7 ± 3.8**	8: 64.3 ± 5.4	8: **78.0 ± 4.7**	8: 11.3 ± 3.6	8: **48.0 ± 5.7**
C2-QINCO-L	4: 61.7 ± 5.5	4: **73.3 ± 4.9**	4: 72.3 ± 5.1	4: **80.0 ± 4.5**	4: 17.7 ± 4.3	4: **29.0 ± 5.1**
	8: 47.0 ± 5.6	8: **61.7 ± 5.5**	8: 63.3 ± 5.5	8: **74.0 ± 5.0**	8: 9.7 ± 3.3	8: **21.3 ± 4.7**

LIBERO. Table 18 reports Spatial, Object, Goal, and LIBERO-10 separately for the multi-task setting. It compares Reconstruction-only with ProAct at both execution horizons and retains each suite's supplied confidence interval from 500 evaluation episodes.

Table 18: Suite-level multi-task LIBERO success (%) at the checkpoint with the best teacher-forced validation accuracy. Each tokenizer occupies two lines for $H_{\text{exec}} = 4$ and 8 success rate $\pm$ the supplied 95% confidence-interval half-width. Each suite contains 10 tasks and is evaluated for 500 episodes using one fixed evaluation seed. R.O. denotes Reconstruction Only; **bold** marks the better objective per suite and horizon.

Tokenizer	Spatial		Object		Goal		LIBERO-10	
	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct
TF-VQ-S	4: 54.6 ± 4.4	4: **61.6 ± 4.3**	4: 71.8 ± 4.0	4: **74.0 ± 3.8**	4: 28.2 ± 3.9	4: **38.2 ± 4.2**	4: 47.8 ± 4.4	4: **52.8 ± 4.4**
	8: 42.6 ± 4.3	8: **52.4 ± 4.4**	8: 67.0 ± 4.1	8: **71.0 ± 4.0**	8: 23.0 ± 3.7	8: **40.6 ± 4.3**	8: **49.0 ± 4.4**	8: 48.6 ± 4.3
TF-VQ-L	4: 57.6 ± 4.3	4: **59.4 ± 4.3**	4: 71.2 ± 4.0	4: **83.8 ± 3.2**	4: 29.0 ± 4.0	4: **45.8 ± 4.4**	4: 42.2 ± 4.3	4: **45.4 ± 4.4**
	8: 47.0 ± 4.4	8: **50.2 ± 4.4**	8: **76.4 ± 3.7**	8: 76.0 ± 3.7	8: 28.4 ± 4.0	8: **39.2 ± 4.3**	8: **45.6 ± 4.4**	8: 45.4 ± 4.4
TF-RVQ-S	4: 67.6 ± 4.1	4: **71.6 ± 4.0**	4: 80.0 ± 3.5	4: **85.8 ± 3.1**	4: 65.2 ± 4.2	4: **68.2 ± 4.1**	4: 59.6 ± 4.3	4: **63.0 ± 4.2**
	8: 59.0 ± 4.3	8: **61.0 ± 4.3**	8: 76.4 ± 3.7	8: **85.6 ± 3.1**	8: 61.2 ± 4.3	8: **68.2 ± 4.1**	8: 61.2 ± 4.3	8: **63.4 ± 4.2**
TF-RVQ-L	4: 60.0 ± 4.3	4: **66.8 ± 4.1**	4: 75.8 ± 3.7	4: **85.0 ± 3.1**	4: 51.4 ± 4.4	4: **63.0 ± 4.2**	4: 42.8 ± 4.4	4: **56.6 ± 4.3**
	8: 52.0 ± 4.4	8: **53.0 ± 4.4**	8: 71.2 ± 4.0	8: **78.0 ± 3.6**	8: 46.8 ± 4.4	8: **55.0 ± 4.3**	8: 37.4 ± 4.2	8: **54.6 ± 4.4**
CT-RVQ-S	4: 48.8 ± 4.4	4: **73.0 ± 3.9**	4: 69.4 ± 4.1	4: **82.8 ± 3.3**	4: 27.0 ± 3.9	4: **64.8 ± 4.2**	4: 10.2 ± 2.7	4: **60.2 ± 4.3**
	8: 53.8 ± 4.4	8: **67.2 ± 4.1**	8: **81.8 ± 3.4**	8: 80.2 ± 3.5	8: 25.0 ± 3.8	8: **63.8 ± 4.2**	8: 24.8 ± 3.8	8: **61.4 ± 4.3**
CT-RVQ-L	4: 47.8 ± 4.3	4: **67.2 ± 4.1**	4: 74.4 ± 3.8	4: **83.6 ± 3.3**	4: 38.2 ± 4.2	4: **57.8 ± 4.3**	4: 33.8 ± 4.1	4: **54.4 ± 4.4**
	8: 44.8 ± 4.3	8: **58.0 ± 4.3**	8: 66.0 ± 4.1	8: **77.2 ± 3.7**	8: 38.4 ± 4.3	8: **53.4 ± 4.4**	8: 35.2 ± 4.2	8: **47.2 ± 4.4**
C2-RVQ-S	4: 75.0 ± 3.8	4: **75.2 ± 3.8**	4: 81.8 ± 3.4	4: **83.0 ± 3.3**	4: 59.8 ± 4.3	4: **68.6 ± 4.1**	4: **67.0 ± 4.1**	4: 53.6 ± 4.3
	8: **63.4 ± 4.2**	8: 63.0 ± 4.2	8: 84.0 ± 3.2	8: **84.6 ± 3.2**	8: 64.4 ± 4.2	8: **67.6 ± 4.1**	8: **65.4 ± 4.1**	8: 56.8 ± 4.4
C2-RVQ-L	4: 55.6 ± 4.4	4: **63.0 ± 4.2**	4: 74.6 ± 3.8	4: **83.4 ± 3.2**	4: 50.4 ± 4.3	4: **58.4 ± 4.4**	4: 36.8 ± 4.2	4: **54.0 ± 4.4**
	8: 46.8 ± 4.3	8: **50.2 ± 4.3**	8: 73.2 ± 3.9	8: **80.4 ± 3.5**	8: 46.0 ± 4.4	8: **55.6 ± 4.3**	8: 34.2 ± 4.2	8: **47.2 ± 4.4**
TF-NRVQ-S	4: 44.4 ± 4.3	4: **54.0 ± 4.3**	4: 65.6 ± 4.2	4: **73.4 ± 3.9**	4: 28.8 ± 4.0	4: **31.2 ± 4.0**	4: 6.0 ± 2.1	4: **7.2 ± 2.3**
	8: 37.0 ± 4.3	8: **57.2 ± 4.4**	8: 69.8 ± 4.1	8: **81.0 ± 3.4**	8: 30.8 ± 4.1	8: **32.8 ± 4.1**	8: 17.2 ± 3.3	8: **19.0 ± 3.4**
TF-NRVQ-L	4: 61.6 ± 4.3	4: **67.2 ± 4.1**	4: 71.0 ± 4.0	4: **80.6 ± 3.5**	4: 50.8 ± 4.4	4: **54.2 ± 4.3**	4: 37.6 ± 4.2	4: **44.8 ± 4.3**
	8: **55.6 ± 4.4**	8: 53.4 ± 4.4	8: 67.4 ± 4.1	8: **74.4 ± 3.8**	8: 45.2 ± 4.4	8: **47.0 ± 4.4**	8: 28.0 ± 4.0	8: **38.8 ± 4.3**
CT-NRVQ-S	4: 0.0 ± 0.0	4: **64.6 ± 4.1**	4: 0.0 ± 0.0	4: **81.4 ± 3.4**	4: 0.0 ± 0.0	4: **46.8 ± 4.4**	4: 0.0 ± 0.0	4: **34.0 ± 4.1**
	8: 0.0 ± 0.0	8: **68.8 ± 4.1**	8: 0.0 ± 0.0	8: **86.4 ± 3.0**	8: 0.0 ± 0.0	8: **54.8 ± 4.4**	8: 0.0 ± 0.0	8: **44.8 ± 4.4**
CT-NRVQ-L	4: 0.0 ± 0.0	4: **58.8 ± 4.3**	4: 0.0 ± 0.0	4: **71.0 ± 4.0**	4: 0.0 ± 0.0	4: **45.4 ± 4.4**	4: 0.0 ± 0.0	4: **36.4 ± 4.3**
	8: 0.0 ± 0.0	8: **46.4 ± 4.3**	8: 0.0 ± 0.0	8: **64.2 ± 4.2**	8: 0.0 ± 0.0	8: **42.0 ± 4.3**	8: 0.0 ± 0.0	8: **26.2 ± 3.9**
C2-NRVQ-S	4: 76.6 ± 3.7	4: **78.4 ± 3.6**	4: **84.8 ± 3.1**	4: 81.6 ± 3.4	4: **67.6 ± 4.1**	4: 60.8 ± 4.3	4: **65.6 ± 4.2**	4: 61.8 ± 4.2
	8: **70.0 ± 4.0**	8: 67.0 ± 4.1	8: **84.4 ± 3.2**	8: 83.4 ± 3.2	8: **65.8 ± 4.2**	8: 57.2 ± 4.3	8: **64.0 ± 4.2**	8: 62.4 ± 4.2
C2-NRVQ-L	4: **57.2 ± 4.4**	4: 56.2 ± 4.3	4: 75.8 ± 3.8	4: **76.4 ± 3.7**	4: 47.4 ± 4.4	4: **49.4 ± 4.4**	4: 35.6 ± 4.2	4: **37.6 ± 4.3**
	8: **43.4 ± 4.4**	8: 42.2 ± 4.4	8: 66.6 ± 4.1	8: **74.8 ± 3.8**	8: 42.8 ± 4.4	8: **44.0 ± 4.3**	8: 32.4 ± 4.1	8: **37.0 ± 4.2**
TF-FSQ-S	4: 36.2 ± 4.2	4: **63.2 ± 4.3**	4: 67.6 ± 4.1	4: **79.0 ± 3.5**	4: 11.0 ± 2.8	4: **47.2 ± 4.3**	4: 8.0 ± 2.4	4: **54.4 ± 4.3**
	8: 33.6 ± 4.1	8: **51.2 ± 4.4**	8: 72.4 ± 3.9	8: **82.4 ± 3.3**	8: 10.8 ± 2.7	8: **43.4 ± 4.3**	8: 12.0 ± 2.9	8: **56.6 ± 4.3**
TF-FSQ-L	4: 15.4 ± 3.1	4: **64.8 ± 4.2**	4: 36.4 ± 4.2	4: **83.2 ± 3.3**	4: 9.6 ± 2.6	4: **45.0 ± 4.4**	4: 2.0 ± 1.2	4: **48.6 ± 4.4**
	8: 12.6 ± 2.9	8: **52.2 ± 4.3**	8: 45.8 ± 4.3	8: **80.4 ± 3.4**	8: 6.8 ± 2.2	8: **46.2 ± 4.4**	8: 1.6 ± 1.1	8: **49.6 ± 4.3**
TF-QINCO-S	4: **52.4 ± 4.4**	4: 51.2 ± 4.4	4: 58.8 ± 4.4	4: **66.4 ± 4.2**	4: **44.0 ± 4.4**	4: 42.0 ± 4.3	4: **24.2 ± 3.7**	4: 16.0 ± 3.2
	8: 35.4 ± 4.2	8: **45.4 ± 4.4**	8: 51.4 ± 4.4	8: **60.6 ± 4.3**	8: **35.6 ± 4.2**	8: 34.6 ± 4.2	8: **15.2 ± 3.1**	8: 15.0 ± 3.1
TF-QINCO-L	4: 36.4 ± 4.2	4: **41.0 ± 4.3**	4: 54.8 ± 4.3	4: **68.6 ± 4.1**	4: **34.8 ± 4.2**	4: 32.2 ± 4.1	4: **14.8 ± 3.1**	4: 8.0 ± 2.4
	8: 25.8 ± 3.8	8: **32.2 ± 4.1**	8: 34.0 ± 4.2	8: **66.0 ± 4.1**	8: **28.2 ± 4.0**	8: 27.4 ± 4.0	8: 7.2 ± 2.3	8: **9.4 ± 2.6**
CT-QINCO-S	4: 13.2 ± 2.9	4: **32.6 ± 4.1**	4: 40.2 ± 4.3	4: **57.4 ± 4.3**	4: 24.6 ± 3.8	4: **42.0 ± 4.3**	4: 1.8 ± 1.2	4: **9.8 ± 2.6**
	8: 16.4 ± 3.3	8: **32.2 ± 4.1**	8: 35.8 ± 4.2	8: **51.4 ± 4.3**	8: 21.6 ± 3.6	8: **36.6 ± 4.2**	8: 2.2 ± 1.3	8: **12.2 ± 2.8**
CT-QINCO-L	4: 26.6 ± 3.9	4: **53.0 ± 4.4**	4: 53.4 ± 4.4	4: **64.4 ± 4.2**	4: 23.6 ± 3.8	4: **37.4 ± 4.3**	4: 9.8 ± 2.6	4: **15.2 ± 3.1**
	8: 21.2 ± 3.6	8: **42.6 ± 4.3**	8: 35.0 ± 4.2	8: **62.2 ± 4.3**	8: 21.2 ± 3.6	8: **37.8 ± 4.3**	8: 5.2 ± 1.9	8: **15.2 ± 3.1**
C2-QINCO-S	4: 5.0 ± 1.9	4: **53.2 ± 4.4**	4: 15.8 ± 3.2	4: **62.8 ± 4.2**	4: 7.4 ± 2.3	4: **27.0 ± 3.9**	4: 0.8 ± 0.8	4: **9.0 ± 2.5**
	8: 11.0 ± 2.7	8: **50.4 ± 4.3**	8: 15.2 ± 3.1	8: **58.8 ± 4.3**	8: 10.6 ± 2.7	8: **25.8 ± 3.8**	8: 0.8 ± 0.8	8: **8.6 ± 2.5**
C2-QINCO-L	4: 16.2 ± 3.2	4: **68.4 ± 4.1**	4: 56.6 ± 4.4	4: **81.8 ± 3.4**	4: 20.8 ± 3.6	4: **61.6 ± 4.3**	4: 7.8 ± 2.4	4: **42.8 ± 4.4**
	8: 13.0 ± 2.9	8: **49.8 ± 4.4**	8: 41.2 ± 4.3	8: **74.8 ± 3.8**	8: 21.4 ± 3.6	8: **51.2 ± 4.4**	8: 5.0 ± 1.9	8: **38.6 ± 4.3**

RoboTwin. Table 19 reports Grab roller, Place burger/fries, and Place cans in the plastic box separately for the multi-task setting. It compares Reconstruction-only with ProAct at both execution horizons using 50 episodes per task and horizon.

Table 19: Task-level multi-task RoboTwin success (%) at the checkpoint with the best teacher-forced validation accuracy. Each tokenizer occupies two lines for $H_{\text{exec}} = 4$ and 8 success rate $\pm$ 95% confidence-interval half-width (percentage points), computed by episode bootstrap from the success counts over 50 episodes per task and horizon. Place cans denotes placing cans in the plastic box. R.O. denotes Reconstruction Only; **bold** marks the better objective per task and horizon.

Tokenizer	Grab roller		Place burger/fries		Place cans	
	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct
TF-VQ-S	4: 80.0 ± 10.9 8: 70.0 ± 12.8	4: **84.0 ± 10.1** 8: **74.0 ± 12.1**	4: **76.0 ± 11.8** 8: 62.0 ± 13.5	4: 74.0 ± 12.1 8: **68.0 ± 13.0**	4: **64.0 ± 13.4** 8: **78.0 ± 11.4**	4: 30.0 ± 12.9 8: 50.0 ± 14.1
TF-VQ-L	4: 60.0 ± 13.6 8: 56.0 ± 13.8	4: **88.0 ± 9.0** 8: **78.0 ± 11.4**	4: 26.0 ± 12.4 8: 34.0 ± 13.3	4: **52.0 ± 14.0** 8: **74.0 ± 12.1**	4: 10.0 ± 8.4 8: 34.0 ± 13.3	4: **36.0 ± 13.4** 8: **58.0 ± 13.7**
TF-RVQ-S	4: **94.0 ± 6.6** 8: 84.0 ± 10.1	4: 92.0 ± 7.5 8: **92.0 ± 7.5**	4: **76.0 ± 11.8** 8: **60.0 ± 13.6**	4: 58.0 ± 13.7 8: 56.0 ± 13.8	4: **46.0 ± 14.0** 8: **64.0 ± 13.4**	4: 42.0 ± 13.8 8: 60.0 ± 13.6
TF-RVQ-L	4: 50.0 ± 14.1 8: 40.0 ± 13.7	4: **78.0 ± 11.4** 8: **66.0 ± 13.1**	4: 14.0 ± 9.6 8: 12.0 ± 9.0	4: **34.0 ± 13.3** 8: **28.0 ± 12.6**	4: 6.0 ± 6.6 8: 16.0 ± 10.2	4: **16.0 ± 10.2** 8: **22.0 ± 11.6**
CT-RVQ-S	4: 82.0 ± 10.5 8: **82.0 ± 10.5**	4: **90.0 ± 8.3** 8: 80.0 ± 10.9	4: 60.0 ± 13.6 8: 62.0 ± 13.5	4: **72.0 ± 12.5** 8: **72.0 ± 12.5**	4: **42.0 ± 13.8** 8: 48.0 ± 14.0	4: 40.0 ± 13.7 8: **50.0 ± 14.1**
CT-RVQ-L	4: 46.0 ± 14.0 8: **48.0 ± 14.0**	4: **56.0 ± 13.8** 8: **48.0 ± 14.0**	4: 6.0 ± 6.6 8: 10.0 ± 8.4	4: **28.0 ± 12.6** 8: **28.0 ± 12.6**	4: 0.0 ± 0.0 8: 10.0 ± 8.4	4: **16.0 ± 10.2** 8: **26.0 ± 12.4**
C2-RVQ-S	4: **78.0 ± 11.4** 8: **82.0 ± 10.5**	4: 54.0 ± 13.9 8: 54.0 ± 13.9	4: **46.0 ± 14.0** 8: **42.0 ± 13.8**	4: 16.0 ± 10.2 8: 12.0 ± 9.0	4: **20.0 ± 11.1** 8: **40.0 ± 13.7**	4: 0.0 ± 0.0 8: 0.0 ± 0.0
C2-RVQ-L	4: **72.0 ± 12.5** 8: **50.0 ± 14.1**	4: 54.0 ± 13.9 8: 42.0 ± 13.8	4: **18.0 ± 10.7** 8: 10.0 ± 8.4	4: **18.0 ± 10.7** 8: **28.0 ± 12.6**	4: 2.0 ± 3.9 8: 8.0 ± 7.6	4: **4.0 ± 5.4** 8: **22.0 ± 11.6**
TF-NRVQ-S	4: **88.0 ± 9.0** 8: 62.0 ± 13.5	4: 86.0 ± 9.5 8: **72.0 ± 12.5**	4: 60.0 ± 13.6 8: **52.0 ± 14.0**	4: **64.0 ± 13.4** 8: **52.0 ± 14.0**	4: **40.0 ± 13.7** 8: **56.0 ± 13.8**	4: 34.0 ± 13.3 8: 50.0 ± 14.1
TF-NRVQ-L	4: 54.0 ± 13.9 8: 60.0 ± 13.6	4: **70.0 ± 12.8** 8: **66.0 ± 13.1**	4: 26.0 ± 12.4 8: 28.0 ± 12.6	4: **34.0 ± 13.3** 8: **30.0 ± 12.9**	4: 10.0 ± 8.4 8: 20.0 ± 11.1	4: **20.0 ± 11.1** 8: **22.0 ± 11.6**
CT-NRVQ-S	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **76.0 ± 11.8** 8: **88.0 ± 9.0**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **40.0 ± 13.7** 8: **44.0 ± 13.9**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **28.0 ± 12.6** 8: **72.0 ± 12.5**
CT-NRVQ-L	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **76.0 ± 11.8** 8: **62.0 ± 13.5**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **36.0 ± 13.4** 8: **30.0 ± 12.9**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **14.0 ± 9.6** 8: **4.0 ± 5.4**
C2-NRVQ-S	4: **74.0 ± 12.1** 8: **76.0 ± 11.8**	4: 70.0 ± 12.8 8: **76.0 ± 11.8**	4: **60.0 ± 13.6** 8: 60.0 ± 13.6	4: 58.0 ± 13.7 8: **62.0 ± 13.5**	4: 34.0 ± 13.3 8: 54.0 ± 13.9	4: **38.0 ± 13.6** 8: **60.0 ± 13.6**
C2-NRVQ-L	4: 72.0 ± 12.5 8: 68.0 ± 13.0	4: **74.0 ± 12.1** 8: **70.0 ± 12.8**	4: **46.0 ± 14.0** 8: **36.0 ± 13.4**	4: 44.0 ± 13.9 8: 34.0 ± 13.3	4: 4.0 ± 5.4 8: **12.0 ± 9.0**	4: **12.0 ± 9.0** 8: 8.0 ± 7.6
TF-FSQ-S	4: 86.0 ± 9.5 8: **90.0 ± 8.3**	4: **90.0 ± 8.3** 8: 88.0 ± 9.0	4: **70.0 ± 12.8** 8: 58.0 ± 13.7	4: **70.0 ± 12.8** 8: **72.0 ± 12.5**	4: 50.0 ± 14.1 8: 60.0 ± 13.6	4: **58.0 ± 13.7** 8: **74.0 ± 12.1**
TF-FSQ-L	4: **90.0 ± 8.3** 8: **86.0 ± 9.5**	4: 88.0 ± 9.0 8: **86.0 ± 9.5**	4: **66.0 ± 13.1** 8: 66.0 ± 13.1	4: 60.0 ± 13.6 8: **70.0 ± 12.8**	4: **46.0 ± 14.0** 8: 54.0 ± 13.9	4: 28.0 ± 12.6 8: **66.0 ± 13.1**
TF-QINCO-S	4: 40.0 ± 13.7 8: 30.0 ± 12.9	4: **82.0 ± 10.5** 8: **74.0 ± 12.1**	4: 2.0 ± 3.9 8: 6.0 ± 6.6	4: **52.0 ± 14.0** 8: **52.0 ± 14.0**	4: 2.0 ± 3.9 8: 6.0 ± 6.6	4: **24.0 ± 12.0** 8: **20.0 ± 11.1**
TF-QINCO-L	4: 42.0 ± 13.8 8: 52.0 ± 14.0	4: **70.0 ± 12.8** 8: **68.0 ± 13.0**	4: 10.0 ± 8.4 8: 2.0 ± 3.9	4: **42.0 ± 13.8** 8: **48.0 ± 14.0**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **26.0 ± 12.4** 8: **22.0 ± 11.6**
CT-QINCO-S	4: 38.0 ± 13.6 8: 40.0 ± 13.7	4: **88.0 ± 9.0** 8: **84.0 ± 10.1**	4: 18.0 ± 10.7 8: 12.0 ± 9.0	4: **42.0 ± 13.8** 8: **42.0 ± 13.8**	4: 0.0 ± 0.0 8: 6.0 ± 6.6	4: **24.0 ± 12.0** 8: **22.0 ± 11.6**
CT-QINCO-L	4: 36.0 ± 13.4 8: 32.0 ± 13.1	4: **66.0 ± 13.1** 8: **54.0 ± 13.9**	4: 28.0 ± 12.6 8: 18.0 ± 10.7	4: **60.0 ± 13.6** 8: **58.0 ± 13.7**	4: 6.0 ± 6.6 8: 8.0 ± 7.6	4: **16.0 ± 10.2** 8: **20.0 ± 11.1**
C2-QINCO-S	4: 62.0 ± 13.5 8: 44.0 ± 13.9	4: **88.0 ± 9.0** 8: **78.0 ± 11.4**	4: 40.0 ± 13.7 8: 24.0 ± 12.0	4: **56.0 ± 13.8** 8: **44.0 ± 13.9**	4: **26.0 ± 12.4** 8: 14.0 ± 9.6	4: **26.0 ± 12.4** 8: **20.0 ± 11.1**
C2-QINCO-L	4: **54.0 ± 13.9** 8: **38.0 ± 13.6**	4: 34.0 ± 13.3 8: **38.0 ± 13.6**	4: 8.0 ± 7.6 8: 2.0 ± 3.9	4: **22.0 ± 11.6** 8: **16.0 ± 10.2**	4: 0.0 ± 0.0 8: 6.0 ± 6.6	4: **20.0 ± 11.1** 8: **20.0 ± 11.1**

E.1.3 Robustness to Checkpoint and Data Setting

The primary protocol fixes two reporting choices: it selects the policy checkpoint by teacher-forced validation accuracy and trains one policy jointly on all tasks of a benchmark. Neither choice is forced by the method, so we check both. Because repeating them on every simulator is computationally expensive, we restrict these auxiliary comparisons to Robomimic and keep every tokenizer configuration in each one.

Last checkpoint. Table 20 replaces the checkpoint with the best teacher-forced validation accuracy with the last one while holding multi-task training fixed. Averaging all tokenizer–task pairs, ProAct raises last-checkpoint success from 58.5% to 73.9% at $H_{\text{exec}} = 4$ and from 55.2% to 71.3% at $H_{\text{exec}} = 8$.

Table 20: Task-level multi-task RoboMimic results at the last checkpoint. Each tokenizer occupies two lines for $H_{\text{exec}} = 4$ and 8 success rate $\pm$ 95% confidence-interval half-width (percentage points). R.O. denotes Reconstruction Only; **bold** marks the better objective per task and horizon.

Tokenizer	Can		Lift		Square	
	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct
TF-VQ-S	4: 88.3 ± 3.6 8: 87.3 ± 3.8	4: **93.3 ± 2.9** 8: **93.3 ± 2.8**	4: 81.3 ± 4.4 8: 87.7 ± 3.7	4: **99.0 ± 1.1** 8: **99.0 ± 1.1**	4: 55.3 ± 5.6 8: 56.0 ± 5.6	4: **61.0 ± 5.5** 8: **60.0 ± 5.6**
TF-VQ-L	4: 85.7 ± 4.0 8: **84.7 ± 4.1**	4: **87.7 ± 3.8** 8: 83.7 ± 4.2	4: 91.7 ± 3.1 8: 88.7 ± 3.6	4: **97.7 ± 1.7** 8: **96.7 ± 2.0**	4: 42.7 ± 5.6 8: **45.3 ± 5.6**	4: **44.7 ± 5.6** 8: 33.7 ± 5.4
TF-RVQ-S	4: 89.3 ± 3.5 8: 92.3 ± 3.0	4: **91.0 ± 3.2** 8: **96.0 ± 2.2**	4: **91.7 ± 3.2** 8: 91.3 ± 3.2	4: 90.7 ± 3.3 8: **92.3 ± 3.0**	4: 60.3 ± 5.5 8: 60.7 ± 5.6	4: **69.7 ± 5.3** 8: **65.0 ± 5.4**
TF-RVQ-L	4: 72.7 ± 5.0 8: 66.3 ± 5.4	4: **87.7 ± 3.8** 8: **83.7 ± 4.2**	4: **81.3 ± 4.4** 8: 73.0 ± 4.9	4: 81.0 ± 4.5 8: **81.7 ± 4.4**	4: 38.0 ± 5.5 8: 35.3 ± 5.4	4: **45.3 ± 5.7** 8: **47.7 ± 5.7**
CT-RVQ-S	4: 89.7 ± 3.5 8: 91.7 ± 3.1	4: **93.7 ± 2.8** 8: **97.0 ± 1.9**	4: **91.3 ± 3.2** 8: 93.7 ± 2.8	4: 89.0 ± 3.5 8: **94.3 ± 2.6**	4: **61.3 ± 5.5** 8: 59.7 ± 5.6	4: 59.0 ± 5.6 8: **63.3 ± 5.4**
CT-RVQ-L	4: 75.3 ± 4.9 8: 69.7 ± 5.3	4: **82.0 ± 4.4** 8: **73.0 ± 5.0**	4: 71.0 ± 5.1 8: 76.0 ± 4.8	4: **82.7 ± 4.3** 8: **78.7 ± 4.7**	4: 32.7 ± 5.3 8: 30.0 ± 5.2	4: **45.0 ± 5.6** 8: **32.3 ± 5.3**
C2-RVQ-S	4: 93.7 ± 2.8 8: 90.7 ± 3.3	4: **96.7 ± 2.1** 8: **95.3 ± 2.4**	4: **95.7 ± 2.3** 8: **95.7 ± 2.3**	4: 88.3 ± 3.7 8: 91.7 ± 3.1	4: **61.3 ± 5.5** 8: **68.0 ± 5.3**	4: 58.3 ± 5.6 8: 59.7 ± 5.6
C2-RVQ-L	4: 74.3 ± 4.9 8: 69.0 ± 5.2	4: **78.3 ± 4.6** 8: **76.0 ± 4.9**	4: **77.0 ± 4.8** 8: **73.0 ± 5.0**	4: 74.7 ± 5.0 8: 72.7 ± 5.1	4: 38.3 ± 5.6 8: 26.3 ± 5.0	4: **41.0 ± 5.6** 8: **32.0 ± 5.3**
TF-NRVQ-S	4: 89.0 ± 3.5 8: 85.7 ± 4.0	4: **93.0 ± 2.9** 8: **95.3 ± 2.4**	4: 85.3 ± 4.0 8: **92.3 ± 3.0**	4: **88.3 ± 3.6** 8: 88.7 ± 3.6	4: 64.0 ± 5.4 8: 60.0 ± 5.5	4: **71.0 ± 5.1** 8: **67.7 ± 5.3**
TF-NRVQ-L	4: 63.7 ± 5.4 8: 54.7 ± 5.6	4: **71.0 ± 5.2** 8: **63.3 ± 5.4**	4: 64.3 ± 5.4 8: 63.3 ± 5.5	4: **80.0 ± 4.5** 8: **77.0 ± 4.8**	4: 29.3 ± 5.1 8: 19.0 ± 4.4	4: **42.0 ± 5.6** 8: **28.3 ± 5.1**
CT-NRVQ-S	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **93.3 ± 2.8** 8: **92.3 ± 3.0**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **82.0 ± 4.3** 8: **95.3 ± 2.4**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **57.3 ± 5.6** 8: **64.3 ± 5.5**
CT-NRVQ-L	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **69.7 ± 5.2** 8: **59.3 ± 5.6**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **77.3 ± 4.7** 8: **78.0 ± 4.7**	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **30.0 ± 5.2** 8: **25.7 ± 4.9**
C2-NRVQ-S	4: 89.0 ± 3.5 8: 91.0 ± 3.2	4: **93.3 ± 2.8** 8: **93.0 ± 2.9**	4: **93.3 ± 2.8** 8: **93.3 ± 2.8**	4: 83.7 ± 4.2 8: 89.7 ± 3.5	4: **57.3 ± 5.6** 8: 58.7 ± 5.6	4: **57.3 ± 5.6** 8: **66.7 ± 5.4**
C2-NRVQ-L	4: 68.7 ± 5.3 8: 51.7 ± 5.7	4: **74.7 ± 4.9** 8: **65.3 ± 5.3**	4: **71.7 ± 5.1** 8: 64.7 ± 5.4	4: 71.0 ± 5.1 8: **67.0 ± 5.3**	4: 27.7 ± 5.1 8: 24.7 ± 4.9	4: **34.7 ± 5.4** 8: **29.0 ± 5.1**
TF-FSQ-S	4: 89.3 ± 3.5 8: 87.3 ± 3.7	4: **91.3 ± 3.2** 8: **91.7 ± 3.1**	4: 81.3 ± 4.4 8: 83.0 ± 4.2	4: **91.0 ± 3.2** 8: **90.3 ± 3.4**	4: 50.3 ± 5.7 8: 46.7 ± 5.7	4: **57.0 ± 5.6** 8: **55.7 ± 5.7**
TF-FSQ-L	4: 87.3 ± 3.7 8: 87.0 ± 3.8	4: **94.0 ± 2.7** 8: **92.3 ± 3.0**	4: 92.0 ± 3.1 8: 89.7 ± 3.5	4: **93.7 ± 2.8** 8: **93.3 ± 2.9**	4: 47.3 ± 5.6 8: 43.3 ± 5.6	4: **56.7 ± 5.6** 8: **56.0 ± 5.6**
TF-QINCO-S	4: 54.3 ± 5.6 8: 50.7 ± 5.7	4: **93.0 ± 2.9** 8: **87.0 ± 3.8**	4: 70.7 ± 5.1 8: 64.0 ± 5.4	4: **84.0 ± 4.1** 8: **82.3 ± 4.3**	4: 18.3 ± 4.4 8: 12.0 ± 3.7	4: **58.3 ± 5.7** 8: **50.0 ± 5.6**
TF-QINCO-L	4: 44.7 ± 5.6 8: 36.3 ± 5.5	4: **88.7 ± 3.6** 8: **85.7 ± 3.9**	4: 66.0 ± 5.4 8: 55.7 ± 5.6	4: **82.7 ± 4.3** 8: **85.0 ± 4.0**	4: 13.7 ± 3.9 8: 6.3 ± 2.7	4: **58.0 ± 5.6** 8: **47.3 ± 5.6**
CT-QINCO-S	4: 61.0 ± 5.5 8: 52.7 ± 5.7	4: **85.0 ± 4.0** 8: **77.7 ± 4.7**	4: 74.0 ± 5.0 8: 70.7 ± 5.2	4: **82.0 ± 4.3** 8: **76.3 ± 4.8**	4: 28.0 ± 5.1 8: 20.3 ± 4.6	4: **50.3 ± 5.6** 8: **32.0 ± 5.3**
CT-QINCO-L	4: 55.3 ± 5.7 8: 42.0 ± 5.6	4: **78.7 ± 4.7** 8: **75.7 ± 4.8**	4: 71.7 ± 5.1 8: 68.3 ± 5.3	4: **82.3 ± 4.4** 8: **80.0 ± 4.5**	4: 21.0 ± 4.6 8: 12.0 ± 3.7	4: **47.0 ± 5.6** 8: **35.3 ± 5.4**
C2-QINCO-S	4: 62.0 ± 5.5 8: 46.0 ± 5.6	4: **86.7 ± 3.8** 8: **83.3 ± 4.2**	4: 67.0 ± 5.4 8: 64.3 ± 5.4	4: **79.3 ± 4.6** 8: **79.0 ± 4.6**	4: 15.7 ± 4.1 8: 12.7 ± 3.8	4: **52.0 ± 5.7** 8: **49.7 ± 5.7**
C2-QINCO-L	4: 62.3 ± 5.5 8: 52.3 ± 5.6	4: **65.7 ± 5.4** 8: **64.0 ± 5.4**	4: 64.0 ± 5.5 8: 60.3 ± 5.5	4: **77.3 ± 4.7** 8: **72.0 ± 5.1**	4: 19.0 ± 4.4 8: 7.7 ± 3.0	4: **33.3 ± 5.3** 8: **20.7 ± 4.6**

Single-task policies. Table 21 replaces multi-task training with policies trained and evaluated on one Robomimic task, again at the checkpoint with the best teacher-forced validation accuracy. Unlike the task-macro-averaged main table, these entries retain the confidence interval computed from the 300 binary rollout outcomes.

The primary conclusion also holds under single-task training: averaged equally over all tokenizer–task pairs, ProAct raises success from 68.2% to 78.0% at $H_{\text{exec}} = 4$ and from 63.8% to 76.7% at $H_{\text{exec}} = 8$.

Table 21: RoboMimic results for policies trained on the corresponding single task, at the checkpoint with the best teacher-forced validation accuracy. Each tokenizer occupies two lines for $H_{\text{exec}} = 4$ and 8 success rate ± 95% confidence-interval half-width (percentage points). R.O. denotes Reconstruction Only; **bold** marks the better objective per task and horizon.

Tokenizer	Can		Lift		Square	
	R.O.	ProAct	R.O.	ProAct	R.O.	ProAct
TF-VQ-S	4: 90.3 ± 3.3 8: **93.3 ± 2.8**	4: **90.7 ± 3.3** 8: 92.0 ± 3.1	4: 93.0 ± 2.9 8: 93.0 ± 2.9	4: **99.0 ± 1.1** 8: **99.3 ± 0.9**	4: 64.3 ± 5.4 8: **69.3 ± 5.2**	4: **74.7 ± 4.9** 8: **69.3 ± 5.2**
TF-VQ-L	4: 93.3 ± 2.8 8: 93.0 ± 2.9	4: **93.7 ± 2.8** 8: **94.3 ± 2.6**	4: 96.3 ± 2.1 8: 95.0 ± 2.5	4: **98.7 ± 1.3** 8: **98.7 ± 1.3**	4: 50.7 ± 5.7 8: 61.0 ± 5.5	4: **63.7 ± 5.5** 8: **62.0 ± 5.4**
TF-RVQ-S	4: **94.0 ± 2.7** 8: 93.0 ± 2.9	4: 92.7 ± 2.9 8: **93.7 ± 2.8**	4: 84.7 ± 4.1 8: 87.7 ± 3.7	4: **92.3 ± 3.1** 8: **96.7 ± 2.0**	4: 63.7 ± 5.4 8: **71.7 ± 5.1**	4: **66.7 ± 5.3** 8: 69.7 ± 5.2
TF-RVQ-L	4: 58.3 ± 5.6 8: 46.7 ± 5.7	4: **83.3 ± 4.2** 8: **73.7 ± 4.9**	4: 86.3 ± 3.9 8: 74.7 ± 4.9	4: **91.0 ± 3.2** 8: **91.3 ± 3.1**	4: 51.0 ± 5.6 8: 27.0 ± 4.9	4: **53.3 ± 5.6** 8: **49.0 ± 5.7**
CT-RVQ-S	4: 85.0 ± 4.1 8: 89.3 ± 3.5	4: **89.7 ± 3.4** 8: **95.7 ± 2.3**	4: 85.7 ± 4.0 8: 92.7 ± 3.0	4: **93.3 ± 2.8** 8: **99.7 ± 0.7**	4: **69.0 ± 5.2** 8: **73.0 ± 5.1**	4: 65.0 ± 5.4 8: 71.3 ± 5.1
CT-RVQ-L	4: 73.7 ± 5.0 8: 60.0 ± 5.5	4: **81.7 ± 4.4** 8: **73.3 ± 5.0**	4: **93.0 ± 2.9** 8: 88.3 ± 3.6	4: 89.0 ± 3.5 8: **90.3 ± 3.3**	4: 47.0 ± 5.6 8: 33.0 ± 5.2	4: **47.7 ± 5.7** 8: **41.3 ± 5.6**
C2-RVQ-S	4: 87.0 ± 3.8 8: 87.0 ± 3.9	4: **94.7 ± 2.6** 8: **94.3 ± 2.6**	4: **93.0 ± 2.9** 8: 96.0 ± 2.2	4: 90.0 ± 3.4 8: **97.0 ± 1.9**	4: 52.0 ± 5.6 8: **70.3 ± 5.2**	4: **58.3 ± 5.6** 8: 63.7 ± 5.4
C2-RVQ-L	4: 70.0 ± 5.2 8: 64.7 ± 5.4	4: **80.3 ± 4.5** 8: **77.0 ± 4.8**	4: **90.3 ± 3.4** 8: **89.7 ± 3.5**	4: 84.7 ± 4.1 8: 85.3 ± 4.0	4: 48.0 ± 5.6 8: 34.7 ± 5.4	4: **52.0 ± 5.7** 8: **49.0 ± 5.7**
TF-NRVQ-S	4: 94.3 ± 2.6 8: 92.0 ± 3.1	4: **94.7 ± 2.5** 8: **95.3 ± 2.4**	4: 94.7 ± 2.6 8: **97.7 ± 1.7**	4: **95.7 ± 2.3** 8: 96.7 ± 2.0	4: 68.0 ± 5.2 8: 67.7 ± 5.4	4: **76.0 ± 4.8** 8: **77.0 ± 4.8**
TF-NRVQ-L	4: 64.7 ± 5.4 8: 48.3 ± 5.6	4: **74.0 ± 5.0** 8: **63.3 ± 5.5**	4: 68.7 ± 5.3 8: 70.0 ± 5.3	4: **86.0 ± 3.9** 8: **83.3 ± 4.2**	4: 33.3 ± 5.3 8: 19.3 ± 4.5	4: **34.3 ± 5.4** 8: **24.0 ± 4.8**
CT-NRVQ-S	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **93.3 ± 2.8** 8: **93.3 ± 2.8**	4: 90.0 ± 3.4 8: **94.7 ± 2.5**	4: **93.3 ± 2.8** 8: 89.3 ± 3.5	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **47.0 ± 5.7** 8: **65.7 ± 5.4**
CT-NRVQ-L	4: 56.3 ± 5.7 8: 37.3 ± 5.5	4: **62.7 ± 5.5** 8: **46.3 ± 5.7**	4: 77.0 ± 4.8 8: **74.3 ± 4.9**	4: **77.7 ± 4.7** 8: 67.3 ± 5.2	4: 24.7 ± 5.0 8: 14.3 ± 4.0	4: **28.7 ± 5.1** 8: **21.0 ± 4.6**
C2-NRVQ-S	4: **94.7 ± 2.5** 8: 87.7 ± 3.7	4: 89.3 ± 3.5 8: **91.3 ± 3.2**	4: 93.0 ± 2.9 8: **98.7 ± 1.3**	4: **94.7 ± 2.5** 8: 95.3 ± 2.4	4: 61.0 ± 5.5 8: 59.3 ± 5.5	4: **65.7 ± 5.4** 8: **63.3 ± 5.5**
C2-NRVQ-L	4: 67.0 ± 5.4 8: 50.3 ± 5.6	4: **70.7 ± 5.1** 8: **61.7 ± 5.5**	4: **79.7 ± 4.5** 8: **76.0 ± 4.8**	4: 78.0 ± 4.7 8: 74.7 ± 4.9	4: **34.0 ± 5.3** 8: 25.0 ± 4.8	4: 31.0 ± 5.2 8: **28.7 ± 5.1**
TF-FSQ-S	4: 94.0 ± 2.7 8: **95.0 ± 2.5**	4: **95.3 ± 2.4** 8: 93.0 ± 2.9	4: 93.7 ± 2.8 8: 97.3 ± 1.8	4: **98.0 ± 1.6** 8: **98.3 ± 1.4**	4: **74.7 ± 5.0** 8: **71.7 ± 5.1**	4: 72.0 ± 5.1 8: 71.3 ± 5.1
TF-FSQ-L	4: 93.7 ± 2.7 8: **91.7 ± 3.1**	4: **94.7 ± 2.5** 8: 91.0 ± 3.3	4: 96.3 ± 2.1 8: **99.3 ± 0.9**	4: **97.7 ± 1.7** 8: 98.3 ± 1.4	4: 63.3 ± 5.4 8: **73.3 ± 5.0**	4: **65.0 ± 5.4** 8: 66.3 ± 5.3
TF-QINCO-S	4: 70.0 ± 5.2 8: 52.3 ± 5.7	4: **80.7 ± 4.5** 8: **76.7 ± 4.7**	4: 77.7 ± 4.7 8: 74.0 ± 4.9	4: **92.0 ± 3.1** 8: **91.3 ± 3.2**	4: 36.3 ± 5.4 8: 18.0 ± 4.3	4: **57.3 ± 5.6** 8: **54.3 ± 5.6**
TF-QINCO-L	4: 58.7 ± 5.6 8: 41.7 ± 5.6	4: **87.3 ± 3.8** 8: **82.0 ± 4.3**	4: 79.3 ± 4.6 8: 72.0 ± 5.1	4: **91.3 ± 3.2** 8: **92.0 ± 3.1**	4: 19.3 ± 4.5 8: 11.0 ± 3.5	4: **60.7 ± 5.5** 8: **66.3 ± 5.4**
CT-QINCO-S	4: 73.0 ± 5.0 8: 57.0 ± 5.6	4: **86.0 ± 3.9** 8: **87.7 ± 3.7**	4: 89.0 ± 3.6 8: 88.3 ± 3.6	4: **95.7 ± 2.3** 8: **97.0 ± 1.9**	4: 41.7 ± 5.6 8: 28.0 ± 5.1	4: **60.3 ± 5.6** 8: **53.7 ± 5.6**
CT-QINCO-L	4: 62.0 ± 5.4 8: 51.3 ± 5.7	4: **87.7 ± 3.8** 8: **84.7 ± 4.1**	4: 83.0 ± 4.2 8: 79.7 ± 4.5	4: **88.3 ± 3.7** 8: **86.7 ± 3.9**	4: 24.3 ± 4.9 8: 14.7 ± 4.0	4: **58.0 ± 5.6** 8: **55.0 ± 5.6**
C2-QINCO-S	4: 67.7 ± 5.3 8: 54.0 ± 5.7	4: **92.3 ± 3.0** 8: **88.0 ± 3.6**	4: **90.7 ± 3.3** 8: 80.0 ± 4.6	4: 89.7 ± 3.5 8: **96.0 ± 2.2**	4: 14.0 ± 3.9 8: 12.0 ± 3.7	4: **51.7 ± 5.6** 8: **53.7 ± 5.6**
C2-QINCO-L	4: 65.0 ± 5.4 8: 54.0 ± 5.7	4: **87.3 ± 3.7** 8: **80.7 ± 4.4**	4: 84.0 ± 4.2 8: 81.7 ± 4.4	4: **87.0 ± 3.8** 8: **87.3 ± 3.7**	4: 26.7 ± 5.1 8: 18.7 ± 4.4	4: **51.3 ± 5.7** 8: **48.3 ± 5.7**

E.2 Detailed Ablation Results

Tables 3 and 22 summarize the results overall and by quantizer family. Tables 23–25 report all four training conditions on Can, Lift, and Square, using the checkpoint with the best teacher-forced validation accuracy and both execution horizons.

Table 22: RoboMimic component ablation on the 20-configuration cohort. Entries give success (%) at the checkpoint with the best teacher-forced validation accuracy at $H_{\text{exec}} = 4 \,/\, 8$, averaged equally over tasks and configurations within each family. **Bold** marks the best objective at each horizon. `CT-NRVQ-S/L` collapsed under Reconstruction-only training; these configurations are excluded from the aggregate calculation.

Quantizer	R.O.	w/o TBR	w/o TP	ProAct
VQ	74.7 / 76.1	79.1 / 74.6	**84.3 / 83.8**	81.8 / 79.4
RVQ	71.2 / 69.0	75.3 / 73.6	**77.3** / 74.1	75.5 / **74.9**
NRVQ	68.4 / 65.0	**73.1** / 68.0	71.3 / 67.9	71.2 / **69.1**
FSQ	76.2 / 74.4	77.9 / 76.0	**83.3 / 83.1**	80.8 / 82.1
QinCo	50.8 / 40.8	63.7 / 57.2	65.2 / 55.7	**71.1 / 66.2**
Overall	65.4 / 61.0	72.0 / 67.9	73.8 / 69.2	**74.5 / 72.3**

Table 23: Multi-task RoboMimic ablation on Can for all 22 configurations, using the checkpoint with the best teacher-forced validation accuracy. The two rows per tokenizer report $H_{\text{exec}} = 4$ and 8 success (%) $\pm$ 95% confidence-interval half-width. TP and TBR denote Token Perturbation and Token Balance Regularization; **bold** marks the best treatment per horizon.

Tokenizer	Reconstruction-only	w/o TBR	w/o TP	ProAct
TF-VQ-S	4: 86.7 ± 3.8 8: 91.0 ± 3.2	4: 88.0 ± 3.7 8: 89.3 ± 3.5	4: 94.3 ± 2.6 8: 89.0 ± 3.6	4: **95.3 ± 2.4** 8: **95.3 ± 2.4**
TF-VQ-L	4: 87.7 ± 3.7 8: 86.0 ± 3.9	4: 89.3 ± 3.5 8: 80.0 ± 4.5	4: **94.0 ± 2.7** 8: **94.7 ± 2.5**	4: 90.7 ± 3.3 8: 86.3 ± 3.9
TF-RVQ-S	4: 86.0 ± 3.9 8: 90.7 ± 3.3	4: 97.0 ± 1.9 8: 95.0 ± 2.5	4: **98.3 ± 1.4** 8: **95.3 ± 2.4**	4: 92.0 ± 3.1 8: 94.0 ± 2.7
TF-RVQ-L	4: 73.0 ± 5.1 8: 63.7 ± 5.4	4: 80.7 ± 4.5 8: 74.7 ± 4.9	4: 82.3 ± 4.4 8: 74.7 ± 4.9	4: **86.7 ± 3.9** 8: **80.7 ± 4.5**
CT-RVQ-S	4: 91.0 ± 3.3 8: 91.0 ± 3.3	4: 94.3 ± 2.6 8: 95.7 ± 2.3	4: **96.7 ± 2.0** 8: **96.0 ± 2.2**	4: 95.0 ± 2.5 8: **96.0 ± 2.2**
CT-RVQ-L	4: 75.3 ± 4.8 8: 69.0 ± 5.2	4: 78.7 ± 4.6 8: 74.7 ± 5.0	4: **85.7 ± 4.0** 8: 72.3 ± 5.0	4: 80.0 ± 4.5 8: **77.7 ± 4.7**
C2-RVQ-S	4: 94.0 ± 2.7 8: 92.0 ± 3.1	4: 92.7 ± 2.9 8: **95.7 ± 2.3**	4: 94.3 ± 2.6 8: 93.7 ± 2.8	4: **97.7 ± 1.7** 8: 94.3 ± 2.6
C2-RVQ-L	4: 71.7 ± 5.1 8: 66.0 ± 5.3	4: 82.3 ± 4.4 8: 71.3 ± 5.1	4: **82.7 ± 4.3** 8: **74.0 ± 5.0**	4: 80.0 ± 4.5 8: **74.0 ± 5.0**
TF-NRVQ-S	4: 89.3 ± 3.5 8: 90.0 ± 3.4	4: **95.0 ± 2.5** 8: 94.0 ± 2.7	4: 94.0 ± 2.7 8: **95.7 ± 2.3**	4: 94.3 ± 2.6 8: **95.7 ± 2.3**
TF-NRVQ-L	4: 70.3 ± 5.1 8: 60.0 ± 5.5	4: **81.0 ± 4.4** 8: **67.7 ± 5.2**	4: 60.7 ± 5.5 8: 45.3 ± 5.6	4: 73.0 ± 5.0 8: 65.3 ± 5.4
CT-NRVQ-S	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: 91.3 ± 3.2 8: **94.3 ± 2.6**	4: **94.3 ± 2.6** 8: 94.0 ± 2.7	4: 91.0 ± 3.2 8: 92.3 ± 3.0
CT-NRVQ-L	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **75.3 ± 4.9** 8: **64.0 ± 5.6**	4: 72.3 ± 5.1 8: 61.7 ± 5.5	4: 73.0 ± 5.0 8: 59.0 ± 5.5
C2-NRVQ-S	4: 91.7 ± 3.1 8: 92.3 ± 3.0	4: 89.3 ± 3.5 8: 91.7 ± 3.2	4: **95.3 ± 2.4** 8: **96.3 ± 2.1**	4: 89.3 ± 3.5 8: 93.3 ± 2.8
C2-NRVQ-L	4: 68.7 ± 5.3 8: 54.0 ± 5.7	4: 72.3 ± 5.0 8: **69.3 ± 5.3**	4: **76.7 ± 4.8** 8: 65.0 ± 5.4	4: **76.7 ± 4.8** 8: 62.3 ± 5.5
TF-FSQ-S	4: 91.0 ± 3.2 8: 89.7 ± 3.5	4: 90.7 ± 3.3 8: 90.3 ± 3.3	4: **95.3 ± 2.4** 8: **94.3 ± 2.6**	4: 89.0 ± 3.5 8: 92.3 ± 3.0
TF-FSQ-L	4: 91.0 ± 3.2 8: 87.3 ± 3.8	4: 93.3 ± 2.8 8: 91.0 ± 3.2	4: 92.7 ± 2.9 8: 92.0 ± 3.1	4: **94.0 ± 2.7** 8: **92.7 ± 3.0**
TF-QINCO-S	4: 55.3 ± 5.7 8: 47.3 ± 5.6	4: 75.7 ± 4.9 8: 73.3 ± 5.0	4: 85.3 ± 3.9 8: 66.0 ± 5.3	4: **89.7 ± 3.4** 8: **86.3 ± 3.9**
TF-QINCO-L	4: 51.0 ± 5.6 8: 32.7 ± 5.4	4: 68.0 ± 5.2 8: 51.3 ± 5.6	4: 64.0 ± 5.4 8: 47.7 ± 5.6	4: **86.3 ± 3.9** 8: **85.7 ± 4.0**
CT-QINCO-S	4: 67.0 ± 5.3 8: 56.3 ± 5.7	4: 80.7 ± 4.4 8: **77.7 ± 4.7**	4: 75.7 ± 4.8 8: 58.7 ± 5.6	4: **86.0 ± 4.0** 8: 77.3 ± 4.7
CT-QINCO-L	4: 59.0 ± 5.6 8: 45.0 ± 5.6	4: 69.3 ± 5.2 8: 62.0 ± 5.4	4: 77.3 ± 4.7 8: 60.0 ± 5.6	4: **79.7 ± 4.6** 8: **74.7 ± 4.9**
C2-QINCO-S	4: 59.7 ± 5.5 8: 50.0 ± 5.7	4: 79.3 ± 4.5 8: 73.3 ± 5.0	4: 85.3 ± 4.0 8: 75.7 ± 4.9	4: **91.3 ± 3.2** 8: **87.7 ± 3.8**
C2-QINCO-L	4: 61.7 ± 5.5 8: 47.0 ± 5.6	4: 73.0 ± 5.0 8: 66.7 ± 5.4	4: **79.3 ± 4.6** 8: **68.3 ± 5.2**	4: 73.3 ± 4.9 8: 61.7 ± 5.5

Table 24: Multi-task RoboMimic ablation on Lift for all 22 configurations, using the checkpoint with the best teacher-forced validation accuracy. The two rows per tokenizer report $H_{\text{exec}} = 4$ and 8 success (%) $\pm$ 95% confidence-interval half-width. TP and TBR denote Token Perturbation and Token Balance Regularization; **bold** marks the best treatment per horizon.

Tokenizer	Reconstruction-only	w/o TBR	w/o TP	ProAct
TF-VQ-S	4: 82.3 ± 4.3 8: 85.7 ± 4.0	4: 90.7 ± 3.3 8: 86.0 ± 3.9	4: 97.3 ± 1.8 8: 98.7 ± 1.3	4: **99.0 ± 1.1** 8: **100.0 ± 0.0**
TF-VQ-L	4: 93.7 ± 2.7 8: 87.0 ± 3.8	4: 85.3 ± 4.0 8: 82.7 ± 4.3	4: **99.7 ± 0.7** 8: **100.0 ± 0.0**	4: 98.7 ± 1.3 8: 98.3 ± 1.4
TF-RVQ-S	4: 91.3 ± 3.2 8: 90.3 ± 3.3	4: 92.0 ± 3.1 8: 95.7 ± 2.3	4: **98.3 ± 1.4** 8: **98.3 ± 1.5**	4: 92.0 ± 3.1 8: 94.3 ± 2.6
TF-RVQ-L	4: 70.7 ± 5.2 8: 77.0 ± 4.7	4: 81.7 ± 4.4 8: 79.7 ± 4.6	4: **89.0 ± 3.5** 8: **88.0 ± 3.7**	4: 79.3 ± 4.6 8: 81.0 ± 4.4
CT-RVQ-S	4: 92.0 ± 3.1 8: 92.3 ± 3.0	4: 93.7 ± 2.7 8: 93.3 ± 2.8	4: **100.0 ± 0.0** 8: **99.7 ± 0.7**	4: 88.7 ± 3.6 8: 95.0 ± 2.4
CT-RVQ-L	4: 74.7 ± 4.9 8: 76.0 ± 4.9	4: 80.3 ± 4.5 8: 80.3 ± 4.5	4: **95.3 ± 2.4** 8: **94.3 ± 2.6**	4: 80.0 ± 4.5 8: 80.3 ± 4.5
C2-RVQ-S	4: 97.3 ± 1.8 8: 96.0 ± 2.2	4: 89.7 ± 3.4 8: 91.0 ± 3.2	4: **98.3 ± 1.4** 8: **100.0 ± 0.0**	4: 90.7 ± 3.3 8: 93.0 ± 2.9
C2-RVQ-L	4: 70.7 ± 5.2 8: 75.3 ± 4.9	4: 79.0 ± 4.6 8: 83.0 ± 4.3	4: **93.7 ± 2.8** 8: **89.7 ± 3.5**	4: 73.7 ± 5.0 8: 74.7 ± 4.9
TF-NRVQ-S	4: 89.0 ± 3.5 8: 88.3 ± 3.6	4: 97.3 ± 1.8 8: 98.0 ± 1.6	4: **98.3 ± 1.4** 8: **100.0 ± 0.0**	4: 88.7 ± 3.6 8: 88.0 ± 3.6
TF-NRVQ-L	4: 65.3 ± 5.3 8: 64.0 ± 5.4	4: **77.0 ± 4.8** 8: 66.3 ± 5.4	4: **77.0 ± 4.8** 8: **80.0 ± 4.6**	4: 71.7 ± 5.1 8: 75.0 ± 4.9
CT-NRVQ-S	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: 87.7 ± 3.7 8: 94.7 ± 2.5	4: **98.0 ± 1.6** 8: **100.0 ± 0.0**	4: 88.3 ± 3.6 8: 96.7 ± 2.0
CT-NRVQ-L	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: 74.3 ± 4.9 8: 76.0 ± 4.8	4: **83.7 ± 4.1** 8: **86.7 ± 3.9**	4: 79.3 ± 4.6 8: 79.0 ± 4.6
C2-NRVQ-S	4: 90.0 ± 3.4 8: 95.3 ± 2.4	4: 89.3 ± 3.4 8: 86.3 ± 3.9	4: **99.3 ± 0.9** 8: **99.7 ± 0.7**	4: 85.0 ± 4.0 8: 87.7 ± 3.7
C2-NRVQ-L	4: 71.7 ± 5.1 8: 68.7 ± 5.2	4: 80.0 ± 4.5 8: 70.3 ± 5.1	4: **88.7 ± 3.6** 8: **86.7 ± 3.8**	4: 72.7 ± 5.1 8: 69.0 ± 5.2
TF-FSQ-S	4: 83.3 ± 4.2 8: 85.0 ± 4.0	4: 94.0 ± 2.7 8: 91.0 ± 3.2	4: **96.7 ± 2.1** 8: **98.7 ± 1.3**	4: 91.7 ± 3.1 8: 93.0 ± 2.9
TF-FSQ-L	4: 88.7 ± 3.6 8: 84.7 ± 4.1	4: 87.7 ± 3.7 8: 88.7 ± 3.6	4: **96.7 ± 2.0** 8: **98.0 ± 1.6**	4: 91.0 ± 3.3 8: 94.0 ± 2.7
TF-QINCO-S	4: 62.3 ± 5.5 8: 61.3 ± 5.5	4: 76.7 ± 4.8 8: 72.3 ± 5.1	4: **91.7 ± 3.1** 8: **89.3 ± 3.5**	4: 83.7 ± 4.2 8: 84.0 ± 4.2
TF-QINCO-L	4: 64.7 ± 5.5 8: 56.3 ± 5.6	4: 72.0 ± 5.1 8: 70.0 ± 5.2	4: 80.7 ± 4.5 8: 73.0 ± 5.0	4: **82.7 ± 4.3** 8: **82.0 ± 4.3**
CT-QINCO-S	4: 74.0 ± 4.9 8: 65.3 ± 5.5	4: 82.3 ± 4.3 8: 78.0 ± 4.7	4: **93.3 ± 2.8** 8: **89.3 ± 3.5**	4: 83.7 ± 4.2 8: 77.0 ± 4.8
CT-QINCO-L	4: 73.7 ± 5.0 8: 66.3 ± 5.3	4: 74.3 ± 4.9 8: 67.3 ± 5.4	4: **86.7 ± 3.8** 8: **86.7 ± 3.9**	4: 85.7 ± 4.0 8: 76.7 ± 4.8
C2-QINCO-S	4: 70.0 ± 5.2 8: 64.3 ± 5.4	4: 84.7 ± 4.2 8: 78.7 ± 4.6	4: **96.7 ± 2.0** 8: **96.7 ± 2.0**	4: 79.3 ± 4.5 8: 78.0 ± 4.7
C2-QINCO-L	4: 72.3 ± 5.1 8: 63.3 ± 5.5	4: 74.0 ± 4.9 8: 73.0 ± 5.0	4: **93.3 ± 2.8** 8: **91.0 ± 3.3**	4: 80.0 ± 4.5 8: 74.0 ± 5.0

Table 25: Multi-task RoboMimic ablation on Square for all 22 configurations, using the checkpoint with the best teacher-forced validation accuracy. The two rows per tokenizer report $H_{\text{exec}} = 4$ and 8 success (%) $\pm$ 95% confidence-interval half-width. TP and TBR denote Token Perturbation and Token Balance Regularization; **bold** marks the best treatment per horizon.

Tokenizer	Reconstruction-only	w/o TBR	w/o TP	ProAct
TF-VQ-S	4: 51.3 ± 5.7 8: **64.0 ± 5.4**	4: **65.3 ± 5.4** 8: 58.0 ± 5.6	4: 59.7 ± 5.6 8: 58.0 ± 5.5	4: 64.0 ± 5.5 8: 63.3 ± 5.4
TF-VQ-L	4: 46.3 ± 5.7 8: 42.7 ± 5.6	4: 55.7 ± 5.6 8: 51.7 ± 5.7	4: **61.0 ± 5.6** 8: **62.3 ± 5.5**	4: 43.3 ± 5.6 8: 33.3 ± 5.3
TF-RVQ-S	4: 60.3 ± 5.5 8: 54.3 ± 5.7	4: 63.3 ± 5.5 8: 68.0 ± 5.3	4: 60.0 ± 5.6 8: 65.7 ± 5.4	4: **69.0 ± 5.3** 8: **70.3 ± 5.1**
TF-RVQ-L	4: 36.0 ± 5.4 8: 26.0 ± 4.9	4: 44.0 ± 5.6 8: 34.0 ± 5.4	4: 32.0 ± 5.3 8: 23.0 ± 4.8	4: **53.3 ± 5.7** 8: **43.3 ± 5.6**
CT-RVQ-S	4: 61.7 ± 5.5 8: 55.3 ± 5.6	4: **67.0 ± 5.3** 8: **65.0 ± 5.4**	4: 61.0 ± 5.5 8: 63.7 ± 5.5	4: 59.0 ± 5.6 8: 64.3 ± 5.5
CT-RVQ-L	4: 36.0 ± 5.4 8: 32.7 ± 5.3	4: 39.0 ± 5.5 8: 33.7 ± 5.4	4: 37.0 ± 5.5 8: 27.3 ± 5.0	4: **40.7 ± 5.5** 8: **36.3 ± 5.4**
C2-RVQ-S	4: **64.0 ± 5.5** 8: 61.7 ± 5.5	4: 57.3 ± 5.6 8: 60.7 ± 5.4	4: 60.3 ± 5.6 8: 57.0 ± 5.5	4: 57.7 ± 5.6 8: **64.0 ± 5.5**
C2-RVQ-L	4: 35.3 ± 5.4 8: 32.3 ± 5.3	4: 42.0 ± 5.6 8: 34.3 ± 5.4	4: 26.7 ± 5.0 8: 21.0 ± 4.6	4: **44.0 ± 5.6** 8: **35.7 ± 5.5**
TF-NRVQ-S	4: 61.7 ± 5.5 8: 63.7 ± 5.5	4: 68.3 ± 5.2 8: 65.3 ± 5.4	4: 64.3 ± 5.5 8: 59.3 ± 5.6	4: **70.0 ± 5.2** 8: **69.3 ± 5.2**
TF-NRVQ-L	4: 34.0 ± 5.3 8: 21.0 ± 4.6	4: 31.0 ± 5.2 8: 22.0 ± 4.7	4: 11.0 ± 3.6 8: 9.3 ± 3.3	4: **37.0 ± 5.5** 8: **31.0 ± 5.2**
CT-NRVQ-S	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: 61.7 ± 5.6 8: 67.0 ± 5.3	4: 56.3 ± 5.7 8: 55.0 ± 5.6	4: **64.0 ± 5.5** 8: **70.7 ± 5.1**
CT-NRVQ-L	4: 0.0 ± 0.0 8: 0.0 ± 0.0	4: **41.0 ± 5.5** 8: **29.0 ± 5.2**	4: 26.7 ± 5.0 8: 20.0 ± 4.5	4: 32.7 ± 5.3 8: 24.0 ± 4.8
C2-NRVQ-S	4: 60.7 ± 5.5 8: 61.0 ± 5.5	4: 62.0 ± 5.5 8: 57.3 ± 5.6	4: **64.3 ± 5.5** 8: 64.3 ± 5.4	4: 61.0 ± 5.6 8: **67.7 ± 5.3**
C2-NRVQ-L	4: 28.7 ± 5.1 8: 21.3 ± 4.6	4: **35.0 ± 5.4** 8: **28.0 ± 5.1**	4: 25.7 ± 4.9 8: 13.0 ± 3.8	4: **35.0 ± 5.4** 8: 24.7 ± 4.9
TF-FSQ-S	4: 49.7 ± 5.6 8: 49.7 ± 5.6	4: 53.7 ± 5.6 8: 49.0 ± 5.6	4: **61.7 ± 5.5** 8: **58.0 ± 5.6**	4: 60.0 ± 5.5 8: 57.0 ± 5.6
TF-FSQ-L	4: 53.3 ± 5.7 8: 50.0 ± 5.7	4: 48.3 ± 5.6 8: 46.0 ± 5.7	4: 56.7 ± 5.6 8: 57.3 ± 5.5	4: **59.3 ± 5.6** 8: **63.7 ± 5.4**
TF-QINCO-S	4: 27.7 ± 5.0 8: 12.7 ± 3.8	4: 40.0 ± 5.5 8: 32.3 ± 5.3	4: 28.3 ± 5.1 8: 20.0 ± 4.5	4: **54.0 ± 5.6** 8: **51.0 ± 5.7**
TF-QINCO-L	4: 19.0 ± 4.4 8: 8.7 ± 3.1	4: 29.7 ± 5.2 8: 21.7 ± 4.6	4: 15.0 ± 4.0 8: 5.7 ± 2.6	4: **53.0 ± 5.6** 8: **48.7 ± 5.6**
CT-QINCO-S	4: 34.0 ± 5.5 8: 18.7 ± 4.4	4: **46.7 ± 5.6** 8: **44.3 ± 5.7**	4: 30.0 ± 5.2 8: 17.3 ± 4.3	4: 43.0 ± 5.7 8: 36.7 ± 5.5
CT-QINCO-L	4: 23.7 ± 4.8 8: 17.3 ± 4.3	4: 31.0 ± 5.2 8: 21.7 ± 4.6	4: 22.3 ± 4.7 8: 12.7 ± 3.8	4: **48.3 ± 5.7** 8: **41.0 ± 5.7**
C2-QINCO-S	4: 21.3 ± 4.7 8: 11.3 ± 3.6	4: **52.0 ± 5.6** 8: 34.7 ± 5.4	4: 42.7 ± 5.6 8: 30.0 ± 5.1	4: 50.3 ± 5.6 8: **48.0 ± 5.7**
C2-QINCO-L	4: 17.7 ± 4.3 8: 9.7 ± 3.3	4: **37.7 ± 5.5** 8: **32.0 ± 5.2**	4: 26.3 ± 5.0 8: 15.0 ± 4.0	4: 29.0 ± 5.1 8: 21.3 ± 4.7

E.3 Detailed Policy-Prediction Analysis

We isolate TBR by comparing ProAct with w/o TBR across all 22 valid tokenizer pairs. Let $\mathcal{L}_{\text{val}}^{m}$ denote the teacher-forced policy validation loss at the checkpoint with the best teacher-forced validation accuracy for tokenizer treatment m:

$$\Delta\mathcal{L}_{\text{val}}^{\text{TBR}} = \mathcal{L}_{\text{val}}^{\text{ProAct}} - \mathcal{L}_{\text{val}}^{\text{w/o TBR}}. \tag{63}$$

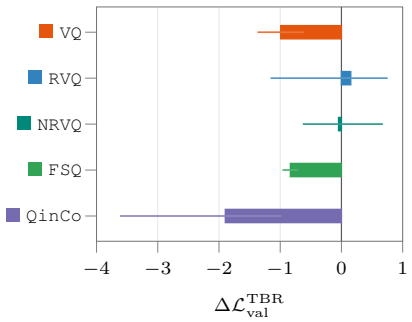


Figure 19: Family-mean validation-loss change after adding TBR; negative favors ProAct. Spans show min–max changes across tokenizers within each family.

Relative to w/o TBR, ProAct lowers mean validation loss by 0.655 and raises mean token accuracy by 1.88 percentage points. Each metric improves for 14 of 22 tokenizers. Family-mean loss decreases for VQ, NRVQ, FSQ, and QinCo, while RVQ increases slightly (Figure 19).

Table 26 reports averaged policy metrics; the curves below instead report per-position NLL normalized by $\log K$. Tokenizer treatments change the encoded targets, so these metrics describe each representation's induced prediction difficulty, rather than losses against identical labels.

Table 26: Policy validation metrics for all 22 multi-task RoboMimic tokenizers at the checkpoint with the best teacher-forced validation accuracy. Lower loss and higher accuracy are better; **bold** marks the better value in each pair. A green down-arrow marks ProAct values whose tokenizer also has lower validation reconstruction error.

Tokenizer	Validation loss		Validation accuracy (%)	
	w/o TBR	ProAct	w/o TBR	ProAct
TF-VQ-S	4.646	**3.276**	37.7	**44.3**
TF-VQ-L	2.064	**1.449**	68.3	**74.7**
TF-RVQ-S	**3.773**	4.529	**44.7**	35.8
TF-RVQ-L	**6.531**	6.793↓	**40.9**	33.7↓
CT-RVQ-S	**4.443**	4.736↓	**43.9**	42.8↓
CT-RVQ-L	**6.630**	7.336↓	**40.8**	35.2↓
C2-RVQ-S	4.704	**3.547**	43.9	**45.4**
C2-RVQ-L	**6.351**	6.405↓	**41.8**	41.5↓
TF-NRVQ-S	5.343	**5.191**↓	**39.3**	36.6↓
TF-NRVQ-L	9.577	**9.209**↓	20.2	**23.3**↓
CT-NRVQ-S	4.675	**4.044**	**42.5**	42.1
CT-NRVQ-L	**9.514**	9.605	19.3	**20.9**
C2-NRVQ-S	**4.000**	4.676↓	**44.5**	43.1↓
C2-NRVQ-L	**9.196**	9.296	20.2	**23.3**
TF-FSQ-S	3.481	**2.776**	41.9	**54.7**
TF-FSQ-L	4.013	**3.050**	43.7	**48.5**
TF-QINCO-S	5.575	**4.311**	40.1	**44.7**
TF-QINCO-L	7.770	**4.154**	35.1	**45.8**
CT-QINCO-S	5.899	**4.347**	40.6	**43.9**
CT-QINCO-L	6.812	**5.022**	40.8	**42.0**
C2-QINCO-S	5.438	**4.458**	42.0	**44.6**
C2-QINCO-L	6.957	**4.778**	37.9	**44.6**

Per-tokenizer token-prediction profiles. We show the paired normalized-NLL profiles underlying Figure 3, retaining its tokenizer order.

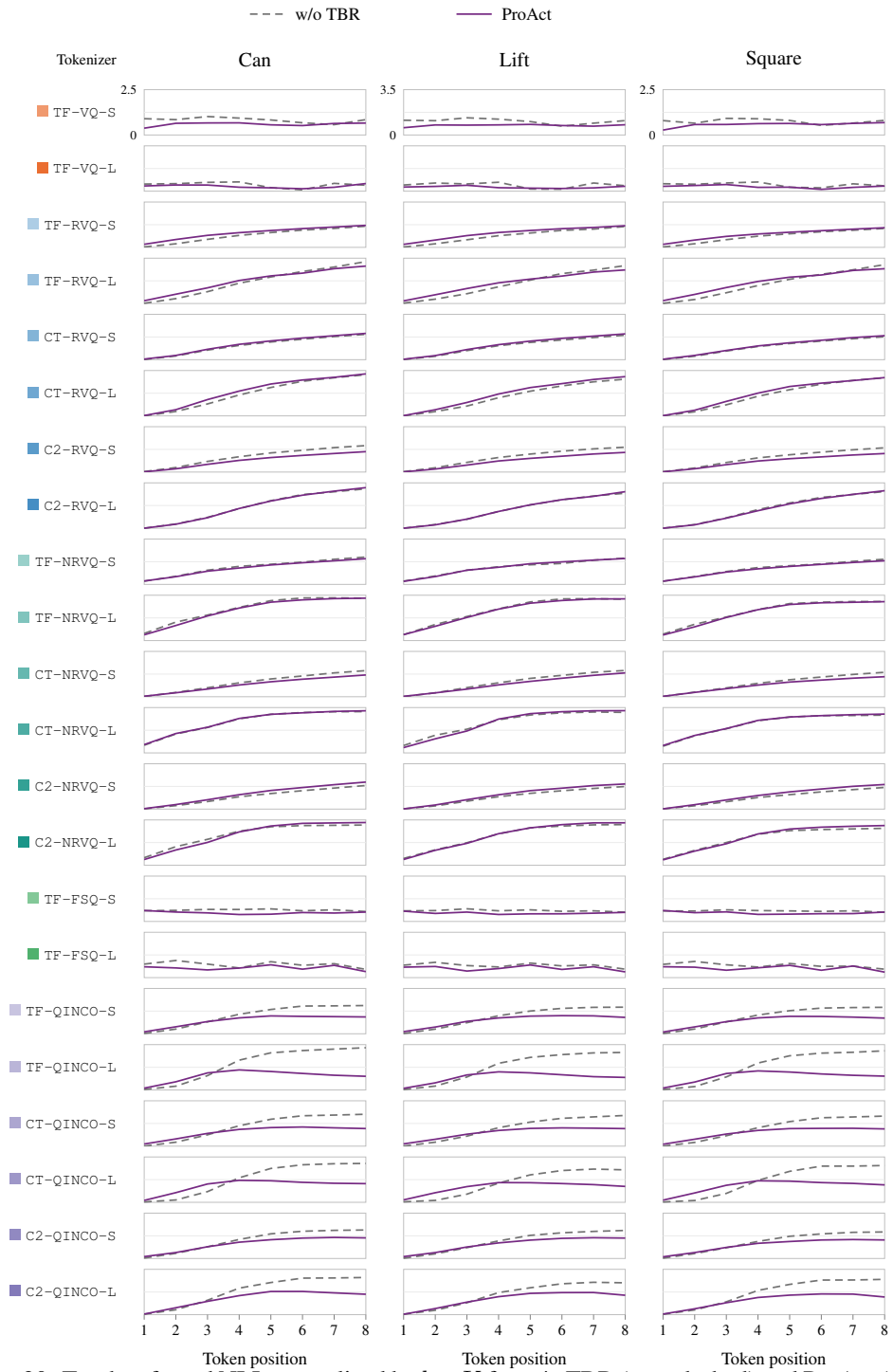


Figure 20: Teacher-forced NLL normalized by $\log K$ for w/o TBR (gray dashed) and ProAct (purple solid). Each row is one tokenizer; columns are tasks. Within each task, all rows share the first row's labeled y-range.

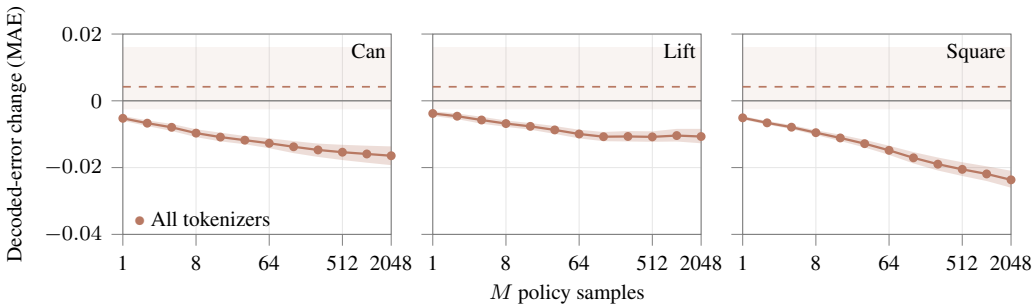


Figure 21: Decoded-action robustness on full validation states: w/o TBR minus R.O.. Solid curves (`CT-NRVQ-S/L` excluded) show paired Max-M mismatch-error changes; darker bands are 95% confidence intervals. Dashed lines show tokenizer validation reconstruction-error changes.

E.4 Detailed Decoded-Error Analysis

We isolate Token Perturbation through R.O. → w/o TBR and w/o TP → ProAct on validation demonstration pairs (O, A). Greedy decoding is deterministic; $M = 1$ draws one stochastic sequence (Appendix C.5). For either control–treatment pair (a, b), let $\widehat{A}^{\pi,s}_{\text{greedy}}$ and $\widehat{A}^{\pi,s,(m)}$ denote method s's greedy and mth sampled decoded actions. Using the normalized raw-action L_1 error ℓ, define

$$\begin{aligned} \ell_s^{(M)}(A) &= \max_{1\le m\le M} \ell(A, \widehat{A}^{\pi,s,(m)}), \\ \Delta\mathcal{E}^{b-a}_{\pi,\text{greedy}} &= \mathbb{E}\left[\ell(A, \widehat{A}^{\pi,b}_{\text{greedy}}) - \ell(A, \widehat{A}^{\pi,a}_{\text{greedy}})\right], \\ \Delta\mathcal{E}^{b-a}_{\pi,\max-M} &= \mathbb{E}\left[\ell_b^{(M)}(A) - \ell_a^{(M)}(A)\right]. \end{aligned} \tag{64}$$

Here M counts sequences sampled per policy, and negative changes favor TP. For Max-M, we retain only mismatches with each method's encoded target and require both methods to produce a mismatch on the same observation:

$$\Delta\mathcal{E}^{b-a}_{\pi,\max-M} = \mathbb{E}\left[\max_{\substack{m\le M:\\ \widehat{C}^{b,(m)}\neq\mathcal{T}_b(A)}} \ell(A, \widehat{A}^{\pi,b,(m)}) - \max_{\substack{m\le M:\\ \widehat{C}^{a,(m)}\neq\mathcal{T}_a(A)}} \ell(A, \widehat{A}^{\pi,a,(m)}) \,\middle|\, \begin{matrix}\exists m\le M{:}\widehat{C}^{a,(m)}\neq\mathcal{T}_a(A)\\ \exists m\le M{:}\widehat{C}^{b,(m)}\neq\mathcal{T}_b(A)\end{matrix}\right]. \tag{65}$$

Max-M probes the conditional error tail: for n independent mismatch errors with conditional CDF $F_{\neq}$,

$$\Pr\left[\max_{j\le n} E_j \le x \mid n\right] = F_{\neq}(x)^n. \tag{66}$$

Larger M is more likely to expose severe mismatches. For R.O. → w/o TBR, mean Greedy and Max-2048 changes are -0.0046 and -0.0156, improving 53 and 48 of 60 tokenizer–task conditions, respectively. For w/o TP → ProAct, they are -0.0086 and -0.0313, improving 62 and 61 of 66 conditions. The larger average tail reductions are consistent with decoder robustness, but family effects vary: RVQ slightly worsens for R.O. → w/o TBR under Max-2048, and VQ worsens for w/o TP → ProAct under both estimators.

Figures 22–23 give all tokenizer–task results. Absolute panels show MAE with supplied intervals and contextual reconstruction-error ticks; change panels show treatment minus control, with paired intervals only for Max-2048 (Greedy and $M = 1$ are point estimates). Daggered CT-NRVQ-S/L rows are excluded only from the R.O. w/o TBR aggregate; w/o TP/ProAct includes all 22 configurations.

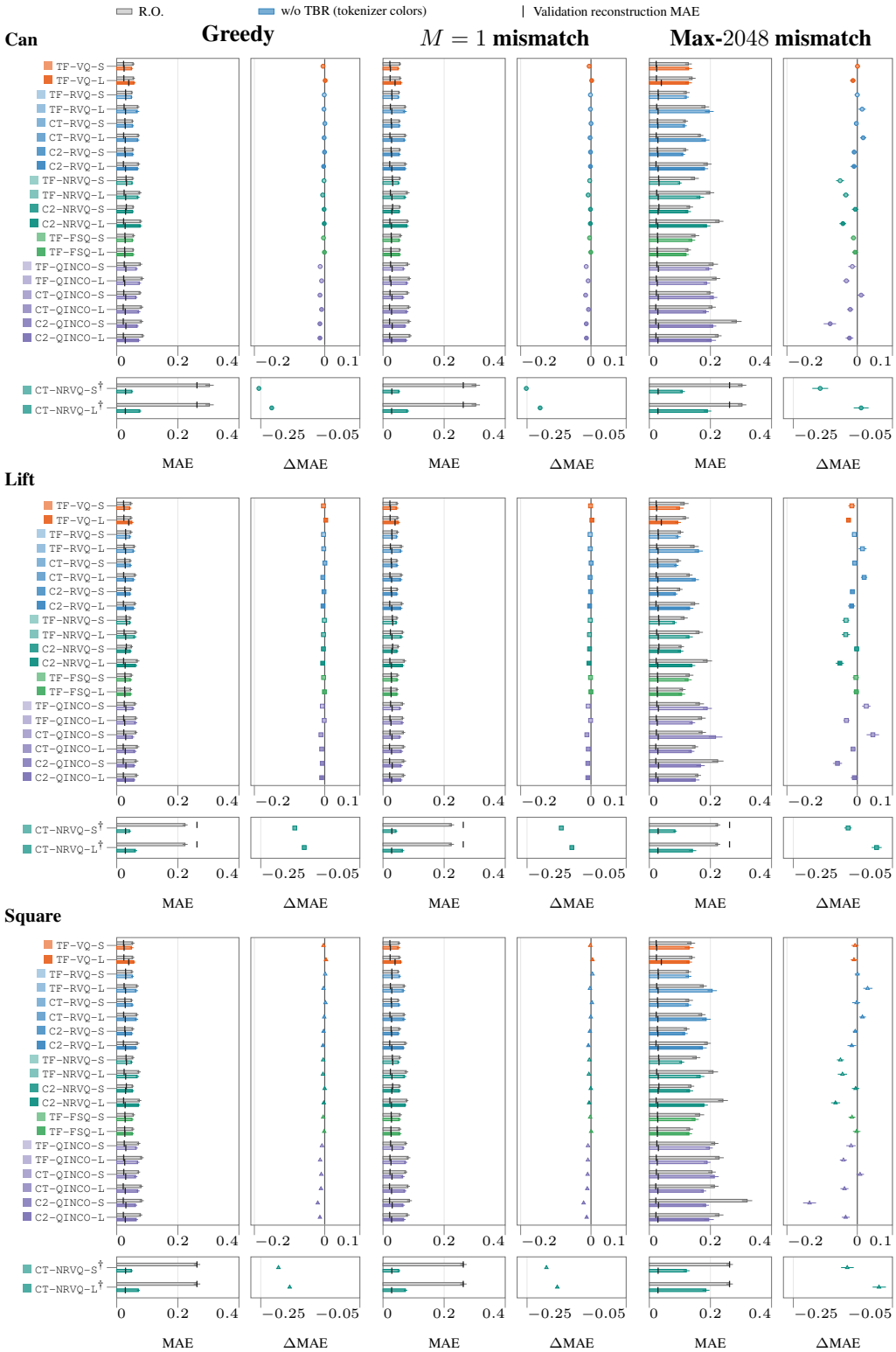


Figure 22: R.O. → w/o TBR decoded-error details for Can, Lift, and Square (top to bottom). Each metric pairs absolute decoded MAE with its treatment-minus-control change (ΔMAE). Gray and colored bars show R.O. and w/o TBR errors with supplied method intervals; black ticks mark validation reconstruction MAE. Only Max-2048 changes have paired intervals. Separate daggered CT-NRVQ-S/L rows are excluded from the aggregate and use a distinct change-axis scale.

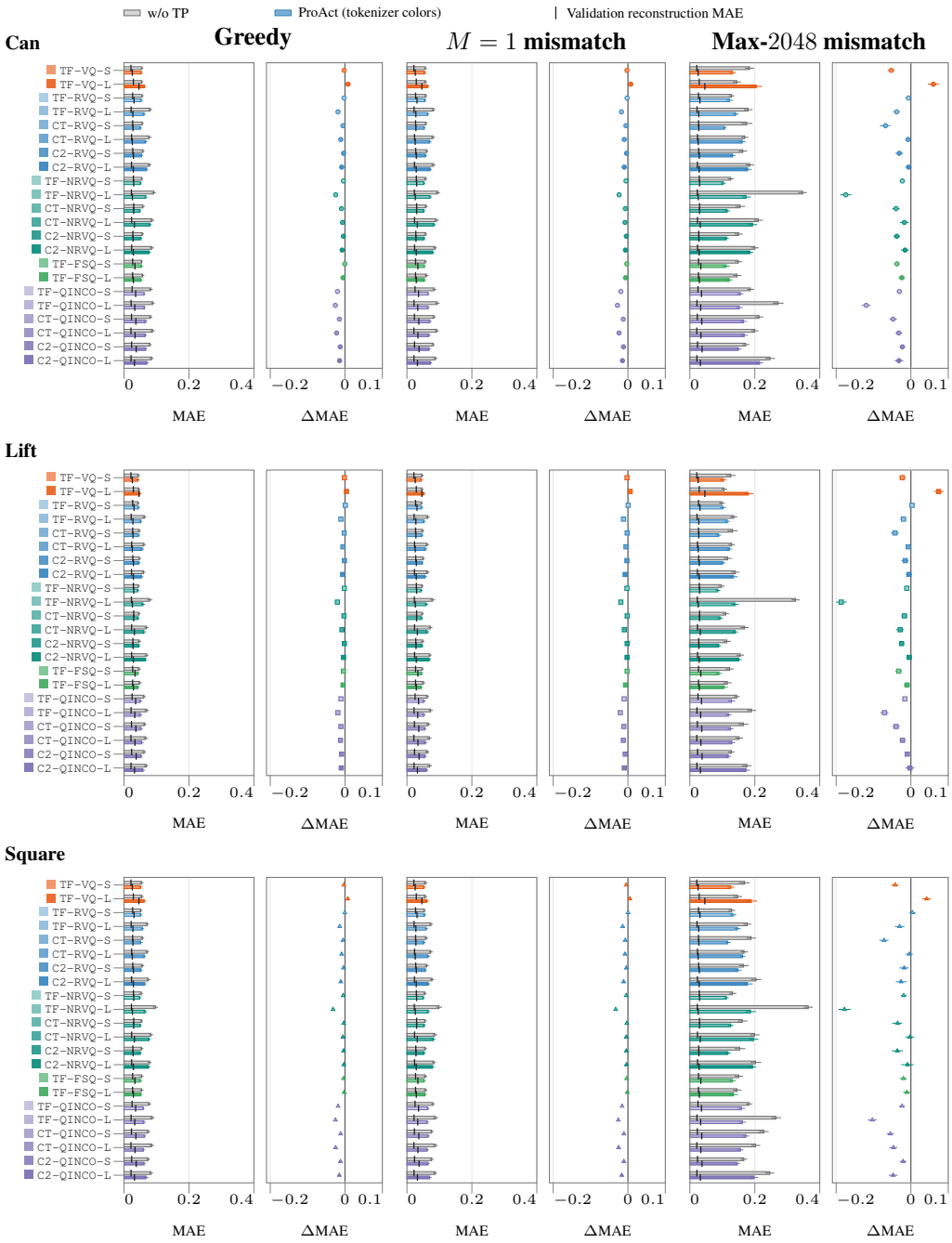


Figure 23: w/o TP → ProAct decoded-error details for Can, Lift, and Square (top to bottom), retaining all 22 tokenizers. Each metric pairs absolute decoded MAE with its treatment-minus-control change (ΔMAE). Gray and colored bars show w/o TP and ProAct errors with supplied method intervals; black ticks mark validation reconstruction MAE. Only Max-2048 changes have paired intervals.

E.5 Detailed Real-Robot Results

Shared setup and training. The physical UR5 experiments in Section 6.4 use a multi-task dataset of 780 demonstrations: 301 for Cube Stacking and 479 for Cup Hanging. For each tokenizer configuration and training objective, a single tokenizer and a single policy span both tasks; the same policy checkpoint is evaluated on each task.

Table 27 lists the ProAct tokenizer-training settings; implementation details follow Appendices D.2 and D.3.

Table 27: Real-robot ProAct hyperparameters. Arrows denote linear annealing with final scale 0.5; β remains fixed.

Component	Hyperparameter	Symbol	Value
TBR	Activation-preconditioning damping	λ	0.1
	Maximum full-covariance dimension	$d_{\max}$	512
TP	Sample-selection probability	β	0.1
	Token-mask probability	α	$1.0 \rightarrow 0.5$
	Neighborhood size	δ	$10 \rightarrow 5$

Evaluation protocol. Action chunks have length $H = 8$ and use the delta end effector pose representation, with nominal code capacity $K = 128$, $L = 8$. The policy executes four actions before replanning ($H_{\text{exec}} = 4$). Because the real-robot dataset is small, its held-out validation split is too small for teacher-forced validation accuracy to serve as a reliable checkpoint-selection signal; we therefore evaluate the final policy checkpoint for every condition, with 30 trials per task. Object initial poses are randomized for every trial. Table 4 reports the full per-task comparison in the main text. Across all five configurations, Cube Stacking successes increase from 28/150 (18.7%) to 85/150 (56.7%), and Cup Hanging successes increase from 28/150 (18.7%) to 81/150 (54.0%). Together, these give 56/300 (18.7%) for Reconstruction-only and 166/300 (55.3%) for ProAct.