WHEN DO DISCRETE ACTIONS MAKE GOOD CONTINUOUS CONTROLLERS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Though modern AI models like LLMs predict discrete tokens, today's robotic foundation models represent actions as continuous vectors. In this paper, we challenge this common consensus by asking if continuous action parametrizations are superior to their discrete counterparts. We conduct an extensive comparison of single-step regression and multi-step flow variants of both continuous and discrete policies in both from-scratch single-task and multi-task VLA training, in both simulation and on real hardware. In all regimes, we find that discrete action prediction, even the most naive, uniform-bin quantization, can match or even outperform continuous-action policies in task success rate. Moreover, we find that the challenge to discrete action policies is *not* action precision, but rather factorization error introduced by product-distribution predictions across time and action dimensions. We show that, when paired with the correct interventions—iterative computation through discrete flows, or distribution sharpening through classifier-free guidance—discrete flow policies can turn this weakness into a strength, enabling more responsive action changes during a predicted action chunk. Lastly, we show that attempts to explicitly inject geometric information into discrete policies, e.g. through corruption processes that emulate continuous flow matching, do *not* yield consistent improvements. Taken together, our findings question existing assumptions that robotic action prediction requires distinct design decisions compared to other AI domains, and motivates the possibility of discrete action prediction as a compelling alternative to today's continuous action parametrizations. Link for the webpage https://anonymous-wolf-submissions.github.io/iclr-submission

1 INTRODUCTION

A major goal of modern robot learning is to perform complex and varied tasks to high precision. As a result, the community has devoted significant study to the question of how to parametrize the actions predicted by robotic policies and foundation models, such as vision-language-action models (VLAs) (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; NVIDIA et al., 2025) and world action models (WAMs) (Li et al., 2025; Ye et al., 2026; Kim et al., 2026; Li et al., 2026). The common consensus is that robot models should predict continuous-valued actions during deployment, as these are lossless and therefore support high-precision behavior. For example, most of today's VLAs predict actions using continuous-space generative models, such as flow (Black et al., 2024; Physical Intelligence et al., 2025) or diffusion models (Chi et al., 2023), which are believed to better capture training data diversity whilst enhancing the precision of action predictions.

At the same time, modern robot learning is adopting increasingly more features from other AI domains, notably language modeling. These include test-time reasoning (Wei et al., 2022), on-task self-correction (Huang et al., 2022), future goal prediction (Du et al., 2023), and long-term memory (Wang et al., 2023). In fact, recent work has showcased the possibility that frontier language models can directly control robot embodiments through text generation (Brohan et al., 2023; Kim et al., 2024). Yet, today's frontier AI models, even ones that process multiple modalities such as text, images, and audio, both take as input and produce discretely quantized tokens. Whether robot actions should follow the same discrete format therefore raises a key question:

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.

Are robot actions unique among other AI modalities in that they must be predicted as continuous values, or can **discrete action representations** compete with, or even outperform, their continuous counterparts?

1.1 CONTRIBUTIONS

In this work, we undertake a systematic comparison between the performance of continuous and discrete action parameterizations for both single-task robot policies and multi-task vision-language-action (VLA) models. In short, we show that **discrete action parametrization can be competitive with, and even uniformly outperform, continuous action prediction**, provided that correct design decisions are made. These findings provide a deeper understanding of the performance of recent discrete diffusion/flow VLA policies (Liang et al., 2025; Wen et al., 2025; Chen et al., 2025).

In greater detail, we consider parametrizations of the *action head*, or part of a robot model responsible for action prediction. Our comparison builds on classical supervised control and imitation learning (Pomerleau, 1988; Ross et al., 2011; Osa et al., 2018). We focus our study on 4 representative parametrizations: (1) continuous regression (CR), which predicts continuous-valued actions in a single forward pass and is trained with a regression objective (Bojarski et al., 2016; Kim et al., 2025) (2) continuous flow (CF), which generates continuous actions using flow matching, a family applied to robot control and motion planning (Hu et al., 2024; Chisari et al., 2024; Black et al., 2024; Zhang et al., 2024; Nguyen et al., 2025) (3) discrete regression (DR), which predicts a quantized action via cross-entropy classification (Brohan et al., 2022) and (4) discrete flow (DF), which predicts quantized actions in multiple steps via the discrete flow matching objective (Gat et al., 2024). Hybrid methods such as Behavior Transformer (Shafiullah et al., 2022) combine discrete action-bin prediction with a continuous offset; in contrast, we study purely discrete and continuous action heads separately to isolate the effect of action representation. Recent work compares regression and flow policies under architecture-matched settings (Pan et al., 2025). We evaluate these parametrizations from single-task policies on RoboMimic (Mandlekar et al., 2021) to multi-task VLA-scale policies using RoboCasa (Nasiriany et al., 2026), DROID (Khazatsky et al., 2024), and LIBERO (Liu et al., 2023), as well as real-world robot experiments. All are evaluated using closed-loop task success, action prediction accuracy, and within-chunk temporal variation. We identify 3 key findings that upend prior beliefs about the relative performance of discrete and continuous action modalities, challenging the prevailing view that precise robot control requires predicting continuous action prediction:

- **Finding 1: Control Precision** Contrary to prior belief, discrete action parametrizations, even discrete regression (DR) with the simplest bin discretization, can exhibit greater control precision than their continuous counterparts, despite the possibly lossy quantization of actions! We support this claim in both sim and real settings in Section 3. We show that this arises from greater policy reactivity, both to the action-chunk time index and the conditioning observation.
- **Finding 2: Iterative Refinement and Guidance Mitigate Factorization Error** Independently predicted tokens can combine incompatible choices across time and action dimensions, even when individual token distributions are accurate. DF solves this by iterative computation, while classifier-free guidance reduces ambiguity by changing token preferences. Both improve control significantly, supporting joint consistency as main explanation for their gains. In the RoboCasa comparison, guidance raises DR success from 25.8% to 50.2%, approaching DF's 53.8% and CF's 52.3%, largely ahead of CR's 38.8%.
- **Finding 3: Explicit Geometry and Structured Tokenization Provide Limited Additional Gains** (Section 5). Encoding action distances or using alternative tokenizers improves some configurations, but none exceeds our best DF result with uniform bins and the standard mixture path. Adjacent bins also acquire similar output embeddings, suggesting that discrete heads can learn local geometry without encoding it explicitly.

2 PRELIMINARIES: ACTION REPRESENTATIONS AND PREDICTION HEADS

Notation. Let o denote the policy's conditioning information, such as images, language, and proprioception, and let $\mathbf{a}_{1:H} = (\mathbf{a}_1, \ldots, \mathbf{a}_H) \in \mathbb{R}^{H \times M}$ be an action chunk (Zhao et al., 2023) of horizon H, with M action coordinates per timestep. We represent actions in *end-effector (EEF) space*, with

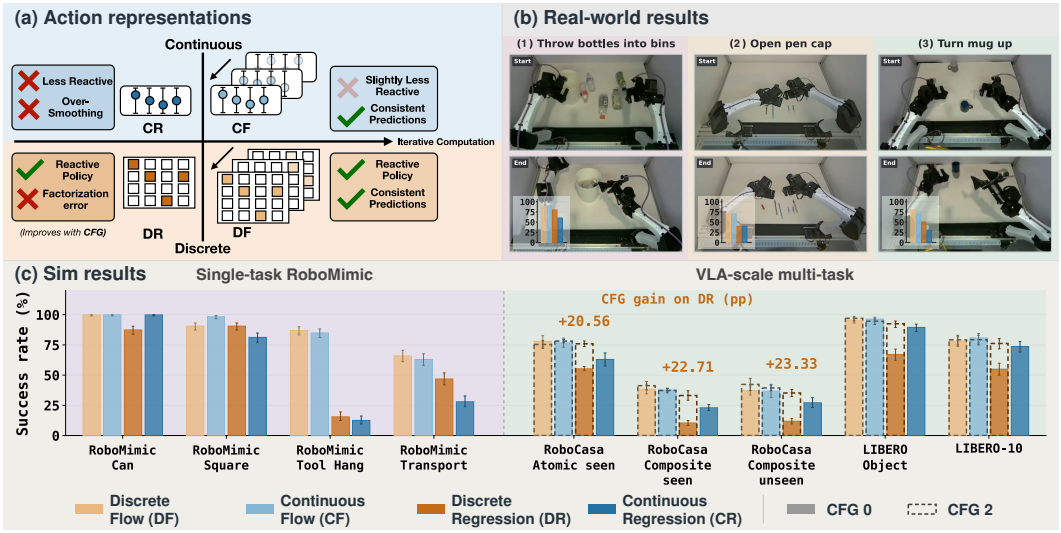


Figure 1: **(a)** Action Representation and comparison; **(b)** Closed-loop success on RoboMimic, and VLA-sim (RoboCasa and LIBERO); **(c)** VLA-real (three real-world tasks); Within each group, heads appear in the order DF, CF, DR, CR; orange denotes discrete heads and blue denotes continuous heads. For RoboCasa and LIBERO DF, CF, and DR, filled bars show CFG 0 and transparent dashed bars show CFG 2. Details are in Appendix C.

3 translation coordinates $\mathbf{a}^{\mathrm{tr}} \in \mathbb{R}^3$, a 6D rotation representation $\mathbf{a}^{\mathrm{rot}} \in \mathbb{R}^6$ parameterizing a rotation matrix in $\mathrm{SO}(3)$, and a binary gripper coordinate $\mathbf{a}^{\mathrm{g}} \in \{0, 1\}$, and $\mathbf{a}_i = [\mathbf{a}_i^{\mathrm{tr}}, \mathbf{a}_i^{\mathrm{rot}}, \mathbf{a}_i^{\mathrm{g}}]$. From training pairs $(o, \mathbf{a}_{1:H}) \sim \mathcal{D}$, we learn (possibly stochastic) policies $\hat{\mathbf{a}}_{1:H} \sim \pi_\theta(o)$ that execute the chunk open loop and replan. All heads share a backbone design; full objectives and implementation details are in Appendix B.

Continuous regression (CR). The policy $\pi_\theta^{\mathrm{CR}}(o)$ predicts a chunk in one pass using squared-error regression. Its population minimizer is the conditional mean $\mathbb{E}[\mathbf{a}_{1:H} \mid o]$, which need not be a demonstrated action when several behaviors are plausible.

Continuous flow (CF). Flow matching learns a conditional velocity field v_θ (Lipman et al., 2022; Liu et al., 2022). From $\mathbf{x}_0 = \boldsymbol{\epsilon}_{\mathrm{C}} \sim \mathcal{N}(0, \mathbf{I})$, generation follows $\dot{\mathbf{x}}_t = v_\theta(\mathbf{x}_t, t, o)$ over $t \in [0, 1]$. After N Euler integration steps, the policy returns $\hat{\mathbf{a}}_{1:H} \leftarrow \mathbf{x}_{t_N}$, with $t_N = 1$. Here, flow time t indexes generation, rather than the action horizon.

Discrete action representation. We quantize each action coordinate into one of B bins after normalization and clipping. Let $[B] = \{1, \ldots, B\}$ and $L = HM$. Flattening the quantized chunk in a fixed order gives $\mathbf{z} = (z_1, \ldots, z_L) \in [B]^L$, whose entries we call *tokens*. Thus, B is both the number of bins per coordinate and the action-token vocabulary size; unless otherwise stated, $B = 512$. The quantizer $Q : \mathbb{R}^{H \times M} \to [B]^L$ and decoder $D_Q : [B]^L \to \mathbb{R}^{H \times M}$ satisfy $\mathbf{z} = Q(\mathbf{a}_{1:H})$ and $\hat{\mathbf{a}}_{1:H} = D_Q(\hat{\mathbf{z}})$. We primarily use **uniform quantization**. For action coordinate i, let $[\ell_i, u_i]$ be its range and $\Delta_i = (u_i - \ell_i)/B$ its bin width. With scalar action $a_{h,i}$ and token position $j = (h-1)M + i$, encoding and decoding are $Q(\mathbf{a}_{1:H})_j := \mathrm{clip}_{[1,B]}\left(1 + \left\lfloor \frac{a_{h,i} - \ell_i}{\Delta_i} \right\rfloor\right)$, $[D_Q(\mathbf{z})]_{h,i} := \ell_i + (z_j - \frac{1}{2})\Delta_i$. Clipping keeps the bin index in $[B]$, including at the upper endpoint; decoding returns the bin center and restores the chunk shape. Both maps use normalized coordinates, with normalization inverted before execution. Recipe details and alternative tokenizations are described in Appendix B.2.

Discrete regression (DR). In one forward pass, the head outputs $L \times B$ logits whose position-wise softmax defines $p_{\theta,j}^{\mathrm{DR}}(\cdot \mid o)$ (Brohan et al., 2022). Conditioned on the model inputs, it factorizes the token distribution as $p_\theta^{\mathrm{DR}}(\mathbf{z} \mid o) = \prod_{j=1}^{L} p_{\theta,j}^{\mathrm{DR}}(z_j \mid o)$. For $\mathbf{z} = Q(\mathbf{a}_{1:H})$, we train with tokenwise cross-entropy: $\mathcal{L}_{\mathrm{DR}} = \mathbb{E}_{(o, \mathbf{a}_{1:H}) \sim \mathcal{D}}\left[-\frac{1}{L}\sum_{j=1}^{L} \log p_{\theta,j}^{\mathrm{DR}}(z_j \mid o)\right]$. The factorization makes maximizing the model's joint probability tractable: we greedily select $\hat{z}_j = \arg\max_{k \in [B]} p_{\theta,j}^{\mathrm{DR}}(k \mid o)$ at each position (Devlin et al., 2019). Writing $\hat{\mathbf{z}} = (\hat{z}_1, \ldots, \hat{z}_L)$, the policy returns $\hat{\mathbf{a}}_{1:H} := D_Q(\hat{\mathbf{z}})$.

Discrete flow (DF). Discrete flow (Gat et al., 2024) generates a token chunk through iterative updates, starting from a product source $q_0(\mathbf{z}_0) = \prod_{j=1}^{L} q_{0,j}(z_{0,j})$. Let $\alpha : [0,1] \to [0,1]$ be a continuously differentiable, nondecreasing schedule with $\alpha(0) = 0$, $\alpha(1) = 1$, and $\alpha(t) < 1$ for $t < 1$. For a clean chunk $\mathbf{z} = Q(\mathbf{a}_{1:H})$, the conditional probability path is

$$q_t(\mathbf{z}_t \mid \mathbf{z}) = \prod_{j=1}^{L} \left[\alpha(t)\mathbf{1}\{z_{t,j} = z_j\} + (1 - \alpha(t))q_{0,j}(z_{t,j})\right]. \tag{1}$$

At each position, this path interpolates between the source and the clean target token. The source $q_0(\cdot)$ may be uniform over $[B]$ or concentrated on an auxiliary mask token (Sahoo et al., 2024), which does not decode to an action bin. As in DR, the head predicts a clean-token distribution at each position, but additionally conditions on the current chunk and flow time: $p^{\mathrm{DF}}_{\theta,j}(\cdot \mid \mathbf{z}_t, t, o)$. For each training pair, we sample $t \sim \mathcal{U}[0,1]$ and $\mathbf{z}_t \sim q_t(\cdot \mid \mathbf{z})$, and minimize clean-token cross-entropy (Gat et al., 2024):

$$\mathcal{L}_{\mathrm{DF}} = \mathbb{E}_{(o,\mathbf{a}_{1:H})\sim\mathcal{D},\, t,\, \mathbf{z}_t}\left[-\tfrac{1}{L}\textstyle\sum_{j=1}^{L} \log p^{\mathrm{DF}}_{\theta,j}(z_j \mid \mathbf{z}_t, t, o)\right]. \tag{2}$$

At inference, initialize $\mathbf{z}_{t_0} \sim q_0$ and use $0 = t_0 < \cdots < t_N = 1$. At step n, draw proposals $\widetilde{z}_{n,j} \sim p^{\mathrm{DF}}_{\theta,j}(\cdot \mid \mathbf{z}_{t_n}, t_n, o)$ and update each position, writing $\alpha_n = \alpha(t_n)$, as

$$z_{t_{n+1},j} = \begin{cases} \widetilde{z}_{n,j}, & \text{with probability } \eta_n, \\ z_{t_n,j}, & \text{otherwise,} \end{cases} \qquad \eta_n = \frac{\alpha_{n+1} - \alpha_n}{1 - \alpha_n}. \tag{3}$$

Finally, the policy returns $\hat{\mathbf{a}}_{1:H} \leftarrow (o; \boldsymbol{\epsilon}_{\mathrm{D}}) := D_Q(\mathbf{z}_{t_N})$. Importantly, while each single DF step predicts independently per position (as in DR), DF introduces position-wise correlation through multi-stage computation of the final output tokens.

Classifier-free guidance. CFG (Ho & Salimans, 2022) combines conditional and unconditional predictions. For discrete heads, suppressing other inputs, $p^{\mathrm{CFG}}_{\theta,j}(k \mid c) \propto p_{\theta,j}(k \mid c)^{1+\lambda} p_{\theta,j}(k \mid \emptyset)^{-\lambda}$, normalized over bins. Here c is the guided conditioning, $\emptyset$ is its null counterpart, and $\lambda \geq 0$ controls guidance strength (0 means no guidance). DR greedily decodes these distributions; DF uses them to sample clean-token proposals. CF combines conditional and unconditional velocity predictions before each integration step; we omit CFG for CR. Details are in Appendix B.5.

3 Discrete Action Prediction Can Match Continuous Parametrizations

In this section, we demonstrate that discrete action prediction matches, and often outperforms continuous action prediction *across* both flow and regression policies. To evaluate performance, and refute possible precision loss under action discretization, we measure (1) task success rate under closed-loop deployment, and (2) mean ℓ_2 action prediction error on held-out validation data.

Evaluation Protocol. To demonstrates that our findings are broadly applicable, we consider a wide breadth of tasks; we defer further implementation details to Appendix A.

- **Single-Task Imitation Learning.** To test whether discrete policies' advantages depend on pretrained VLM, we train separate policies from scratch, without a pretrained VLM backbone, on the Can, Square, Tool Hang and Transport tasks in the RoboMimic suite (Mandlekar et al., 2021). In particular Tool Hang and Transport are representative of tasks that demand high control precision.
- **VLA Scale Tasks in sim.** We post-train the PaliGemma Beyer et al. (2024) VLM, used by the π-series models Black et al. (2024); Physical Intelligence et al. (2025), with new action heads corresponding to each action head parametrization. As representatives of VLA training, we pretrain on a mixture of 3 datasets: RoboCasa Nasiriany et al. (2024; 2026), DROID Khazatsky et al. (2024), and MimicGen Mandlekar et al. (2023), and evaluate on each. Evaluations on RoboCasa are grouped into 3 categories: atomic-seen and composite-seen consisting of atomic and composite tasks, respectively, that are contained in the training data; and composite-unseen, consisting of tasks that require compositions of in-distribution atomic-seen, but in novel combinations.
- **Real World VLA tasks**. We used the same backbone and action head setting in sim but trained on different real datasets and SFT on three tasks: throw bottles into bins, open pen caps, and turn mug up. The rollout results are provided in Fig. 1(b)

Results. We depict the closed-loop performance of all four policy parametrizations : CR, CF, DR, DF in Fig. 1(b,c). We see two surprising findings: (i) across ***VLA*** and ***single-task, from-scratch*** training, discrete flow is competitive with, and often slightly surpasses, continuous flow. More surprisingly, (ii) discrete regression *consistently outperforms* continuous regression. The fact that discrete *regression* outperforms its continuous counterpart, even on single-task from scratch training, debunks two widely held hypotheses: (a) that precision is a fundamental bottleneck to discrete policies (Shafiullah et al., 2022); (b) that discrete policies are only beneficial in VLA settings, where it has been hypothesized, that discrete predictions are more compatible with token-based VLM backbones (Brohan et al., 2023; Driess et al., 2026). As further evidence that discrete policies are not fundamentally limited by precision, we present their end-effector reconstruction error in Fig. 2, showing that discrete policies remain competitive (and in some cases slightly outperforms) their continuous counterparts. In fact, in Fig. 2, we show that discrete flow policies in particular enjoy *lower* end-effector error early on in the action chunk generation and along the finer motion regime, suggesting the potential for greater precision when actions are discrete.

3.1 What Enables the Performance of Discrete Policies?

Paradoxically, our previous findings suggest that the opposite of the common wisdom regarding discrete policies might be true: perhaps, discrete policies are *uniformly better* at high-precision action prediction than their continuous counterparts?

However, this belief is not quite correct. In Appendix E.2, we show find that the ℓ_2 reconstruction errors across all parametrizations are roughly similar, and show *no consistent correlation with task success*. For example, DR exhibits less error than CR on some benchmarks, and more on others, despite *always* enjoying greater closed loop task success.

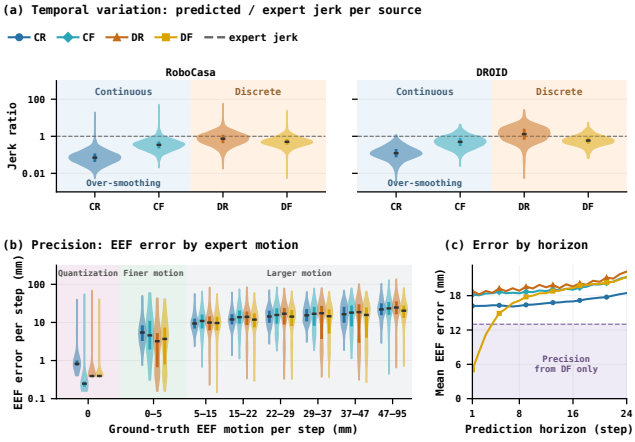


Figure 2: **Temporal variation and precision on held-out chunks. (a)** Jerk-ratio violin plots; Continuous underestimates expert temporal variation. **(b)** EEF error, grouped by ground-truth motion."Quantization" highlights the discrete heads' static-error floor. **(c)** Mean RoboCasa EEF error by horizon; DF has the lowest error in the first steps of the chunk. More details are in Appendix D.1.

Temporal Variation, via the Jerk Ratio, Reflects Discrete Policy Competitiveness. Instead, we propose an alternative explanation for the success of discrete policies: that they enable **greater temporal variation** across action chunks than do their continuous counterparts. To make this rigorous, Fig. 2 (a) depicts each chunk's **jerk ratio**. [1] For a given action chunk $\mathbf{a}_{1:H}$, let $\mathbf{a}_h^{\text{tr}}$ denote the translation coordinates of the end-effector. We define the difference approximation to the jerk at step h as $\Delta^3\mathbf{a}_h^{\text{tr}} = \mathbf{a}_{h+3}^{\text{tr}} - 3\mathbf{a}_{h+2}^{\text{tr}} + 3\mathbf{a}_{h+1}^{\text{tr}} - \mathbf{a}_h^{\text{tr}}, \quad h = 1, \ldots, H-3$. The chunk's mean jerk magnitude and predicted-to-target jerk ratio are

$$\text{Jerk}(\mathbf{a}_{1:H}) = \frac{1}{H-3}\sum_{h=1}^{H-3}\left\|\Delta^3\mathbf{a}_h^{\text{tr}}\right\|_2, \quad R_{\text{jerk}} = \frac{\text{Jerk}(\hat{\mathbf{a}}_{1:H})}{\text{Jerk}(\mathbf{a}_{1:H})}. \tag{4}$$

Thus, jerk ratio of $R_{\text{jerk}} = 1$ means that the learned policy's jerk matches the ground-truth jerk on a training example; $R_{\text{jerk}} < 1$ means jerk is underestimated, so that the policy has *oversmoothed* the training data with respect to time.

Across all datasets, Fig. 2 (a) reveals that continuous regression enjoys a jerk ratio substantially less than 1, whereas discrete regression is close to (and occasionally above) 1. This suggests that discrete

[1] Recall that *jerk* refers to a the third-time derivative of its actions, and thus reflects the *variability* of actions during a given action chunk.

action modalities can more easily represent large temporal variation in a single pass, likely due to the probabilistic *decoupling* of predictions across time steps. These differences reduce under iterative computation, as Fig. 2 (a) shows CF and DF have much closer jerk ratios than do the single-pass counterparts. To identify differences that persist between DF and CF, Fig. 2 (b,c) reveal that discrete policies better fit other subtle variations: in (b), we see that discrete modalities offer more response to smaller degrees of EEF movement, especially the 0–5 mm range (note the logarithmic plot scale); in (c) we see that DF in particular fits actions earlier in the chunk more accurately, which require smaller relative motion. We hypothesize that continuous objectives underweight/over-smooth small action differences compared to categorical prediction. In Appendix G.1, we corroborate these hypotheses in controlled studies of fitting synthetic functions with varying levels of smoothness.

DR/DF Exhibit Greater Responsiveness to Conditioning Changes than CR/CF.

Beyond temporal variation within a chunk, we examine responsiveness to observation changes, formally $\frac{\partial \pi}{\partial o}$. Fig. 3 shows that both continuous heads produce smaller changes in their mean predicted chunks than their discrete counterparts, underestimating expert change magnitudes on held-out data. DR tracks the expert response most closely, while DF remains closer than CF at every tested state displacement. On the other hand, additional diagnostics in Appendix D.3 also show that CF stays closer to nearby training actions than DF, corroborating findings from Pan et al. (2025). These results suggest that stronger adherence to the training data may come at the cost of responsiveness. DF may offer a better balance, retaining the relative stability under disturbances observed for flow heads while responding more closely to expert changes.

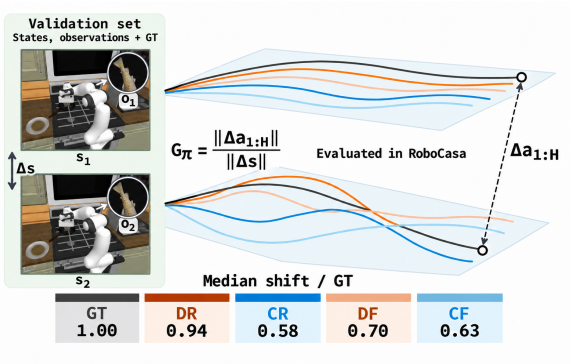


Figure 3: Median state displacements and chunk changes (normalized units). Model columns show raw proposal-mean shifts; sampling floors are not subtracted.

Takeaway 1: Discrete policies generally **match or exceed** the control performance of their continuous counterparts across the evaluated testbeds. They obtain stronger performance in both VLA and from-scratch settings. These gains arise from **better capturing variation** across within-chunk time steps and conditioning observations, rather than outright reconstruction error.

4 Factorization Error and Iterative Computation in Discrete Policies

Thus far, Section 3 establishes that discrete regression (DR) outperforms continuous regression (CR), in part due to better capturing temporal variation, subtle action changes, and response to the input variables. In this section, we show that the greater temporal reactivity observed in DR comes at the cost of action inconsistency, or what we call **factorization error** (Finding A), harming policy performance. We then identify that the key interventions which enhance performance—applying classifier-free-guidance (Finding B) and introducing iterative computation by moving to discrete flow (Finding C) —correct this inconsistency via complementary mechanisms described below.

Factorization Error in Discrete Policies. DR predicts a categorical distribution for each action token, giving the factorized distribution $p_\theta(\mathbf{z} \mid o) = \prod_j p_{\theta,j}(z_j \mid o)$. When an observation admits several plausible behaviors, different token distributions can assign probability to alternatives behaviors independently. These independent choices can combine incompatible alternatives across time steps and action dimensions. We refer to this failure to preserve dependencies between token choices as **factorization error**. This may occur even when every token marginal is learned correctly, because the product of the marginals need not recover the joint distribution of demonstrated chunks.

Evaluation Protocol for Factorization Error. To isolate token ambiguity as a potential source of incompatible action tokens, we vary (i) the number of function evaluations (NFE) in DF, comparing

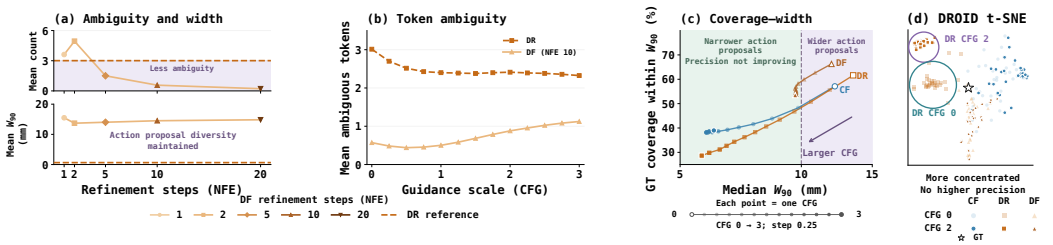


Figure 4: **Iterative refinement reduces token ambiguity. (a,b)** At CFG 0, DF at NFE $1, 2, 5, 10, 20$ show separated-mode prevalence (a) ambiguity decreasing, action diversity remains $\exp(\mathcal{H}(p))$ (b); the orange dashed curve is DR. **(c)** Median central-90% width W_{90} of action proposal and its ground truth coverage, run from CFG 0 to 3. **(d)** DROID t-SNE; the star marks the GT. ; additional diagnostics and protocols are in Appendix E.1.

to DR as a baseline, and (ii) the classifier-free guidance (CFG, Section 2) scale. Note that greater NFE provides more opportunity for self-correction, whereas greater CFG "sharpens" the generated action distribution. Due to space constraints, policy performance under varying CFG/NFE is reported in Appendix E.3; in short, performance improves with NFE and appropriate CFG scale.

Here, we focus on two metrics, deferring precise details to Appendix E.1: **1.** *Mode-mixture*: we identify distributions with at least two "incompatible" modes in the same action chunk. We report the median count of such modes within one chunk across NFE Fig. 4(a, top) and across CFG scales Fig. 4(b). **2.** *Translation Quantiles*. Next, we measure the *diversity* of actions generated across repeated policy samples from the same observation. This lets us test (and ultimately rule out) an alternative hypothesis that DF improves performance over DR by better capturing multimodality. Specifically, we measure the central-90% interval width $W_{90} = Q_{0.95} - Q_{0.05}$ for each of the three translation coordinates per time step, and take the Euclidean norm of the 3 per-axis widths. We plot the norm of median widths against number of function evaluations (NFE) in Fig. 4(a, bottom) and CFG scale Fig. 4(c). Lastly, Fig. 4(d) qualitatively visualizes action diversity.

Finding A: Factorization Error Across Horizon and Coordinates Can Limit DR. In Fig. 4(a), we show that DR suffers from more mode-mixture than DF, even worse at zero CFG. As further evidence, Fig. 5 illustrates the mechanism in the simple Push-T environment. Here, DR assigns probability to separated movement alternatives, and their relative mass changes across the prediction horizon. Appendix E.4 presents the analogous ambiguity in RoboCasa. These examples confirm the quantitative findings in Fig. 4, illustrating how factorized decoding can introduce *spurious* temporal variation, even when individual token predictions capture plausible alternatives.

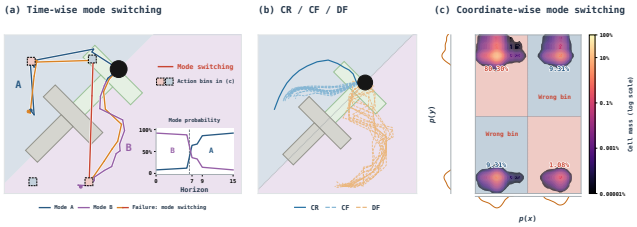


Figure 5: **(a)** Temporal mode switching within an action chunk as relative mode probabilities vary across the prediction horizon; **(b)** Action chunks predicted by CR, CF, and DF from the same starting position. **(c)** Coordinate-wise factorization within a timestep: independently combining coordinates from two valid modes can produce two additional, incompatible actions.

Finding B: Classifier-Free Guidance Reduces Ambiguity and Makes DR Competitive with DF. As further evidence of the presence of factorization error, we observe that increasing guidance from CFG 0 to CFG 2 raises pooled RoboCasa success from 27.07% to 49.20% (Appendix E.2). This coincides with drastic reduction in the mean-width of the confidence intervals (Fig. 4(c)), which is consistent with reductions in spurious temporal variation. As noted above, guidance is known to "sharpen" the resulting generated distribution (Ho & Salimans, 2022; Karras et al., 2024), and thus is likely to be resolving mode ambiguity. Moreover, Fig. 4(c,d) shows that CFG reduces coverage of plausible ground-truth EEF positions, suggesting that performance advantages from CFG primarily

reflect reduced factorization error rather than improved prediction accuracy. Across sources, CFG almost never improves EEF error by more than 1 mm (Appendix D.2).

Finding C: Iterative Computation Aids Discrete Policies via Reducing Factorization Error. While CFG guidance is a partial-yet-imperfect remedy of factorization error, Here we conclude we show that iterative computation, as in DF, reduces factorization error further. To support this claim, we find that DF reduces the number of inconsistent-mode predictions per chunk by a factor of 5.28 relative to DR, bringing the average below one at NFE $\geq$ 10 (Fig. 4(a,top)). This coincides with a 4.7% increase in RoboCasa success (Appendix E.3). These suggest that iterative computation makes discrete action predictions more cohesive than guidance alone. We note that, for continuous action predictions, the benefits of iterative computation are somewhat *opposite*, combating over-smoothing (Fig. 2) and policy under-reactivity (Fig. 5). The high level role of iterative computation, however, is the same, in that it performs self-correction on policy actions (Pan et al., 2025).

Does action diversity explain the performance of DF? (Likely not) Despite exhibiting low factorization error, DF exhibits *greater action diversity* than DR, as shown in all Fig. 4 plots. However, this is unlikely to explain its greater performance. CFG=2 notably reduces DF action diversity, and reduces action diversity in CF and DR considerably more; yet in all three, performance increases, most notably in DF (see Appendix E.3). Moreover, although performance improves up to NFE$=10$, action diversity remains relatively constant (Fig. 4(a,bottom)). These findings suggest that preservation of full action diversity in the policy is not essential for task success.

Can temporal autoregression resolve factorization error? Temporal autoregression provides another way to coordinate action choices across time. We implement a discrete autoregressive (AR) policy that predicts the first action from the observation, then predicts each subsequent action conditioned on the observation and all previously generated actions. Coordinates within each time step are predicted in parallel, requiring 24 sequential predictions for a 24-step chunk. Without CFG, temporal AR achieves 42.8% success on RoboCasa, exceeding DR's 25.8% but remaining below DF. However, with CFG, the AR slightly trails both DR and DF.

CFG $0 \rightarrow 2$	**AR**	DR (ref.)	DF (ref.)
EEF (mm) $\downarrow$	**18.4** $\rightarrow$ **24.4**	16.4 $\rightarrow$ 16.3	14.1 $\rightarrow$ 18.2
Success (%) $\uparrow$	**42.8** $\rightarrow$ **48.3**	27.0 $\rightarrow$ 49.2	52.3 $\rightarrow$ 53.9

Table 1: EEF error and RoboCasa success

Takeaway 2: We identify **factorization error**, a form of temporal inconsistency, as the cause of DR's relative lack of performance compared to CF and DF. Iterative computation in DF yields maximal performance, but with appropriate CFG, DR's performance is **surprising close**; this reveals that iterative computation and distribution sharpening are both effective means for combating factorization errors.

5 Additional Structure Provides Limited Gains for Discrete Policies

Section 4 links the gains from iterative refinement and guidance to reduced ambiguity in discrete action predictions. In this section, we ask ***can explicit action geometry or structured tokenization further improve on simple categorical prediction with uniform bins?*** We now consider two mechanism for adding structure: encoding metric distances between action values during the DF corruption process, or changing how action chunks are tokenized.

Explicit Geometry Provides Limited Control Gains. We introduce action geometry in two ways. For DR, distance-based label smoothing assigns target probability to nearby action bins as well as the demonstrated bin, with probability decreasing with distance. We combine this with logit soft-capping to limit confidence. For DF, we replace the uniform-source mixture path in Equation (1) with a metric-induced corruption path: $q_t^{\text{metric}}(\mathbf{z}_t \mid \mathbf{z}) = \prod_{j=1}^{L} \frac{\exp[-\beta(\alpha(t))\, d(z_{t,j}, z_j)]}{\sum_{b=1}^{B} \exp[-\beta(\alpha(t))\, d_p(b, z_j)]}$, where $d(k, b) = |k - b|/((B-1)/2)$, and β is an increasing concentration schedule. As generation progresses, the metric path concentrates corrupted tokens near the clean target, whereas the mixture path keeps all incorrect bins equally likely (Appendix F.2). Both DF variants retain the clean-token prediction objective in Equation (2). Thus, label smoothing changes DR's prediction targets, whereas the metric path changes DF's corrupted inputs. The schedules and sampling rates are pro-

vided in Appendix F.2. The tested DR variant improves RoboCasa success by 2.1%. The best tested metric-path DF reaches 52.8% pooled success, compared with 53.9% for mixture-path DF. Explicit geometry therefore provides a modest gain for DR but does not improve the best observed DF result.

Structured Tokenizers Do Not Improve over Naive Binning. We next test whether changing the action representation improves control. We evaluate Finite Scalar Quantization(nFSQ) (Mentzer et al., 2024), residual vector quantization (RVQ) (Zeghidour et al., 2021), and FAST (Pertsch et al., 2025), alongside quantile-bin baselines. At their best recorded complete operating points, DF reaches 52.4% RoboCasa SR with nFSQ, 46.2% with FAST, and 38.2% with RVQ (Appendix F.1). nFSQ and quantile representation improves over the flat-bin baseline for DR, showing that the tokenizer can matter for a given configuration. However, none of these exceeds the broader campaign's best flat-bin DF result of 53.9%. The observed gains limited on DR, and do not raise the best recorded performance.

Discrete Policies Can Learn Local Bin Geometry. Why does encoding structural information yield little benefit? We hypothesize that discrete policy encode geometry through the **embedding vectors** associated with the last logit layer. Specifically, each bin b at coordinate j associated with a last-layer embedding $\mathbf{bv}_{j,b}$. For spatially adjacent bins $b, b+1$, we compute the vector cosine $\cos\theta_{j,b} := \langle \mathbf{v}_{j,b}, \mathbf{v}_{j,b+1} \rangle (\|\mathbf{v}_{j,b}\| \|\mathbf{v}_{j,b+1}\|)$. We find that adjacent bins exhibit high cosine similarities. These observations suggest discrete policies do not treat distinct bins as isolated categories, and offers one possible explanation for the limited benefit of imposing that geometry explicitly. Embedding plots, probes, and the selected-checkpoint comparison are provided in Appendix F.3.

Takeaway 3: Explicit geometry and structured tokenization can improve individual configurations, but neither exceeds the best configuration of DF with uniform masking and flat-bin tokenization. Learned local geometry offers a possible explanations for these limited gains.

6 Discussion and Future Work

We began by asking whether robot actions are fundamentally different from other AI modalities in requiring continuous prediction. Our results challenge this distinction. Although robots operate in a continuous physical world, simple action tokens can support precise control in both simulation and real-world tasks. Their success shows that discrete action prediction is more than a convenient interface to a language model. It is an effective way to learn robot control in its own right.

The key difficulty exposed by our experiments is deciding which token choices belong together. Discrete heads readily express action variation, but independent predictions can combine incompatible behaviors. Iterative refinement and guidance help resolve these conflicts while retaining responsiveness. This connects action prediction to a challenge also studied in diffusion language models: accurate token predictions must be coordinated to form a coherent joint output.

The need for joint consistency also suggests a different objective for action tokenization. Although the tested reconstruction-based tokenizers do not improve on our best uniform-bin DF result, tokenizers designed to make coupled action choices easier to predict may help reduce factorization error. For example, tokens could jointly encode coherent motion across several time steps. Their value may depend on the prediction head and the number of generation steps available. Finding tokenizations suited to discrete regression, iterative flow, or generation with only a few refinement steps is therefore a promising direction. Such tokenizers should be evaluated by the consistency and control performance of the resulting policies, alongside reconstruction quality.

More broadly, these findings support learning action generation within the same token-based framework as perception and reasoning. A natural next step is to test whether sharing representations and prediction machinery across modalities improves co-training and transfer, while preserving the action consistency needed for closed-loop control.These results make a strong case for treating discrete action prediction as a central component of future robot foundation models.

AI USE STATEMENT

We used generative AI tools to help implement methods, analyze results, make figures, and edit the paper. We did not use them to form hypotheses or design the experiments. We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

REPRODUCIBILITY STATEMENT

The evaluation protocol and environments are described in Appendix A, and the action heads, training objectives, architectures, and training recipes in Appendix B. Full results and selected settings are in Appendix C. We will release code and model checkpoints upon publication.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022. URL https://arxiv.org/abs/2209.15571.

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

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. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. URL https://arxiv.org/abs/1604.07316.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Feilong Tang, Donglin Wang, and Haoang Li. Unified diffusion VLA: Vision-language-action model via joint discrete denoising diffusion process. *arXiv preprint arXiv:2511.01718*, 2025. URL https://arxiv.org/abs/2511.01718.

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. *arXiv preprint arXiv:2303.04137*, 2023. URL https://arxiv.org/abs/2303.04137.

Eugenio Chisari, Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. *arXiv preprint arXiv:2409.07343*, 2024. URL https://arxiv.org/abs/2409.07343.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

Danny Driess, Jost Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *arXiv preprint arXiv:2407.15595*, 2024. URL https://arxiv.org/abs/2407.15595.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. URL https://arxiv.org/abs/2207.12598.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. AdaFlow: Imitation learning with variance-adaptive flow-based policies. *arXiv preprint arXiv:2402.04292*, 2024. URL https://arxiv.org/abs/2402.04292.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5ee7ed60a7e8169012224dec5fe0d27f-Abstract-Conference.html.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, Vitor Guizilini, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Muhammad Zubair Irshad, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Grace Tin Yan Lam, Pannag R. 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*, abs/2406.09246, 2024. URL https://api.semanticscholar.org/CorpusID:270440391.

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

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos Policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026. URL https://arxiv.org/abs/2601.16163.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026. URL https://arxiv.org/abs/2601.21998.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2503.00200.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Tian Nian, Shunbo Zhou, Xiaokang Yang, Jiangmiao Pang, et al. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. URL https://arxiv.org/abs/2209.03003.

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 *arXiv preprint arXiv:2108.03298*, 2021.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

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

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.

Khang Nguyen, An T. Le, Tien Pham, Manfred Huber, Jan Peters, and Minh Nhat Vu. FlowMP: Learning motion fields for robot planning with conditional flow matching. *arXiv preprint arXiv:2503.06135*, 2025. URL https://arxiv.org/abs/2503.06135.

NVIDIA, Johan Bjorck, Fernando Castañeda, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 2018. URL https://arxiv.org/abs/1811.06711.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much ado about noising: Dispelling the myths of generative robotic control. *arXiv preprint arXiv:2512.01809*, 2025. URL https://arxiv.org/abs/2512.01809.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. URL https://arxiv.org/abs/2501.09747.

Physical Intelligence et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1, 1988. URL https://papers.neurips.cc/paper_files/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L4uaAR4ArM.

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*, volume 35, 2022. URL https://arxiv.org/abs/2206.11251.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi (Jim) Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *ArXiv*, abs/2305.16291, 2023. URL https://api.semanticscholar.org/CorpusID:258887849.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. LLaDA-VLA: Vision language diffusion action models. *arXiv preprint arXiv:2509.06932*, 2025. URL https://arxiv.org/abs/2509.06932.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

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, 2021.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling fast and robust 3D flow-based policy via consistency flow matching for robot manipulation. *arXiv preprint arXiv:2412.04987*, 2024. URL https://arxiv.org/abs/2412.04987.

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. URL https://arxiv.org/abs/2304.13705.

A Environment Details

This section gives the evaluation measures, training recipes, and comparison protocol for the experiments in Sections 3 to 5.

A.1 Evaluation Measures and Comparison Protocol

Settings. Our evidence spans pretrained vision–language–action policies, including RoboCasa, and policies trained from scratch on RoboMimic and Push-T. Controlled function fitting uses separate models with a common trunk architecture to predict deterministic targets from a scalar input. These settings let us examine precision in robot policies and study fitting behavior where the target and codec error are known.

Evaluating action quality. We distinguish three measurements. *Prediction error* compares predicted and demonstrated actions, reporting training fit and held-out error separately. Robot translation errors use millimeters; Push-T uses pixels. *Temporal variation*, assessed through jerk, describes variation in the commanded trajectory. *Task performance* measures closed-loop success on RoboCasa, RoboMimic, and the Push-T runs in Figure 1.

Robot diagnostics use 5 training seeds with per-head checkpoints and inference budgets. Synthetic comparisons use 3 initialization seeds. Formal error definitions, benchmark recipes, and the chronology of the synthetic studies are in Appendices A and G.

A.2 From-Scratch Training Experiments

Single-task RoboMimic policies are state-based and trained from scratch, without a pretrained vision–language backbone. Their evaluation protocol and records are in Appendix C.1.

A.3 VLA-Scale Experiments

RoboCasa. RoboCasa (Nasiriany et al., 2026) with a Franka arm under the relative end-effector axis-angle action contract, one PaliGemma backbone, one data mixture, action horizon 24, shared quantile normalization statistics, seed 7. Evaluation replans every 24 steps, 10 trials per task, on three suites: atomic-seen (180 episodes), composite-seen (160), composite-unseen (160). We use the same whole-suite evaluation set for every head.

B Action Head and Objective Details

This appendix gives the training objectives, scalar codec, and sampling details for the policies introduced in Section 2. Table 2 summarizes the four head families. Per-experiment settings are given in Appendices A and G.

B.1 Continuous Heads

Continuous regression. For $L = HM$, the continuous-regression policy is trained with

$$\mathcal{L}_{\mathrm{CR}} = \mathbb{E}_{(o,\mathbf{a}_{1:H})\sim\mathcal{D}}\left[\frac{1}{L}\|\pi_\theta^{\mathrm{CR}}(o) - \mathbf{a}_{1:H}\|_F^2\right], \qquad \hat{\mathbf{a}}_{1:H} = \pi_\theta^{\mathrm{CR}}(o). \tag{5}$$

The unrestricted population minimizer is $\mathbb{E}[\mathbf{a}_{1:H} \mid o]$. When several actions are plausible under the same conditioning, this mean need not itself correspond to a demonstrated action. Auxiliary random inputs are suppressed in this notation; the expectation also averages over them when present.

Table 2: Four heads: representation crossed with inference computation. Losses and decoders are described at the head-family level; per-experiment details are in Appendices A and G.

Head	Output	Training loss	Generation
CR	real-valued chunk	squared error on the chunk	one pass
CF	real-valued chunk	velocity or clean-target MSE	N integration steps from noise
DR	one token per coordinate	token cross-entropy (hard or distance-smoothed)	one pass, argmax on every token
DF	one token per coordinate	clean-token cross-entropy on corrupted chunks	N updates, conditioned on the current chunk

Continuous flow. Let $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ be independent Gaussian noise with the same shape as $\mathbf{a}_{1:H}$. For $t \sim \mathcal{U}[0,1]$ and a continuously differentiable, nondecreasing schedule satisfying $\alpha(0) = 0$ and $\alpha(1) = 1$, define the stochastic interpolant (Albergo & Vanden-Eijnden, 2022) $I_t = (1-\alpha(t))\boldsymbol{\epsilon} + \alpha(t)\mathbf{a}_{1:H}$ and its derivative $U_t(\mathbf{a}_{1:H}, \boldsymbol{\epsilon}) = \dot{\alpha}(t)(\mathbf{a}_{1:H} - \boldsymbol{\epsilon})$. The velocity field is trained by flow matching (Lipman et al., 2022; Liu et al., 2022):

$$\mathcal{L}_{\mathrm{CF}} = \mathbb{E}_{\substack{(o,\mathbf{a}_{1:H})\sim\mathcal{D} \\ t\sim\mathcal{U}[0,1],\ \boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}} \left[\|v_\theta(I_t, t, o) - U_t(\mathbf{a}_{1:H}, \boldsymbol{\epsilon})\|_F^2\right]. \tag{6}$$

At inference, use a grid $0 = t_0 < \cdots < t_N = 1$ and Euler updates

$$\mathbf{x}_{t_{n+1}} = \mathbf{x}_{t_n} + (t_{n+1} - t_n) v_\theta(\mathbf{x}_{t_n}, t_n, o), \qquad \mathbf{x}_{t_0} = \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{7}$$

for $n = 0, \ldots, N-1$. The resulting policy is $\pi^{\mathrm{CF}}_{\theta,N}(o; \boldsymbol{\epsilon}) := \mathbf{x}_{t_N}$.

B.2 Scalar Action Quantization

Actions are normalized and clipped according to the recipe in Appendix A. For coordinate i, let $[\ell_i, u_i]$ be its quantization range and $\Delta_i = (u_i - \ell_i)/B$ its bin width. The ranges may differ across coordinates. With token position $j = (h-1)M + i$, the uniform quantizer and bin-center decoder are

$$\begin{aligned} Q(\mathbf{a}_{1:H})_j &:= \mathrm{clip}_{[1,B]}\left(1 + \left\lfloor \frac{a_{h,i} - \ell_i}{\Delta_i} \right\rfloor\right), \\ [D_Q(\mathbf{z})]_{h,i} &:= \ell_i + (z_j - \tfrac{1}{2})\Delta_i. \end{aligned} \tag{8}$$

Clipping keeps the index in $[B]$, including at the upper endpoint. The quantizer flattens the chunk in a fixed order, and the decoder restores its $H \times M$ shape. Both maps use normalized action coordinates; normalization is inverted before execution. Thus B is both the number of bins per coordinate and the action-token vocabulary size. Unless otherwise stated, $B = 512$. Alternative tokenizations, including quantile-based bins and more compressive encoders, are described in Appendix F.1.

B.3 Discrete Training Objectives

For $\mathbf{z} = Q(\mathbf{a}_{1:H})$, discrete regression is trained by tokenwise cross-entropy:

$$\mathcal{L}_{\mathrm{DR}} = \mathbb{E}_{(o,\mathbf{a}_{1:H})\sim\mathcal{D}}\left[-\frac{1}{L}\sum_{j=1}^{L} \log p^{\mathrm{DR}}_{\theta,j}(z_j \mid o)\right]. \tag{9}$$

The head outputs $L \times B$ logits, with a softmax over bins at each position (Brohan et al., 2022). Conditional on the model inputs, its factorized distribution is $p^{\mathrm{DR}}_\theta(\mathbf{z} \mid o) = \prod_{j=1}^{L} p^{\mathrm{DR}}_{\theta,j}(z_j \mid o)$. Consequently, maximizing this model's joint probability reduces to selecting each token's mode (Devlin et al., 2019): $\hat{z}_j = \arg\max_{k\in[B]} p^{\mathrm{DR}}_{\theta,j}(k \mid o)$. The policy returns $\pi^{\mathrm{DR}}_\theta(o) := D_Q(\hat{\mathbf{z}})$.

Discrete flow also predicts the clean token at each position, but conditions on the corrupted chunk and flow time. We sample $t \sim \mathcal{U}[0,1]$ and $\mathbf{z}_t \sim q_t(\cdot \mid \mathbf{z})$ using Equation (1), and train with clean-token cross-entropy (Gat et al., 2024):

$$\mathcal{L}_{\mathrm{DF}} = \mathbb{E}_{\substack{(o,\mathbf{a}_{1:H})\sim\mathcal{D} \\ t\sim\mathcal{U}[0,1] \\ \mathbf{z}_t\sim q_t(\cdot|\mathbf{z})}}\left[-\frac{1}{L}\sum_{j=1}^{L} \log p^{\mathrm{DF}}_{\theta,j}(z_j \mid \mathbf{z}_t, t, o)\right]. \tag{10}$$

DR variants with distance-smoothed targets are specified below.

Per-benchmark DR loss. Four action heads cross representation with decoding: continuous regression (CR), continuous flow matching (Lipman et al., 2022) (CF), discrete regression over flat bins with per-token argmax (DR), and discrete flow over flat bins (DF). An autoregressive head (AR) appears in the FAST tokenizer validity and closed-loop comparisons. The DR loss differs between benchmarks. On RoboCasa it is a distance-kernel soft cross-entropy with kernel temperature $\tau = 0.0025$ in normalized action units, so the neighbor bin's unnormalized weight is $\exp(-0.0078/0.0025) \approx 0.045$ of the target bin's. On Push-T it is hard-label cross-entropy. Statements about "cross-entropy without a metric" therefore apply per run.

B.4 Discrete-Flow Source and Sampler

The source factorizes as $q_0(\mathbf{z}_0) = \prod_{j=1}^{L} q_{0,j}(z_{0,j})$. A uniform source has $q_{0,j}(k) = 1/B$ for $k \in [B]$ (Gat et al., 2024); a masked source places all mass on an auxiliary mask token (Sahoo et al., 2024). The mask is an input symbol and does not decode to an action bin. In either case, the clean-token predictor outputs distributions over $[B]$.

The schedule is continuously differentiable and nondecreasing, with $\alpha(0) = 0$, $\alpha(1) = 1$, and $\alpha(t) < 1$ for $t < 1$. For the grid $0 = t_0 < \cdots < t_N = 1$, the mixture-path update in Equation (3) uses $\eta_n = (\alpha(t_{n+1}) - \alpha(t_n))/(1 - \alpha(t_n))$. Proposals and replacement decisions are independent across positions conditional on the current chunk. This rule gives an Euler approximation, in the interpolation coordinate α, to the associated continuous-time Markov chain. Because $\eta_{N-1} = 1$, the final update replaces every position with a clean-token proposal, so the output contains no mask tokens.

The policy is $\pi^{\mathrm{DF}}_{\theta,N}(o;\xi) := D_Q(\mathbf{z}_{t_N})$, where ξ collects randomness in initialization, proposals, and replacement decisions. We suppress N and ξ when clear from context. For a uniform source, tokens updated at earlier steps may be revised later. Subsequent predictions condition on the updated chunk, allowing the generation process to capture dependencies across positions despite the conditional independence of updates within each step.

B.5 Classifier-Free Guidance

CFG (Ho & Salimans, 2022) combines predictions with conditioning c and its null counterpart $\emptyset$. Suppressing other model inputs, discrete heads use

$$p^{\mathrm{CFG}}_{\theta,j}(k \mid c) \propto p_{\theta,j}(k \mid c)^{1+\lambda} p_{\theta,j}(k \mid \emptyset)^{-\lambda}, \qquad k \in [B], \quad \lambda \geq 0, \tag{11}$$

with normalization over bins. Equivalently, if $\ell^c_{\theta,j}$ and $\ell^u_{\theta,j}$ are conditional and unconditional logits, apply softmax to $(1+\lambda)\ell^c_{\theta,j} - \lambda\ell^u_{\theta,j}$. DR greedily decodes these distributions; DF samples clean-token proposals from them. For CF, each integration step instead uses $v^{\mathrm{CFG}}_\theta(\mathbf{x}_t, t, c) = (1+\lambda)v_\theta(\mathbf{x}_t, t, c) - \lambda v_\theta(\mathbf{x}_t, t, \emptyset)$. We apply CFG to DR, DF, and CF, and omit it for CR. Under this convention, $\lambda = 0$ gives the unguided conditional prediction.

B.6 Omitted Details for Policy Architecture

The heads use the same backbone design. CR and DR receive auxiliary random inputs independent of the target action chunk, so random inputs are present for both regression and flow heads. These auxiliary inputs are suppressed in the preceding notation Within each setting, all heads see the same observations and training data. This section gives the networks that map these observations to actions; the losses and samplers are given in Appendices B.1 and B.3.

B.6.1 Single-Task From-Scratch Architectures

RoboMimic policies observe the object state and the end-effector position, orientation, and gripper state of each arm, over the last two time steps. An MLP encodes the flattened observation. Each policy predicts $H = 10$ actions and executes 8 before replanning. An action is a 10-dimensional absolute end-effector command: 3D position, 6D rotation, and gripper (20 dimensions for the two-arm Transport task). Actions are normalized per dimension to $[-1, 1]$. Discrete heads use $B = 512$ bins per coordinate. Tab. 3 lists the two network families.

Table 3: Networks for the single-task RoboMimic policies. Both families are DiT-style transformers conditioned through adaptive layer norm (adaLN) on the observation embedding and the flow time. Parameter counts include the observation encoder and are given for single-arm tasks.

	CR, CF	DR, DF
Blocks	8 encoder + 8 decoder	12
Width / heads / MLP width	256 / 4 / 1024	256 / 4 / 1024
Action tokens	one per time step (10)	one per time step and coordinate (100)
Action input	MLP on the 10-d action	learned embedding of B bins
Action output	linear to 10 values	logits over B bins
Observation MLP width	256	512
Dropout	0	0.1
Parameters	16.4M	15.8M

CF integrates its velocity field with 9 Euler steps. DF runs 100 update steps with the mixture path $\kappa_t = t^3$, followed by one argmax prediction. All models are trained for 300k steps with AdamW (learning rate 10^{-4}, weight decay 10^{-5}, gradient clipping at 10) and an exponential moving average of the weights with decay 0.995, which is used for evaluation. Continuous heads use batch size 1024 and cosine decay; discrete heads use batch size 512, a 5% linear warmup, and cosine decay. Each run uses one L40S GPU. The synthetic-fitting trunk is described in Appendix G.

B.6.2 VLA-Scale Architectures

All four heads share a PaliGemma-3B backbone (Beyer et al., 2024): a SigLIP So400m/14 image encoder and a Gemma-2B language model. We initialize it from the π_0 vision–language weights (Black et al., 2024) and fine-tune it fully. The backbone reads up to three 224×224 camera images (256 tokens each) and up to 48 text tokens with bidirectional attention. A resampler with 64 learned queries (2 blocks, width 1536) compresses the final-layer backbone states, together with one proprioception token, into 64 condition tokens.

The action head is a separate transformer with 8 blocks, width 1536, 12 attention heads, and MLP width 6144. Each block applies self-attention over the action tokens, cross-attention to the 64 condition tokens, and an MLP. The head does not share attention layers with the backbone. It predicts the $H = 24$ action chunk with one token per action coordinate: 3D position change, axis-angle rotation change, and gripper. Learned position and coordinate embeddings and Fourier features of the flow time are added to every token. Tab. 4 lists the parts that differ across heads.

Table 4: Action heads for the VLA-scale policies. All heads use the same 8-block cross-attention transformer (width 1536) and resampler. Parameter counts cover the action head and resampler; the backbone adds 2.92B parameters.

	CR	CF	DR	DF
Parameters	367M	367M	369M	371M

Bins are uniform over the normalized action range. DF uses a uniform source, and the reported DF policy uses the schedule $\alpha(t) = t^3$. The condition is dropped with probability 0.1 and replaced by a learned null condition, which enables CFG at inference (Appendix B.5). All heads are trained with AdamW (learning rate 7.5×10^{-5}, $\beta = (0.9, 0.95)$, weight decay 10^{-4}, gradient clipping at 1), linear warmup, and cosine decay to $0.1\times$ the peak learning rate, in bf16 on 8 B200 GPUs. The global batch size is 1024 (512 for DF).

C Full Performance Results

C.1 RoboMimic Results

RoboMimic policies are state-based and trained from scratch on each task. For each task and head family, Fig. 1 reports the best success rate over action space (absolute or delta), tokenizer (flat or quantile bins), flow variant, and guidance scale. Each evaluation uses 32 episodes.

Fig. 6 reports how closely each absolute-action run fits its training data. For each run, we show the end-effector (EEF) position error on training samples at its last 10 evaluations.

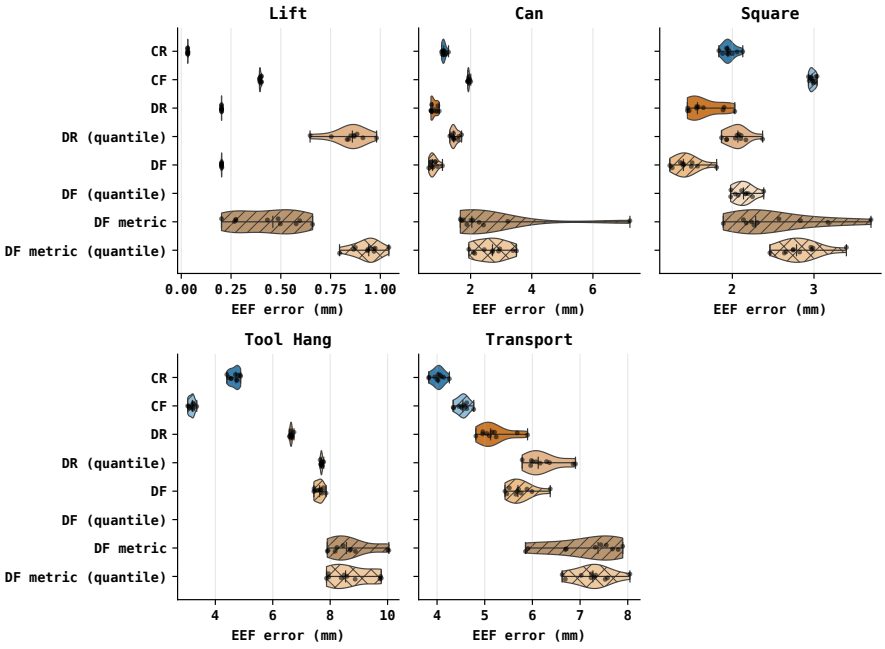


Figure 6: **EEF position error on RoboMimic training samples.** Each violin shows one run's RMS EEF position error (mm) over its last 10 evaluations; dots are individual evaluations and the vertical line marks the median. The spread reflects late-training variation of the metric. DF denotes the mixture path and DF metric the metric-induced path (Appendix F.2); "quantile" marks equal-frequency bins, otherwise bins are uniform.

C.2 Backbone Variants

We also train each head with two other backbone settings: a fine-tuned backbone initialized from the $\pi_{0.5}$ weights, and a frozen backbone initialized from the π_0 weights. Only the action head and resampler are trained in the frozen setting. All other settings follow Appendix B.6.2. Tab. 5 reports, for each head, the operating point with the highest pooled success over checkpoints, guidance scales $\lambda \in \{0, 2\}$, and NFE $\in \{5, 10, 20\}$ for the flow heads.

Table 5: RoboCasa success counts with other backbone settings. Each row is the best operating point for that head; all three suites are evaluated at that point. Replan horizon 24.

Backbone	Head	CFG / NFE	Atomic	Composite seen	Composite unseen	Overall SR (%)
$\pi_{0.5}$, fine-tuned	CR	0 / –	117/180	39/160	56/160	42.4
$\pi_{0.5}$, fine-tuned	CF	2 / 20	148/180	69/160	74/160	58.2
$\pi_{0.5}$, fine-tuned	DR	2 / –	144/180	62/160	73/160	55.8
$\pi_{0.5}$, fine-tuned	DF	2 / 5	148/180	75/160	78/160	60.2
π_0, frozen	CR	0 / –	49/180	1/160	1/160	10.2
π_0, frozen	CF	2 / 20	55/180	6/160	6/160	13.4
π_0, frozen	DR	2 / –	45/180	4/160	2/160	10.2
π_0, frozen	DF	2 / 5	49/180	5/160	2/160	11.2

D Temporal Variation and Policy Responsiveness Experiments

This section supplements Section 3.

D.1 Temporal Variation on Held-Out Chunks

Figure 2: offline replay. Each of eight sources supplies 1,000 matched 24-step chunks, giving 32,000 model–chunk observations. All heads use CFG 0; both flow heads use NFE 10. Jerk is the mean norm of third finite differences of translation commands, without physical-time normalization. The ratio divides predicted jerk by target jerk after the same normalization and clipping.

The logarithmic scatter and its coordinate-wise medians omit 488 exact-zero ratios and 400 zero-target-denominator observations. All positive finite ratios remain. EEF violins retain all observations, with independently normalized density widths, medians, and interquartile ranges. Their horizontal positions are for layout only. Vertical scales vary by source. Including zero numerators wherever the denominator is positive, CR's per-source median ratio ranges from 0.036 to 0.286, and CF's from 0.231 to 0.549. CR lies below one on 98.8–99.8% of defined-ratio chunks and CF on 85.0–98.1%. The corresponding median ranges are 0.294–1.313 for DR and 0.343–0.589 for DF. DF's EEF-error 95th percentile is below DR's on seven sources, with LIBERO-plus the exception.

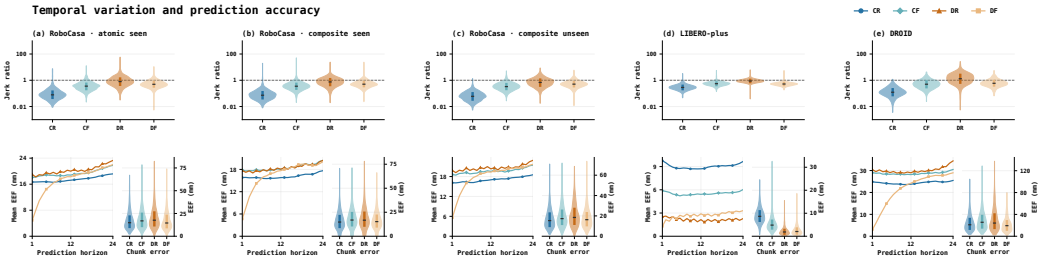


Figure 7: **Per-source view of Figure 2.** Top: jerk-ratio violins for each of the five displayed sources. Bottom: mean EEF error by horizon (left axis) and chunk-error violins (right axis).

D.2 Ground-Truth Action Error Under CFG

Tab. 1 reports that CFG raises DF's EEF error. To find the source of this increase, we sweep the CFG weight from $\lambda = 0$ to 3 in steps of 0.25. We use 4,000 held-out observations, 500 from each of 8 validation sources (DROID, LIBERO, and six RoboCasa splits). The predictions come from the dense CFG sweep in Fig. 4(c). Error is the per-step XYZ Euclidean distance to the ground-truth action in physical units, averaged over the valid steps of the chunk, then over draws, then over observations. All heads are scored against the same target.

Pooled over all sources (Fig. 8a). DF has lower error than DR at small λ: 15.16 mm versus 18.06 mm unguided. DF's error is lowest at $\lambda = 0.5$ (14.86 mm) and then rises to 17.97 mm at

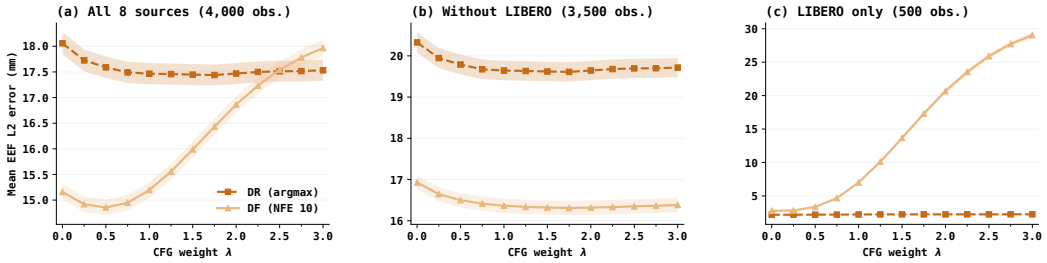


Figure 8: **EEF error against ground truth as a function of the CFG weight λ.** Mean XYZ end-effector L2 error (mm) over the 24-step chunk, for DR (argmax decoding) and DF (NFE 10, mean over 32 draws). Shaded bands show ± 1 standard error over observations. $\lambda = 0$ is unguided (Appendix B.5). **(a)** All 8 validation sources. **(b)** The 7 sources other than LIBERO. **(c)** LIBERO only.

Table 6: Proposal spread and response to one-step simulator disturbances. σ is mean nominal RMS spread over the full bank. The independent 32-vs-32 transport statistics use the existing 24-snapshot subsample per head and every other perturbation. Triples correspond to $\epsilon = 0.1, 0.3, 1.0$. Relative excess is computed from the displayed rounded transport means. CR's deterministic law has zero floor, so its relative excess is undefined. Spread and transport use different subsets.

Head	σ	F_h	$T_h - F_h$	R_h
DF	1.217	1.245	0.233 / 0.490 / 1.071	0.19 / 0.39 / 0.86
CF	0.764	0.591	0.036 / 0.102 / 0.403	0.06 / 0.17 / 0.68
DR	0.182	0.141	0.301 / 0.529 / 0.927	2.13 / 3.75 / 6.57
CR	0.000	0.000	0.147 / 0.275 / 0.568	undefined

$\lambda = 3$, passing DR at about $\lambda = 2.5$. DR's error falls slightly with guidance and is nearly flat for $\lambda \geq 1$ (17.44–17.53 mm).

D.3 Distributional Robustness and the Cost of Contraction

Question **Can a policy preserve diverse actions while remaining robust and responsive to observation changes?**

Our tests reveal two sides of the tradeoff: flow heads retain broader proposals with smaller relative disturbance responses, but excessive contraction can suppress the action changes required by real trajectories.

D.3.1 Flows Retain Broader Proposals With Smaller Relative Changes

We perturb executed actions, render the resulting observations, and compare the distributions of predicted chunks at CFG 0 (NFE 10 for flows). Table 6 reports RMS proposal spread σ, same-observation sampling distance F_h, and disturbance-induced distance T_h, measured by empirical Wasserstein distance W_2. Relative excess transport is $R_h = (T_h - F_h)/F_h$; it describes change relative to the sampling baseline. Definitions and protocol are in Appendix D.4.

Flow proposals are broader than regression proposals, with DF broader than CF ($\sigma = 1.217$ versus 0.764). Their disturbance responses are also smaller relative to the same-observation sampling baseline. At $\epsilon = 0.1$, excess transport is 19% of that baseline for DF and 6% for CF, versus 213% for DR. DR's narrow proposal distribution shifts substantially relative to its own variability; CR also shifts despite having no proposal spread. This suggests a mechanism by which regression errors could compound into out-of-distribution drift.

D.3.2 CF's Stronger Contraction Comes at a Cost in Responsiveness

CF stays closer to nearby training actions: its expert-calibrated excess distance is -0.100, versus -0.015 for DF. This apparent adherence can also make a policy too "sticky." We test that possibility on real frame pairs, comparing the head's raw proposal-mean change with the expert's chunk change under the same state displacement. Dividing either change by $\|\Delta s\|$ defines an empirical Lipschitz-like response; the table reports the underlying magnitudes directly. Sampling floors are not subtracted.

DF responds more than CF and is closer to the expert change at every tested lag (Table 7). At lag one, the expert shift is 0.66, compared with 0.45 for DF and 0.36 for CF; at lag 16, the values are 4.32, 3.55, and 3.28. Both flows underfit the observed response magnitude, but CF does so more strongly. DR tracks the expert magnitude most closely, showing that responsiveness on real trajectories and stability under disturbances are distinct requirements.

Table 7: Action changes on held-out expert trajectories. $\|\Delta s\|$ is the state displacement and expert Δchunk is the demonstrated chunk change. Head columns report raw proposal-mean shifts; parentheses show same-observation sampling floors for the stochastic heads. All values are medians over pairs in normalized units; no floor subtraction or response-ratio normalization is applied.

Lag	$\|\Delta s\|$	Expert Δchunk	DR (floor)	CR	DF (floor)	CF (floor)
1	0.018	0.66	0.71 (0.05)	0.35	0.45 (0.38)	0.36 (0.21)
2	0.036	1.13	1.16 (0.05)	0.62	0.77 (0.37)	0.64 (0.22)
4	0.073	1.88	1.77 (0.05)	1.09	1.32 (0.38)	1.18 (0.21)
8	0.149	3.00	2.71 (0.05)	1.93	2.34 (0.38)	2.25 (0.22)
16	0.292	4.32	4.01 (0.05)	3.13	3.55 (0.38)	3.28 (0.22)

Takeaway Flow preserves broader proposals with smaller relative disturbance responses. Within the flow family, CF is more contracted toward training actions but under-responds more to real trajectory changes; DF retains broader proposals and more of the expert's response. This is the tradeoff between distributional stability and responsive control.

D.4 Disturbance and Expert-Reference Diagnostic Protocol

This appendix gives the measurement definitions and sampling protocol for Appendix D.3. DR uses its deployed argmax decoder; its residual proposal spread comes from varying random source-token inputs. CR is deterministic.

D.4.1 Measuring a Distribution's Response to Disturbance

Let $p_h(\cdot \mid o)$ be head h's law over decoded, normalized action chunks. Its spread and its sensitivity are different quantities:

$$\sigma_h(o) = \left(\mathbb{E}_{A\sim p_h(\cdot|o)}\|A - \mu_h(o)\|_2^2\right)^{1/2}, \tag{12}$$

$$W_2(p,q) = \left[\inf_{\gamma\in\Pi(p,q)} \mathbb{E}_{(A,B)\sim\gamma}\|A - B\|_2^2\right]^{1/2}, \tag{13}$$

where $\mu_h(o) = \mathbb{E}[A \mid o]$ and $\Pi(p,q)$ denotes joint distributions with marginals p, q. Spread measures diversity at one observation; $W_2(p_h(\cdot \mid o), p_h(\cdot \mid o'))$ measures how far probability mass must move after a disturbance. Neither measures support cardinality.

We intervene on the simulator, because adding noise only to the proprioceptive input produced no measurable response. At 144 snapshots from DF rollouts across 18 RoboCasa atomic tasks, we restore the simulator state and execute either the nominal action or an action perturbed by $\eta \sim \mathcal{N}(0, \epsilon^2 I)$ in normalized action coordinates. After one control step, we render the resulting observations and query each head at CFG 0 (NFE 10 for flows), drawing 64 chunks per observation.

Relative change requires a same-observation baseline. Even two independent finite samples from the same law have nonzero empirical W_2. Split the nominal draws into independent groups C_1, C_2 and use perturbed draws P_2 with source noise independent of C_1. Each group contains 32 chunks. Exact optimal assignment gives

$$F_h = \langle \widehat{W}_2(C_1, C_2) \rangle, \qquad T_h(\epsilon) = \langle \widehat{W}_2(C_1, P_2) \rangle, \qquad R_h(\epsilon) = \frac{T_h(\epsilon) - F_h}{F_h}. \tag{14}$$

Brackets average over the evaluated snapshots and perturbations; R_h is a ratio of aggregate means. We call it *relative excess transport*: how much the measured between-observation distance exceeds the within-observation sampling baseline. For example, $R = 0.2$ means 20% above that baseline. The ratio depends on sample count and distribution shape; a large floor can reduce it. We therefore report spread, absolute excess, and relative excess together.

Reference adherence must be calibrated against real actions. For held-out observation o, let $M(o)$ contain the $k = 5$ nearest training action chunks, with neighbors selected in normalized proprioception and restricted to the same instruction when enough matches exist. Define

$$r_0(A; o) = \min_{B \in M(o)} \|A - B\|_2, \qquad E_h = \frac{\mathbb{E}_o[\mathbb{E}_{A \sim p_h} r_0(A; o) - r_0(A_{\mathrm{gt}}; o)]}{\mathrm{median}_o \, r_0(A_{\mathrm{gt}}; o)}. \tag{15}$$

We compute E_h within each source, then average over eight sources (512 held-out observations per source, 16 proposals each). The held-out expert reference matters: genuine demonstrated actions also lie some distance from a finite training set. A negative excess means predictions lie closer to this reference than held-out demonstrations do.

Gripper support. Under a bin-scale tolerance, the fraction of gripper commands outside the demonstrated values is about 0.22 for discrete heads and 0.78–0.94 for continuous heads.

The raw on-data shifts in Table 7 compare change magnitudes. Adjacent chunks also shift their temporal window, so the comparison includes replanning alignment. Sampling floors are reported separately and are not subtracted from these shifts.

The one-step disturbance changes normalized proprioception by about 0.003–0.021, below the reference data's median neighbor spacing of 0.32. The diagnostics use CFG 0 and NFE 10, while reported control success uses separately selected settings.

E Factorization Error Experiments

This section supplements Section 4.

E.1 Experimental Methodology

Mode-detection criteria. For each translation-token distribution $p_j(b)$, we smooth probabilities along the bin axis with a Gaussian kernel of standard deviation two bins, normalized over the available bins at each location. We identify local maxima, including endpoint maxima, in the smoothed distribution. Each bin is assigned to its nearest peak, and each peak's mass is the sum of the *unsmoothed* probabilities assigned to it. We retain peaks with at least 10% mass. Two adjacent retained peaks qualify as separated modes if they are at least eight bins apart and the minimum smoothed probability between them is at most 50% of the smaller peak height. Each translation token represents one coordinate at one predicted time step. We call it ambiguous if it has a qualifying pair of modes, and count it once regardless of how many such pairs it has. The mean count averages ambiguous tokens per chunk. The threshold statistic reports the fraction of chunks containing at least k ambiguous tokens. For DF, detection is performed on each final conditional token prediction before averaging over refinement draws.

NFE sweep in Figure 4(a,b). We use 4,000 validation observations (500 from each of eight sources) and 16 DF draws per observation at NFE $1, 2, 5, 10, 20$ and CFG 0. Each chunk has 39 translation tokens: three coordinates at 13 time steps. Panel (a) reports the mean number of ambiguous tokens per chunk under the mode detector above; Figure 9(a) gives the threshold curves.

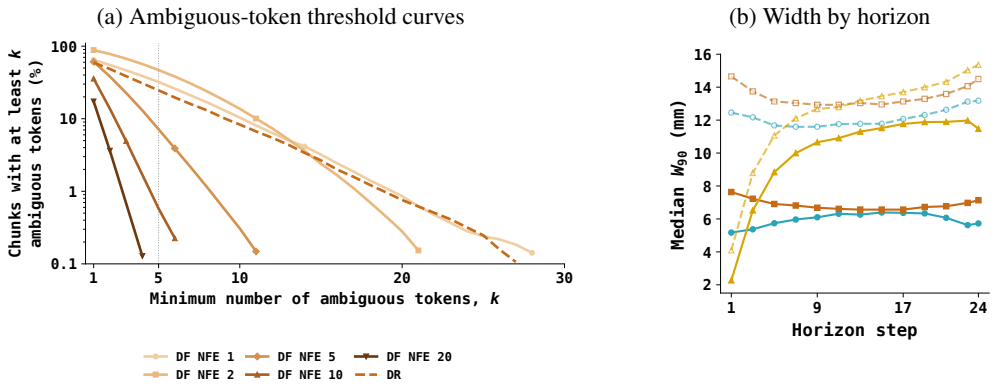


Figure 9: **Ambiguity thresholds and width by horizon.** (a) Percentage of chunks with at least k ambiguous tokens at CFG 0, on the same observations and draws as Fig. 4(a). The log axis stops at 0.1%. (b) Median central-90% width W_{90} at each horizon step for CF (teal), DR (orange), and DF (gold). Dashed lines with open markers are CFG 0; solid lines with filled markers are CFG 2.

Panel (b) reports percentiles of $\exp(\mathcal{H}(p))$ over all translation tokens, draws, and observations. DR references use the same observations and detector. Because the final query time is $t = 1 - 1/\mathrm{NFE}$, the sweep changes both the number of refinement steps and the time of the last prediction. The mean count rises from 3.62 at NFE 1 to 4.94 at NFE 2, then falls to 1.50, 0.57, and 0.22 at NFE 5, 10, and 20 (DR: 3.01). Figure 11 reports entropy and action diversity for the same sweep.

E.2 Factorization Error and L_2 Loss Diagnostic Results

Tab. 8 reports the L_2 prediction errors referenced in Section 3. The remaining figures give additional mode-ambiguity diagnostics for Fig. 4; Fig. 9(b) breaks W_{90} down by horizon.

Table 8: EEF target-prediction error (↓). Slow-validation translation error in mm near 100k training steps; CF uses 100 integration steps and DF 10. Regression uses one pass. These are single-run diagnostics without repeated-seed uncertainty.

Dataset / source	CR	CF	DR	DF
Slow validation (mm)				
DROID	21.60	26.60	25.09	19.67
LIBERO-plus	9.88	7.23	2.53	2.90
RoboCasa pretrain composite	16.44	17.34	15.97	14.84
RoboCasa MimicGen	16.18	16.95	15.76	12.55
RoboCasa target atomic	17.34	18.97	19.55	15.62
RoboCasa composite-seen	12.11	14.87	15.39	12.93

E.3 Task Performance Under Varying NFE and CFG

E.3.1 Task Performance Under Varying NFE

We evaluate the RoboCasa checkpoints for each flow head, at NFE $\in \{5, 10, 20\}$ and CFG $\lambda \in \{0, 2\}$. In general, DF is insensitive to NFE: its mean success stays within 0.5 points across NFE without guidance and within 1.1 points with guidance. CF is also flat without guidance. With guidance, CF drops by 8 points at NFE 5 and recovers at NFE 10 and 20.

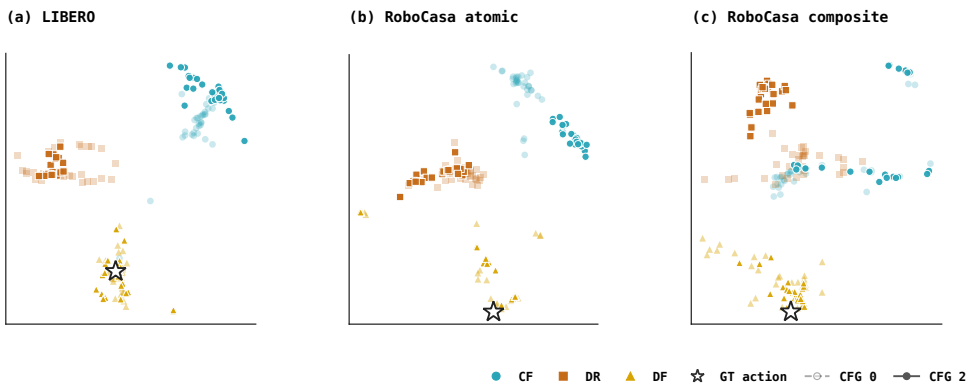


Figure 10: **Additional t-SNE views of sampled action chunks.** LIBERO, RoboCasa atomic, and RoboCasa composite supplement the DROID example in Figure 4. Colors and marker shapes identify CF, DR, and DF; faded and solid marks denote CFG 0 and CFG 2, respectively. Stars denote reference actions. The original embeddings are preserved; these projections provide a qualitative view of concentration under guidance.

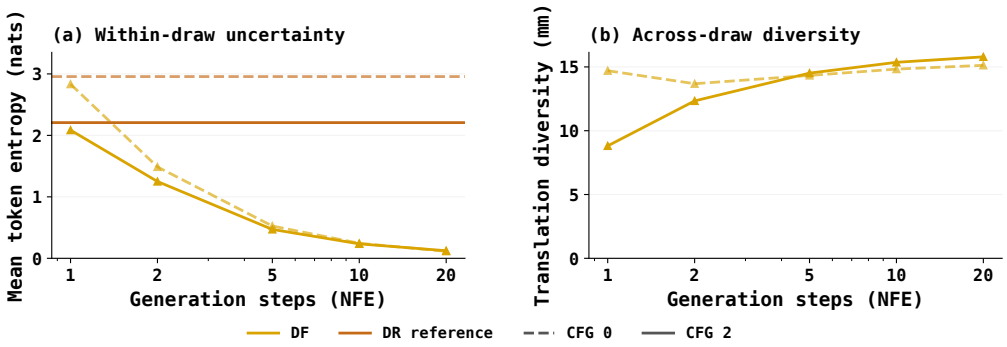


Figure 11: **Refinement sharpens conditionals while retaining across-draw variation.** Mean within-draw translation-token entropy (a) and across-draw translation-command diversity (b) use the same 4,000 validation observations and 16 draws per observation as Figure 4. Gold curves show DF; orange horizontal lines are DR references. Dashed and solid curves denote CFG 0 and CFG 2. From NFE 5 to 20, entropy decreases while action diversity persists. Diversity is the RMS pairwise Euclidean translation-command distance across draws, averaged over horizon positions and observations, in mm.

E.4 Visualization of Mode Inconsistency in RoboCasa

RoboCasa move–hold example. Figure 12 uses CloseFridge marginals and a schematic mode illustration: moved thumbnails are taken from steps 12 and 24 of a DR-argmax replay; the hold thumbnail repeats the initial image.

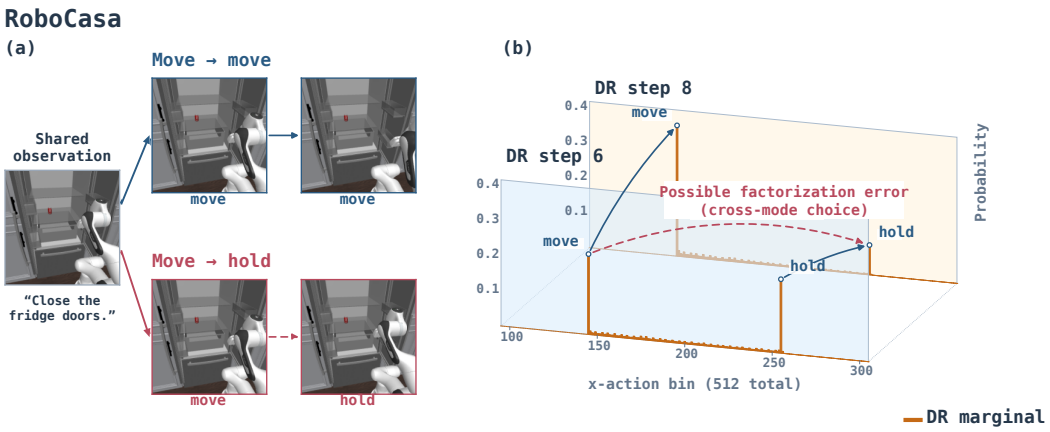


Figure 12: **a** Robocasas example showing that in the same chunk, robot arm has two mode one is move consistenly, the second is first move then hold; **(b** We show that this independent token sampling could cause factorization error, leads to swithcing between motions on different timestep, the same as what we have seen in push-T Fig. 5

F Tokenization and Geometric Priors

This section supplements Section 5.

F.1 Performance Under Different Tokenization

Closed-loop results for tokenizers. Table 9 compares nFSQ, RVQ, and FAST with their frequency-bin baselines. For each tokenizer and head, we report the operating point with the best pooled success, selecting CFG, the replanning horizon, and, for DF, the NFE among 5, 10, and 20. The same setting is used on all three suites. Pooled success combines 180 atomic-seen episodes and 160 episodes from each composite suite.

The nFSQ result (52.4%) is slightly above the frequency-bin metric-path result (51.8%). Neither exceeds the main mixture-path DF result of 53.9%, which uses a different recipe and horizon.

FAST framing and validity. FAST compresses action sequences (Pertsch et al., 2025), so generation must respect its sequence grammar. During training, valid decodes were near 100% for AR, 25–40% for DF, and 0% for DR. The FAST DR run still had 0% validity at the end of training, so it has no closed-loop evaluation. Flat scalar bins do not have this failure: every token sequence decodes. With EOS-terminated framing, FAST AR reaches 48.2% with CFG 2; FAST DF reaches 46.2%.

Tokenizer	Head	CFG	NFE	Atomic	Composite seen	Composite unseen	Overall SR (%)
Frequency bins	DR	0	–	140/180	57/160	52/160	49.8
	DF	2	20	128/180	50/160	52/160	46.0
Frequency bins + metric path	DF	2	20	136/180	67/160	56/160	51.8
nFSQ	DR	2	–	121/180	40/160	47/160	41.6
	DF	2	5	139/180	61/160	62/160	52.4
RVQ	DR	0	–	59/180	12/160	1/160	14.4
	DF	2	50*	113/180	40/160	38/160	38.2
FAST	AR	2	–	131/180	56/160	54/160	48.2
	DF	2	20	123/180	55/160	53/160	46.2

*RVQ needs more NFE to decode, so DF RVQ uses NFE 50.

Table 9: **RoboCasa success by tokenizer and head.** Each row is the best operating point for that tokenizer and head. No tokenizer lifts DF above the best flat-bin DF. Frequency bins behave differently for DR: CFG 0 and CFG 2 give almost the same success (49.8% and 49.0%), and at CFG 0 DR is far above flat-bin DR (25.8%, Section 4). With frequency bins every token is used equally often, so each token carries more information and should be harder to predict, yet single-pass DR handles it well. We do not yet have a concrete explanation, but the results suggest that when inference computation is limited, how actions are encoded matters more.

F.2 Geometry-Aware Probability Paths and Evaluations

This appendix specifies the metric-path interventions summarized in Section 5. The path illustration is analytical; the success rates below come from trained-policy evaluations.

F.2.1 From a Mixture Path to a Metric-Induced Path

Let $y \in \{0, \dots, B-1\}$ denote a clean action token and let $s \in [0, 1]$ increase from noise toward the clean action. A probability path specifies the distribution of corrupted tokens presented to the model at each stage. With a uniform source, the mixture path is

$$p_s^{\text{mix}}(k \mid y) = \frac{1-s}{B} + s\,\mathbf{1}\{k = y\}. \tag{16}$$

Here s is the mixture weight; a separate schedule maps inference time to s. Increasing s raises the probability of the clean token, while every incorrect bin retains the same probability. The corruption does not distinguish a neighboring action value from a distant one.

The metric-induced path introduces this distinction through a Gibbs kernel:

$$p_s^{\text{metric}}(k \mid y) = \frac{\exp[-\beta(s) d_p(k, y)]}{\sum_{j=0}^{B-1} \exp[-\beta(s) d_p(j, y)]}, \qquad d_p(k, y) = \frac{|k-y|^p}{(B-1)/2}. \tag{17}$$

At $\beta = 0$ the distribution is uniform; as β increases, probability concentrates around the clean bin, approaching a point mass as $\beta \to \infty$. For finite positive β, nearby errors are more probable than distant errors. We use $p = 1$ for absolute bin distance and $p = 2$ for squared bin distance. Uniform scalar bins make index distance proportional to distance in normalized action values; equal-frequency bins instead impose this structure in quantile space.

Figure 13 isolates the difference at matched expected bin error. The mixture path can achieve a low average error by being exactly correct often, while its remaining errors are spread across the vocabulary. The metric path instead allocates more of its error to nearby bins. This

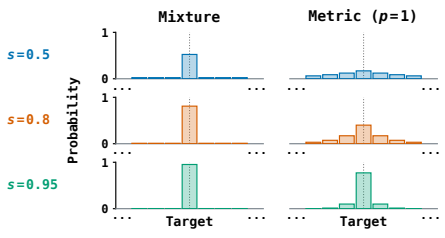


Figure 13: Conditional corruption distributions at matched expected bin error. Colors mark path progress $s = \alpha(t)$; the target is bin 10. Mixture corruption keeps a uniform background, while metric corruption concentrates around the target.

changes the denoising problem even when the average corruption magnitude is the same.

The path and its clock are separate choices. The Gibbs kernel defines a family of distributions; $\beta(s)$ determines how quickly the model moves through that family. Our bin-progress clock is calibrated so that

$$\bar{e}(\beta(s)) = (1-s)\bar{e}(0) + s\bar{e}(\beta_{\max}), \quad \bar{e}(\beta) = \mathbb{E}_{y\sim\pi}\sum_k p_\beta(k \mid y)|k-y|, \tag{18}$$

where the implementation uses a uniform reference histogram π over clean bins. The entropy-progress clock instead linearly decreases the histogram-averaged conditional entropy from $\log B$ to its value at $\beta_{\max}$. Both clocks are numerically inverted and stored as fixed lookup tables. The finite endpoint $\beta_{\max}$ bounds the worst-bin token error using the configured chunk error budget. Thus changing the clock changes the training and inference schedule without changing the form of the Gibbs kernel.

Learning clean tokens and decoding with distance-aware rates. Training retains clean-token cross-entropy, $\mathbb{E}[-\log q_\theta(y \mid X_s, s, o)]$, summed over action-token positions; X_s is the corrupted chunk and o is the observation. Geometry enters through the corruption path. For a current token i and a sampled clean endpoint $\hat{y}$ from the network posterior, the implementation uses kinetic-optimal discrete flow matching (KO-DFM) off-diagonal rates

$$u_s(j \mid i, \hat{y}) = \dot{\beta}(s)\, p_s^{\mathrm{metric}}(j \mid \hat{y})[d_p(i,\hat{y}) - d_p(j,\hat{y})]_+, \qquad j \neq i. \tag{19}$$

Here $[x]_+ = \max(x, 0)$. Conditional on that endpoint, only moves to closer bins have positive rates. The decoder caps each token's rate-vector norm at 50 before the discrete update.

F.2.2 Evaluated Metric-Path Variants

The bin-progress $p = 1$ and entropy-progress $p = 2$ variants are completed training runs with RoboCasa evaluations. Their best pooled success rates are 52.8% and 4.6%, respectively. Applying the bin-progress path to the equal-frequency representation reaches 51.8%. Each rate uses one operating point across all three suites. The first result is competitive with, but does not exceed, the mixture-path DF result of 53.9%. The effect of explicit geometry depends strongly on how the path is constructed.

F.3 Bin Representation and Input-Sensitivity Study

We analyze DR and DF policies trained on XDOF bus-table (plates) and bottle data. These policies use relative stretch actions, a 16-step horizon, 14 action dimensions, a shared 512-bin vocabulary, and 1,536-dimensional bin representations. The checkpoints are DR step 55,000 and DF step 40,000.

Input tables subtract the global mean of all vocabulary rows, while occupied output tables subtract the occupied-row mean before cosine normalization. The analysis restricts the headline statistics to bins with at least 50 aggregate occurrences in the validation sample: 222 bins (145–366). Occupancy was counted over 100,794 valid frames, or 1,411,116 scalar action values; 290 bins had no observed occupancy. The gate is important because unused tails and common components can distort all-vocabulary similarities. Bin separation is measured in the shared normalized coordinate system.

DR/DF centered input adjacency is $-0.0038/0.5733$, and output adjacency is $0.3724/0.3953$. Output embedding-distance versus value-distance Spearman correlations are $0.5835/0.5579$. Bin-relabeling permutation tests give input adjacency $p = 0.81/0.001$ and output adjacency $p = 0.0005$ for both heads. DF input distances saturate at larger separations (reported far-to-near slope ratio 0.027), so the result supports local order. Positive timestep-embedding controls and shuffled-cluster negative controls were also run. Multiple DR checkpoints retain near-zero input adjacency.

Canonical posterior probes use 64 validation frames per head and report 14,252 valid output distributions per head; the input-sensitivity sweep uses eight frames per head, with no classifier-free guidance and effective logit softcap zero. DR total-variation responses remain approximately 0.0125–0.0141 over offsets of 1–256 bins. Replacing its random tokens with true tokens changes decoded

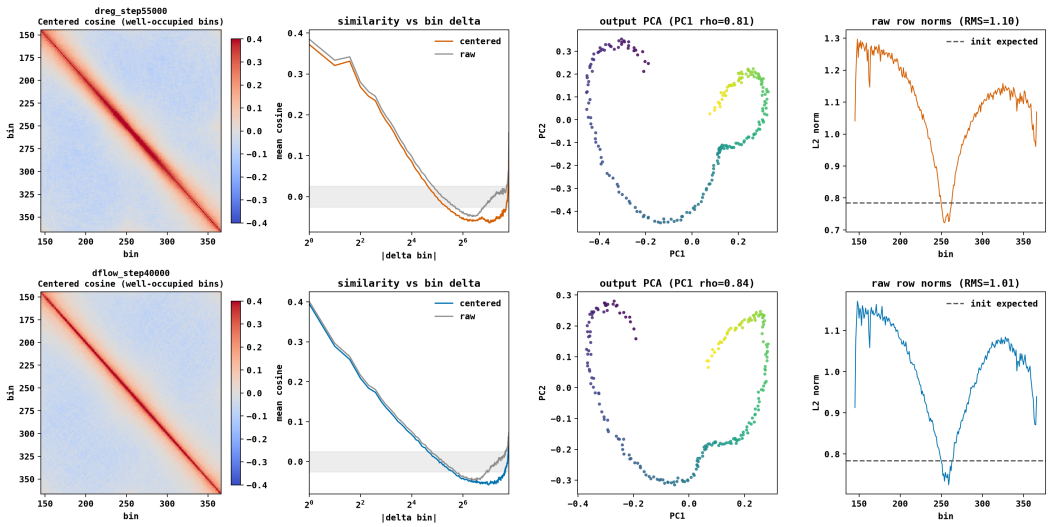


Figure 14: Output-weight diagnostics: DR (top), DF (bottom). Columns show centered cosine, lag-dependent similarity, PCA, and row norms. The banded matrices and decreasing local similarity support ordinal organization in both heads.

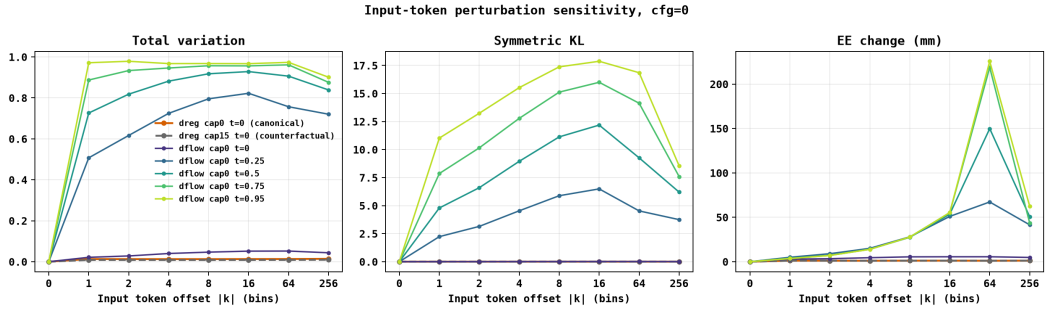


Figure 15: Forward-pass input-offset probes. The plot also shows a counterfactual softcap-15 DR trace; our interpretation uses only the canonical softcap-0 results.

EEF position by 1.18 mm. At DF time 0.75, offsets of 1, 16, and 64 bins change the decoded normalized output by 0.0093, 0.1368, and 0.5121. These interventions establish sensitivity to input tokens.

F.3.1 Combined Output-Representation Visualization

The output PCA projections also organize bins by value, although their first two components explain only about 11% (DR) and 9% (DF) of weight variance. Output biases track the empirical marginal frequency of occupied bins (Pearson $r = 0.809/0.720$ for DR/DF with log-one-plus occupancy).

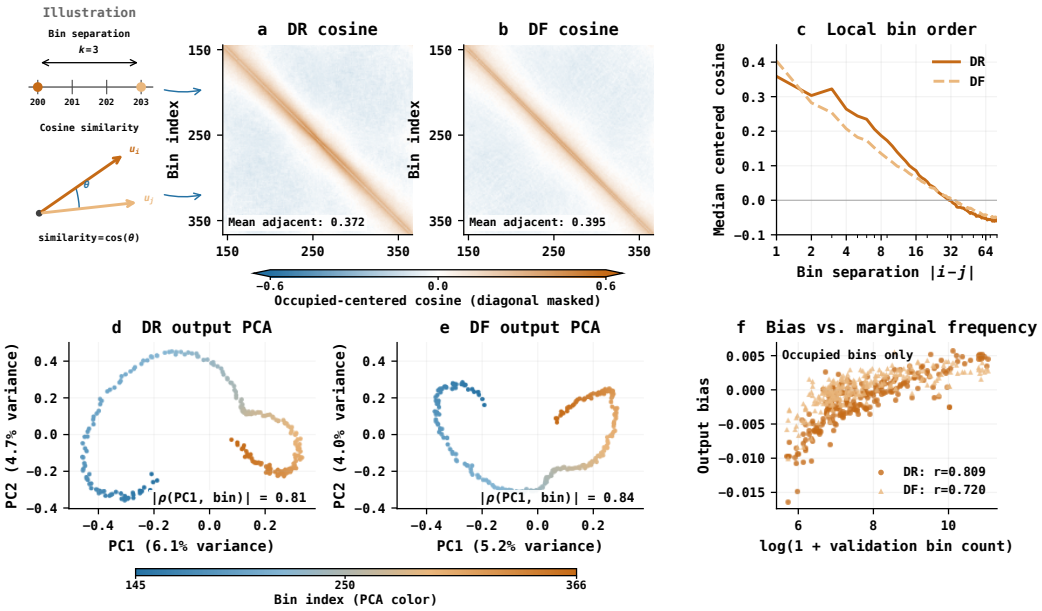


Figure 16: Both discrete heads organize output bins by value. Upper left: generated conceptual illustration defining bin separation and cosine; it contains no empirical measurements. (a,b) Output-weight cosine matrices on the same 222 occupied bins, centered over those rows, with self-similarity masked. (c) Median cosine over pairs at each exact bin separation; only output weights are shown. (d,e) Output-weight PCA, colored by bin index; axes state explained variance, and each head has its own PCA basis. (f) Output bias versus log validation occupancy on the same occupied bins; Pearson $r = 0.809/0.720$ for DR/DF. Zero-count bins are excluded. Scope and controls are in Appendix F.3.

G Synthetic Fitting Protocol and Results

G.1 Which Fine Structure Does Categorical Prediction Preserve?

The robot experiments raise a simple question: can a model get the broad motion right and still miss the fine detail? Robot data mixes too many factors to answer this, so we use synthetic targets where we control the fine structure directly.

Hypothesis 2: categorical prediction can preserve steep fine structure. Following the multi-scale construction of Pan et al. (2025), each target is a broad sine plus a small fine-scale term. In the first target the fine term is a fast sine. In the second it is a square wave, which jumps between two values and so has no finite Lipschitz constant. CF and DF both predict clean targets.

Smooth fine detail: continuous regression wins. On the smooth target (Figure 17A), CR's held-out error is about 250 times lower than DR's. Both continuous heads go below the codec's reconstruction floor. That floor limits any output decoded through argmax, but not continuous outputs. The discrete heads stay well above it, so bin resolution alone does not explain their larger error.

Steep fine detail: the ordering reverses. On the square-wave target (Figure 17B), DR has the lowest held-out error of all heads, with 3.2 times lower MSE and 4.5 times lower MAE than CR. DF also beats both continuous heads, with 31 percent lower error than CF. The prediction panel shows the difference. The regression heads follow the steps but round off the corners, while the categorical heads jump to the next bin right at the edge.

Controls narrow the explanation. Quantized Regression fits the bin centers with squared error. It behaves like CR on both targets, so discrete target values are not what drives DR's result. CF tracks CR and DF trails DR on both targets, so the reversal shows up with one-pass and iterative

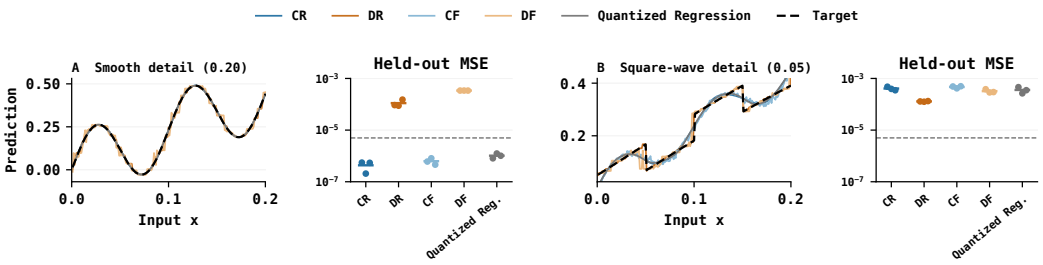


Figure 17: Precision depends on target structure. Each target pairs original predictions (seed 0) with held-out MSE for all three seeds; marks indicate means and the dashed line indicates the hard-bin codec floor. Prediction axes use the local target range; all errors remain included. CF/DF curves show one draw, while stochastic MSE averages eight per-draw errors. Both fine-term amplitude and shape change between targets.

prediction alike. That leaves the categorical formulation as a whole: loss, output parameterization, and argmax decoding. One likely reason is argmax itself. The logits can change smoothly while the decoded bin jumps, which lets the model represent sharp transitions.