BEYOND VALIDATION LOSS: COVERAGE AND RELIABILITY DYNAMICS IN ROBOT POLICY PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

In modern robot learning, generative robot policies are first pretrained through supervised learning from demonstrations, then refined through reinforcement learning, supervised finetuning, or other forms of posttraining. Yet conventional pretraining metrics, such as validation loss and average task success, provide an incomplete account of which policies support both effective direct deployment and subsequent posttraining. In this work, we study closed-loop pass@k, which evaluates the probability that k rollouts from the same initial condition attain at least one success. Thus, pass@1 measures mean task success; large-k pass@k measures whether success remains reachable under repeated sampling. Across a wide range of posttraining recipes, we show that large-k pass@k reveals posttraining differences not captured by pass@1. In particular, checkpoints with similar pass@1 but different large-k pass@k have different posttraining performance. Moreover, we show pass@k peaks at different checkpoints during supervised pretraining, revealing a coverage–reliability tradeoff over the course of training. Based on these observations, we propose two practical interventions: (1) a purely *offline* checkpoint-selection criterion that balances coverage and reliability without closed-loop evaluation, and (2) a generative data augmentation procedure that expands supervision beyond the finite demonstration set, preserving large-k pass@k while maintaining or improving pass@1 and raising the observed coverage–reliability Pareto frontier. Together, our results motivate evaluation, checkpoint selection, and training procedures that explicitly account for posttraining.

1 INTRODUCTION

Robot policies are increasingly trained in stages: supervised behavior cloning (BC) or large-scale pretraining (Chi et al., 2025; Kim et al., 2024; Black et al., 2024), followed by reinforcement learning (RL), supervised finetuning (SFT), or other forms of adaptation. Yet supervised pretraining and posttraining are usually studied as separate problems. The checkpoint handed to posttraining is typically chosen by validation loss, average task success, or simply taking the final checkpoint — criteria that evaluate the supervised pretraining stage in isolation. This leaves a central question unresolved: *how do the learning dynamics of supervised pretraining shape both direct deployment and downstream finetunability, and what criterion can identify checkpoints that jointly support both?*

Language-model research provides strong motivation to look beyond metrics and progress within a single stage of training in isolation. In particular, models with similar pretraining validation loss can perform very differently after posttraining, while prolonged pretraining or stronger SFT performance does not always carry over to subsequent posttraining (Liu et al., 2023; Springer et al., 2025; Kang et al., 2026; Zhang et al., 2026). Recent work further highlights coverage of high-quality outputs as an important property for downstream posttraining (Foster et al., 2025; Chen et al., 2026a).

Robot control, however, presents a different set of challenges than language modeling: policies are densely conditioned on observations, actions are continuous and have far less distributional diversity (Pan et al., 2026), and success requires stable and precise closed-loop execution (Simchowitz et al., 2025; Zhang et al., 2025). These demands may appear to favor increasingly reliable policies over broader behavioral coverage.

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.

Contributions. Despite these differences, we find that supervised training of generative robot policies is governed by a similar tradeoff between broad behavioral coverage and reliable task execution. We study this tradeoff through *closed-loop pass@k from a fixed initial condition* (henceforth, fixed-start pass@k). For each initial configuration, we rollout the stochastic policy k times and ask whether at least one rollout succeeds, then average over the distribution of initial configurations (Sec. 2). Equivalently, pass@k is the expected probability, over the distribution of initial configurations, that at least one of k independently sampled rollouts from the same configuration succeeds. At $k = 1$, this recovers the standard mean task-success rate; we refer to this single-rollout success as *reliability*. As k increases, pass@k measures the fraction of initial conditions for which the policy can produce at least one successful rollout within k attempts; we refer to this as *coverage*. The full pass@k profile therefore lets us study how reliability and coverage evolve during pretraining, and how these differences affect posttraining.

We establish two key findings (Sec. 3). **First**, large-k pass@k tracks downstream posttraining outcomes that pass@1 alone doesn't: checkpoints with similar pass@1 but different large-k pass@k can yield substantially different performance after subsequent SFT or RL finetuning. **Second**, reliability and coverage diverge systematically during supervised pretraining. Large-k pass@k peaks earlier and can decline while pass@1 continues to improve, revealing that further optimization can improve reliability at the expense of coverage. This tradeoff also matters for direct deployment: declining large-k pass@k reflects the loss of low-probability successful behaviors on harder initial conditions, even as average task success improves.

Motivated by these findings, we propose a novel **offline criterion** for selecting policy checkpoints that best balance coverage and reliability (Sec. 4). Estimating pass@k requires many closed-loop rollouts from matched initial configurations, which is prohibitively expensive on real robots. Instead, our criterion is computed entirely offline from the quantile profile of held-out action error. This profile captures favorable-tail coverage and typical-case reliability, allowing us to select the checkpoint that best balances the two. Crucially, it requires no closed-loop policy evaluation, no deployment-specific choice of k, and no tuned tradeoff weight, making it easy to integrate into modern robot policy training pipelines.

Lastly, we introduce a **generative data-augmentation procedure** that raises the Pareto frontier between mean task success and large-k pass@k performance (Sec. 5). We train a joint observation–action generative model to expand supervision beyond the finite demonstration dataset. The resulting augmentation preserves favorable-tail coverage as pretraining proceeds while maintaining, and in some cases improving, reliability as well. By preserving coverage without sacrificing mean task success, it reduces the need for checkpoint selection and provides stronger initializations for subsequent posttraining.

2 Problem Setup

Generative robot policies. We study conditional generative policies $\pi_\theta(a \mid o)$ over action chunks, focusing on diffusion-based policies. Under the noise-prediction parameterization, an expert action chunk a_0 is corrupted at noise level t, and the denoiser is trained as

$$a_t = \alpha_t a_0 + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \qquad \mathcal{L}_{\text{DP}}(\theta) = \mathbb{E}_{(o,a_0),t,\epsilon}\left[\|\epsilon - \epsilon_\theta(a_t, o, t)\|_2^2\right]. \tag{2.1}$$

At inference, $a_T \sim \mathcal{N}(0, I)$ is iteratively denoised using ϵ_θ to produce an action chunk.

Pretraining, posttraining, and checkpoint selection. We consider robot policies trained through an initial supervised pretraining phase followed by a subsequent posttraining phase. Along pretraining, optimization produces a sequence of checkpoints π_τ, where τ indexes training progress, measured in epochs or optimization steps depending on the training setting.[1] The selected checkpoint may then be deployed directly or used to initialize posttraining. In standard practice, checkpoints are commonly selected using validation loss, held-out action MSE, task success rate from closed-loop evaluation, or simply by taking the final checkpoint. Our central question is how the choice of pretraining checkpoint affects performance under direct deployment and after subsequent posttraining, and how to identify checkpoints that support both.

[1] Throughout the paper, τ indexes training progress and should not be confused with the common use of τ for robot trajectories.

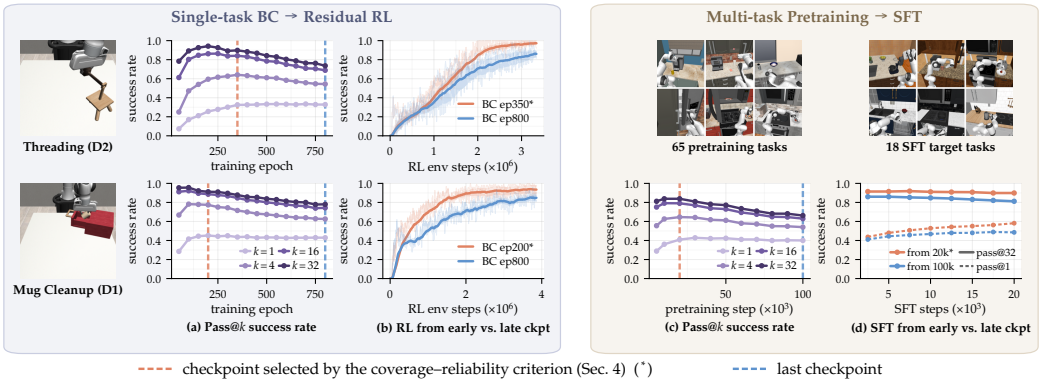


Figure 1: **Later checkpoints trade large-k pass@k for mean task success; checkpoints with similar pass@1 can lead to different posttraining performance. Left:** single-task BC followed by residual RL on two representative MimicGen tasks, with **(a)** showing pass@k during BC and **(b)** residual RL from different BC checkpoints. **Right:** multitask pretraining on RoboCasa tasks followed by SFT on target tasks, with **(c)** showing pass@k during pretraining and **(d)** downstream SFT. Across **(a,c)**, larger-k pass@k peaks and declines earlier than pass@1, so later checkpoints increasingly favor mean task success at the expense of large-k pass@k performance. Orange marks the earlier checkpoint selected by our *offline coverage–reliability criterion* (Sec. 4), while blue marks the final supervised-training checkpoint; matching dashed lines in **(a,c)** indicate the initializations used in **(b,d)**. Despite similar pass@1, finetuning from the orange checkpoint achieves stronger performance than finetuning from the blue checkpoint across both settings.

Fixed-start pass@k. In large language model evaluation on verifiable tasks such as code generation and mathematical reasoning, pass@k is the probability that at least one of k independently sampled solutions is correct (Chen et al., 2021). For robot policies, we define the analogous closed-loop measure. For an initial condition $c \sim p_{\text{eval}}$, let $q_\tau(c)$ denote the one-rollout success probability of policy checkpoint π_τ. Then, we define the pass@k for checkpoint τ as

$$P_k(\tau) = \mathbb{E}_{c \sim p_{\text{eval}}} \left[1 - (1 - q_\tau(c))^k \right]. \tag{2.2}$$

That is, for each held-out initial condition, we reset the simulator and execute k independent rollouts from checkpoint π_τ; pass@k is the fraction of conditions for which at least one rollout succeeds. Pass@1 recovers the standard mean task-success rate.[2]

3 Later Checkpoints Trade Large-k Pass@k for Mean Task Success; Pass@1 Can Miss Posttraining Differences

We begin by examining how pass@k curves across k evolve over supervised pretraining, and whether posttraining from checkpoints with similar pass@1 but different large-k pass@k leads to different performance.

Pretraining evaluation. We study supervised-pretraining dynamics in two settings: (1) single-task BC on MimicGen, and (2) multitask VLA pretraining on RoboCasa. For single-task BC, we train image-conditioned Diffusion Policies (Chi et al., 2025) from scratch on 200 demonstrations for each of five MimicGen tasks (Mandlekar et al., 2023). For multitask VLA pretraining, we use the OpenPI implementation of $\pi_{0.5}$ (Intelligence et al., 2025b) on 65 atomic RoboCasa365 tasks (Nasiriany et al., 2026), with 100 demonstrations per task. Full evaluation details are provided in Appendix H.

For readability, Figure 1(a,c) shows only two of the five selected MimicGen tasks together with the aggregate RoboCasa results. Main-text MimicGen experiments use 200 demonstrations per task. Later experiments (Sec. 5) show all five selected MimicGen tasks and the full RoboCasa evaluation.

[2]We use pass@k as a diagnostic in resettable simulation to study the learning dynamics of pretraining and their relation to finetunability. It is not intended to imply that deployment should require resetting to the same initial condition and executing multiple independent attempts.

During supervised pretraining, pass@k peaks earlier as k increases; mean task success favors later checkpoints. Figure 1(a,c) shows a consistent pattern across single-task BC and multitask pretraining: the pass@k optimum shifts *earlier* as k increases. Thus, later checkpoints can achieve higher mean task success even after large-k pass@k has begun to decline. This divergence between mean task success and large-k pass@k reflects *a redistribution of success across conditions* during supervised pretraining. Recall that $q_\tau(c)$ is the one-rollout success probability from condition c. Pass@1 depends only on its mean; for $k > 1$, we can write $P_k(\tau) = \mathbb{E}_c[f_k(q_\tau(c))]$, where $f_k(q) := 1 - (1-q)^k$ is concave in its argument q. Shifting success probability from harder conditions toward already-reliable ones can therefore increase average success while decreasing larger-k pass@k. As supervised pretraining proceeds, condition-wise success can become increasingly *polarized*: already-solvable conditions become more reliable while harder conditions lose success probability. At the behavioral level, we interpret this redistribution as a tradeoff between *reliability* and *coverage*. See Appendix B for corresponding condition-level decomposition.

Takeaway 1: Over supervised pretraining of robot policies, success can become increasingly *polarized* across conditions: mean task success can continue to improve or stabilize as large-k pass@k declines.

Posttraining evaluation. We show that checkpoints with different pass@k profiles can attain substantially different posttraining performance. For each experiment, we compare the earlier checkpoint selected by our offline coverage–reliability criterion (Sec. 4), called the **selected** checkpoint, with the **final** checkpoint from the same supervised pretraining run. The two have similar pass@1, but the selected checkpoint retains substantially higher large-k pass@k.

We evaluate four posttraining settings. (1) *Filtered BC*: we collect the same number of rollouts from both checkpoint, retain successful episodes, append them to the original training set, and train a new BC policy. (2) *steering with a learned critic*: we train one Q-function per checkpoint on the original training data augmented with unfiltered rollouts from that checkpoint. At held-out state s, both critics score the same eight candidate action chunks $\{a_j\}$. We estimate each candidate's value p_j by branching 16 simulator rollouts from (s, a_j) and measuring the fraction that succeed, then evaluate whether $Q(s, a_j)$ correctly ranks candidates by p_j. (3) *actor-critic residual RL* with DICE-RL (Sun & Song, 2026), an online RL method for finetuning diffusion policies. (4) lastly, on RoboCasa365 only, we perform *supervised finetuning (SFT)* from different pretrained checkpoints using identical target-task finetuning data and optimization. Full experimental details are in Appendix H.

Together, these experiments provide complementary tests of how pretrained-policy coverage affects posttraining. In *filtered BC*, new supervision comes from successful rollouts (Mirchandani et al., 2024); under a fixed rollout budget, lower coverage can yield a similar number of successes from fewer conditions, reducing data diversity. In *critic steering*, broader coverage exposes the critic to a wider range of observation–action outcomes, which can improve generalization when ranking actions at unseen observation; such ranking is central to value-guided test-time steering (Nakamoto et al., 2024) and actor–critic RL. *Actor-critic residual RL* tests whether these differences persist when data collection, critic learning, and policy optimization occur together online. Lastly, *SFT* verifies the findings persist when target-task data are fixed rather than policy-generated rollouts.

Task	Selected	Final
Threading (D2)	**0.371 / 0.886**	0.321 / 0.812
Mug Cleanup (D1)	**0.558 / 0.914**	0.502 / 0.866

Table 1: **Filtered BC.** Pass@1/pass@32 after retraining on successful rollouts generated by the selected or final checkpoint.

Task	Selected	Final
Threading (D2)	**0.534**	0.521
Mug Cleanup (D1)	**0.589**	0.547

Table 2: **Critic action ranking accuracy.** Within-state pairwise accuracy against simulator-estimated action values.

Checkpoints with large-k pass@k enjoy stronger posttraining performance despite similar pass@1 performance. Tab. 1 and 2 isolate this effect under *online* data collection. Across both tasks, retraining on successful rollouts from the selected (higher large-k pass@k) checkpoint produces a stronger filtered BC policy. Indeed, for matched rollout budgets, a greater chance at success on wider variety of initial conditions is more useful than repeated successes on already-solved con-

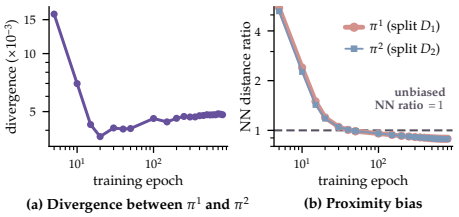
ditions. Similarly, critic trained on the selected-checkpoint rollouts also rank counterfactual actions more accurately at held-out states, indicating better value-function generalization.

Fig. 1(b,d) shows that the difference affects actor-critic finetuning. Under the same interaction budget, residual RL from the selected, higher-large-k checkpoint reaches higher task success than from the final checkpoint. Lastly, on RoboCasa, SFT from the selected checkpoint is also stronger under identical finetuning data and optimization, showing that the checkpoint effect is not limited to methods trained on policy-generated rollouts.

We note that large-k pass@k does not globally rank checkpoints for finetuning: for much earlier checkpoints with lower pass@1, which checkpoint attains higher RL success can depend on finetuning budget (Appendix C). Among checkpoints with comparable pass@1, however, large-k pass@k reveals posttraining differences not captured by pass@1 and conventional offline metrics (Sec.4).

Takeaway 2: Coverage–reliability dynamics during supervised pretraining affect posttraining: checkpoints with similar pass@1 can yield different posttraining performance. Our coverage–reliability criterion selects checkpoints that outperform later checkpoints favored by mean task success and held-out action MSE (Appendix E).

Loss of pass@k performance during training corresponds with overfitting on the empirical dataset, and concentrates on low noise levels. We next examine the learning dynamics behind the late-checkpoint loss of coverage. We find two effects: (1) pretraining shifts from population-level generalization toward overfitting the finite training set, and (2) overfitting increases toward the low-noise end of diffusion and as observation support becomes sparser.



(a) Divergence between π^1 and π^2 (b) Proximity bias

Figure 2: **Independent splits expose dataset-specific fitting.** (a) Mean-squared prediction difference between two policies trained on disjoint splits, evaluated on the same held-out inputs with matched diffusion noise; predictions first converge, then diverge. (b) Ratio between the average nearest-neighbor distance from policy samples to their own training split and the average nearest-neighbor distance from held-out expert actions to that same training split; decreasing values indicate increasing bias toward the training split.

We first test the shift toward training-set overfitting. On `Threading D2`, we split 1,000 demonstrations into two disjoint 200-demo training sets, D_1 and D_2 , and a shared 600-demo held-out set, D_3. We train separate $\pi_\tau^{(1)}$ and $\pi_\tau^{(2)}$ on D_1 and D_2, respectively. Let $\bar{a}_\tau^{(i)}(o, z)$ denote the action chunk generated by $\pi_\tau^{(i)}$ from observation o and initial noise z. Fig. 2(a) measures prediction disagreement $D_{\text{split}}(\tau) = \mathbb{E}_{o\sim D_3, z}[\|\bar{a}_\tau^{(1)}(o, z) - \bar{a}_\tau^{(2)}(o, z)\|_2^2]$. Lower D_{split} means the two policies agree more on the same held-out inputs. This quantity first decreases during training as learning converges, then increases, indicating the learned policies become increasingly overfit to the D_1/D_2 data split. This is corroborated by Fig. 2(b), which measures the ratio R_i between nearest neighbor distances from actions $\bar{a}_\tau^{(i)} \sim \pi_\tau^{(i)}$ and D_i, divided by those from $a^\star \sim D_3$ (see Appendix D for details). Here, $R_i < 1$ means policy samples are closer to their own training split than held-out expert actions are. As pretraining proceeds, R_i decreases for both policies. Together, the two measurements show a transition from shared population-level structure toward split-specific fitting.

These split-data results are consistent with prior work showing that diffusion models shift from generalization toward training-set fitting and memorization under prolonged optimization (Gu et al., 2023; Baptista et al., 2025; Ye et al., 2026; Bertrand et al., 2026; Bonnaire et al., 2026; Favero et al., 2025). In our image-conditioned robot policies, appendix D further shows this overfitting is concentrated at low diffusion noise, and also becomes stronger where observation support is sparse. The intuition is that as the noise level decreases, the set of clean actions compatible with a noisy action shrinks; when observation support is sparse, fewer demonstrations also have nearby observations, leaving fewer training examples compatible with both the observation and the noisy action.

Takeaway 3: As supervised pretraining proceeds, diffusion policies shift from population-level generalization toward finite training-set overfitting, which is strongest at low noise end and under sparse observation support.

4 Diagnosing the Coverage–Reliability Tradeoff and Selecting Checkpoints Offline

Having characterized the transition toward finite-data overfitting, we next ask how to identify the coverage–reliability balance before overfitting, without explicitly evaluating pass@k performance through closed-loop policy deployment.

Best-of-k action MSE captures the favorable tail but selects too early. Because diffusion denoising loss is often a poor indicator of closed-loop performance (Allshire et al., 2026), we use held-out action MSE for offline evaluation throughout the paper (Allshire et al., 2026). A tempting offline metric for approximating closed-loop pass@k is then held-out best-of-k action MSE. Given a held-out dataset $\mathcal{D}_{\text{val}}$, we sample k action chunks at each observation and retain the one closest to the expert action:

$$B_k(\tau) = \mathbb{E}_{(o,a^\star)\sim\mathcal{D}_{\text{val}}}\left[\min_{j\le k} \ell(A_{\tau,j}(o), a^\star)\right]. \tag{4.1}$$

Fig. 3 shows that larger k shifts the optimum of B_k earlier, as it does for pass@k. For small k, the two optima remain relatively close; as k increases, the best-of-k minimum increasingly precedes the corresponding closed-loop pass@k peak. This mismatch arises because B_k measures only the favorable tail independently at each observation, whereas closed loop control requires reliable actions throughout an entire rollout trajectory. An early checkpoint can therefore attain low large-k-B_k while still assigning substantial probability to poor action samples that destabilize execution. This motivates us to move beyond the minimum over samples and examine the full held-out action MSE distribution.

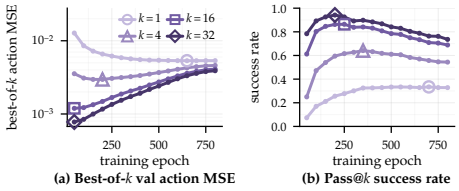


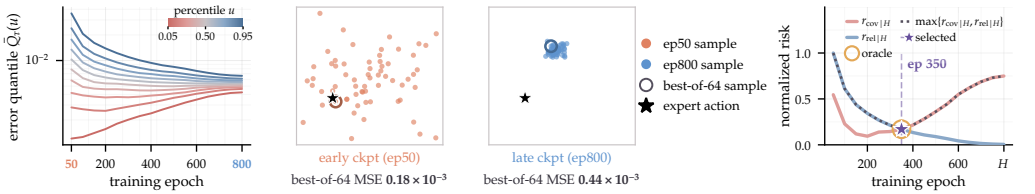
Figure 3: **Best-of-k action MSE selects earlier checkpoints than closed-loop pass@k.** (a) Held-out best-of-k action MSE across training checkpoints for different k. (b) Closed-loop pass@k success rate across the same checkpoints. Matching colors and marker shapes denote the same value of k across panels; hollow markers indicate the minimum MSE in (a) and maximum pass@k in (b). As k increases, both optima shift earlier, but the best-of-k MSE minimum precedes the corresponding pass@k peak, with a larger separation at higher k.

Held-out action-MSE quantiles better reflect coverage and reliability. Best-of-k MSE uses only the minimum error among sampled actions. Here, we now examine the full distribution of action MSE under policy samples. For held-out pair $(o_i, a_i^\star)$ and checkpoint τ, draw $A_j \sim \pi_\tau(\cdot \mid o_i)$, $j = 1, \ldots, M$, and let $Q_{i,\tau}(u)$ be the u-quantile of $\{\ell(A_j, a_i^\star)\}_{j=1}^M$. We average each quantile across held-out observations,

$$\bar{Q}_\tau(u) = \frac{1}{N}\sum_{i=1}^{N} Q_{i,\tau}(u). \tag{4.2}$$

Fig. 4(a) shows a more systematic ordering: lower quantiles reach their minima earlier and then worsen, whereas higher quantiles continue improving later in supervised pretraining. Fig. 4(b) illustrates this directly: the early checkpoint can produce lucky draws closer to the expert action, but its samples are substantially more *dispersed*; the late checkpoint produces more *concentrated* samples, but its best sample is farther from the expert.

A distributional criterion identifies the coverage–reliability balance. Given the previous findings, we propose to use lower quantiles as an offline measure of *coverage*, i.e. whether low-MSE action chunks remain accessible under the policy, and higher quantiles as an measure of *reliability*, how consistently sampled action chunks remain close to the demonstrated action. Based on this, we define the following normalized quantile score for training budget H:



(a) Validation action-MSE quantiles **(b) Action samples at a representative held-out state** **(c) Coverage–Reliability Balance**

Figure 4: **Held-out action-MSE quantiles expose the coverage–reliability balance.** (a) Held-out action-MSE quantiles across training, colored by percentile u from 0.05 (red) to 0.95 (blue). (b) For a held-out observation, 64 matched-noise samples from ep50 and ep800, plus the expert action, are projected with PCA; rings mark the minimum-MSE sample. ep50 is more dispersed yet has lower best-of-64 MSE, whereas ep800 contracts but its best sample lies farther from the expert. Consistent with (a), lower quantiles favor earlier checkpoints, while higher quantiles improve later. (c) We summarize the lower and upper halves as $r_{\mathrm{cov}|H}(\tau)$ and $r_{\mathrm{rel}|H}(\tau)$ and select the checkpoint minimizing $\max\{r_{\mathrm{cov}|H}(\tau), r_{\mathrm{rel}|H}(\tau)\}$; the selected ep350 matches the closed-loop oracle.

$$r_{\tau|H}(u) = \frac{\log \bar{Q}_\tau(u) - \min_{s \le H} \log \bar{Q}_s(u)}{\max_{\tau' \le H} \log \bar{Q}_{\tau'}(u) - \min_{s \le H} \log \bar{Q}_s(u)} \tag{4.3}$$

Here, $r_{\tau|H}(u) = 0$ means checkpoint τ has the lowest observed MSE at percentile u among checkpoints up to H, while $r_{\tau|H}(u) = 1$ means the highest; intermediate values indicate its normalized position between these two extremes. Let $\mathcal{U}$ denote the discrete set of evaluated quantile levels. To evaluate policy coverage and reliability, we average lower and upper halves of the quantile profile as

$$r_{\mathrm{cov}|H}(\tau) = \mathrm{mean}_{u \in \mathcal{U},\, u < 0.5}\, r_{\tau|H}(u), \qquad r_{\mathrm{rel}|H}(\tau) = \mathrm{mean}_{u \in \mathcal{U},\, u \ge 0.5}\, r_{\tau|H}(u). \tag{4.4}$$

using $u = 0.5$ to split the profile evenly without additional tuned threshold. We then select

$$\tau^\star_H = \arg\min_{\tau \le H} \max \left\{ r_{\mathrm{cov}|H}(\tau), r_{\mathrm{rel}|H}(\tau) \right\}. \tag{4.5}$$

The criterion therefore selects the checkpoint that minimizes the larger of $r_{\mathrm{cov}|H}$ and $r_{\mathrm{rel}|H}$. Unlike best-of-k, which uses only the minimum, it accounts for both lower and upper quantiles; unlike average validation MSE, it does not collapse the error distribution to a single average. By minimizing the worse of the two scores, the criterion avoids selecting a checkpoint with strong coverage but poor reliability, or vice versa. Lastly, because the criterion is defined relative to the quantile values, it also does not require task-specific calibration over absolute action-MSE thresholds.

Fig. 4(c) illustrates the criterion on the 200-demonstration `Threading D2` run. Using only offline held-out action MSE, the criterion recovers the closed-loop oracle at epoch 350. We define the closed-loop oracle by applying the same minimax balancing principle across the full pass@k profile. The criterion is also robust to the choice of training horizon once the coverage–reliability dynamics has stabilized: after lower quantiles begin to worsen and higher quantiles saturate, the selected checkpoint remains relatively stable as training continues. Extending training substantially beyond this point only marginally shifts the selected checkpoint (Appendix E). Appendix E further validates the criterion across MimicGen tasks and dataset sizes, as well as multitask pretraining on RoboCasa.

Takeaway 4: Best-of-k action MSE uses only the minimum-error sample and selects checkpoints too early, especially at large k. Across the full held-out action-MSE distribution, lower quantiles favor earlier checkpoints while higher quantiles improve later, enabling an offline criterion that balances coverage and reliability without closed-loop evaluation.

5 Generative Augmentation Preserves Coverage for Downstream Posttraining

Although our proposed coverage–reliability criterion (Sec. 4) selects a favorable compromise without closed-loop evaluation, it is constrained to the observed coverage–reliability Pareto frontier of that run. This raises a more fundamental question: is this tradeoff unavoidable? Can a training

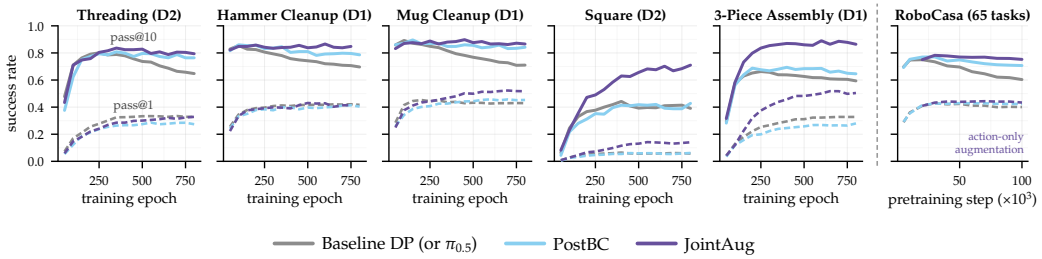


Figure 5: **During pretraining, generative augmentation preserves coverage while maintaining, and in some tasks improving, reliability (pass@1).** Across most MimicGen tasks and RoboCasa, `Baseline DP` loses pass@10 at later checkpoints. `JointAug` prevents this loss of coverage while maintaining or improving pass@1. `PostBC` also preserves pass@10 better than `Baseline DP`, but can do so at the cost of pass@1.

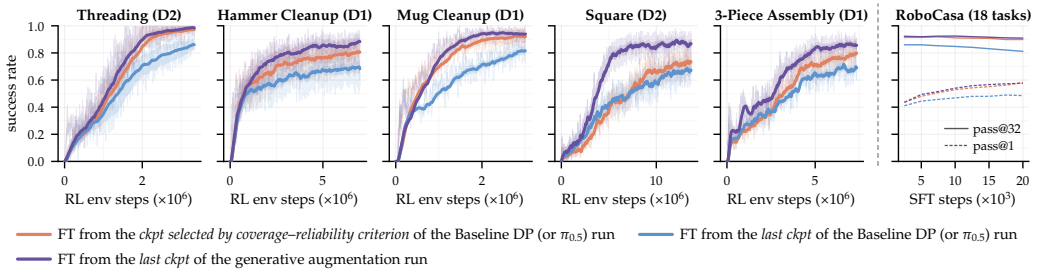


Figure 6: **Generative augmentation produces stronger checkpoints for posttraining.** Columns correspond to Fig. 5. We compare posttraining from the baseline checkpoint selected by our coverage–reliability criterion (Sec. 4), the final `Baseline DP` checkpoint, and the final checkpoint from our generative augmentation run. Across MimicGen tasks, the final generative-augmentation checkpoint substantially outperforms the final baseline and often matches or exceeds the selected checkpoint. On RoboCasa, it matches the selected checkpoint on both pass@1 and pass@32.

intervention give us the best of both worlds by preserving strong large-k pass@k while continuing to improve reliability? If so, we can raise the coverage–reliability Pareto frontier.

Sec. 3 links the loss of coverage to overfitting to the finite demonstration data during policy training. This suggests a direct intervention: expand supervision beyond the observed (o, a) pairs using fresh synthetic observation–action pairs that reflect the population-level expert distribution.

Joint (o, a) augmentation. To expand supervision beyond observed (o, a) pairs while preserving observation–action consistency, we propose *joint (o, a)-augmentation* (`JointAug`), which trains a generative model $G_\gamma(o, a)$ with checkpoints indexed by γ on the joint demonstration distribution and uses generated pairs for policy training. We save generator checkpoints at fixed intervals, and aim to identify a sampling window after the generator has learned population-level structure but before substantial overfitting. To this we end record each checkpoint's ratio between generated actions' nearest-neighbor distances to training expert actions and held-out expert actions (Appendix H.3). A ratio of 1 indicates equal proximity to the two sets, and values below one indicate that generated samples are closer to the training data. We let γ_0 be the first checkpoint where the ratio falls below one, when generated samples first become closer to training than held-out expert actions. We then sample synthetic $(\tilde{o}, \tilde{a})$ pairs uniformly from γ_0 and the four subsequent checkpoints spaced 100 epochs apart. Because checkpoints are saved relatively densely over this stage of generator training, the five-checkpoint window spans only the transition from population-level generalization toward early fitting of the empirical training set. We use five checkpoints as a simple default heuristic across tasks rather than tuning the window for each task. Appendix F.2 shows that performance is robust to the exact generator checkpoint selection.

Then policy training uses equal amounts of real and generated supervision:

$$\mathcal{L}_{\text{aug}} = \tfrac{1}{2}\,\mathbb{E}_{(o,a)\sim\mathcal{D}}[\mathcal{L}_{\text{DP}}(o, a)] + \tfrac{1}{2}\,\mathbb{E}_{(\tilde{o},\tilde{a})\sim G_{\text{mix}}}[\mathcal{L}_{\text{DP}}(\tilde{o}, \tilde{a})], \tag{5.1}$$

where G_{mix} is the uniform mixture of the five selected generator checkpoints. Real samples retain the clean expert targets; generated pairs expand supervision beyond the finite data. Because observations and actions are generated jointly, their correspondence is naturally preserved.

To evaluate **JointAug**, we compare against baseline Diffusion Policy and posterior behavior cloning (**PostBC**, Wagenmaker et al. (2025a)) across the full training horizon. **PostBC** aims to improve policy coverage by estimating observation-conditioned action covariance with a bootstrapped ensemble and perturbing action targets $\tilde{a}$ at clean observations o. In that regime, the perturbed action is not constrained to remain compatible with the clean observation. In contrast, **JointAug** ensures $(\tilde{o}, \tilde{a})$ is learned from the correct joint distribution (and thus, $\tilde{a} \mid \tilde{o}$ is consistent).

In the first five panels of Fig. 5, we compare the methods across MimicGen tasks. **JointAug** achieves higher pass@10 than both **PostBC** and the baseline DP run and prevents the baseline's late-training loss of coverage. Its pass@1 matches the baseline and sometimes improves it. **PostBC** also improves pass@10, but on some tasks lowers pass@1, likely because its perturbed actions can be incompatible with the clean observations. Appendix F.3 analyzes the sampled action distributions. **JointAug** preserves the favorable low-error tail and reduces typical-sample error, helping explain its gains in both coverage and reliability. **JointAug** also provides checkpoints with stronger residual-RL initializations than the final baseline checkpoints (Fig. 6), often matching or exceeding the checkpoint selected by our coverage–reliability criterion from baseline run. By preventing the late-training loss of coverage, **JointAug** also reduces the need for post hoc checkpoint selection. We further find that its performance is robust to the choice of generator checkpoint; more ablations and baseline comparisons are in Appendix F.

A scalable action-only variant. For large-scale VLA pretraining, joint generation is expensive, so on RoboCasa we use a lightweight action-only variant, similar in spirit to **PostBC** but using the policy's own action variation rather than a bootstrapped ensemble for covariance estimation. After a standard pretraining run, we select $c^\star$ using our coverage–reliability criterion and estimate action covariance $\Sigma_{c^\star}(o)$ from multiple policy samples at each training observation. We then resume pretraining from $c^\star$ with perturbed targets $\tilde{a} = a + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \Sigma_{c^\star}(o))$. This retains the action spread at $c^\star$ while keeping the mean unchanged, $\mathbb{E}[\tilde{a} \mid o] = a$. Fig. 5 and 6 show that the variant preserves large-k pass@k as pass@1 improves, and yields SFT performance comparable to the selected early checkpoint.

Takeaway 5: Generative augmentation can preserve coverage while allowing reliability to improve, producing late-pretraining checkpoints with stronger pass@k profiles and better downstream posttraining than standard final checkpoints.

6 Abridged Related Work

Due to space constraints, we discuss key related work here and defer more comprehensive review to Appendix A. In language models, coverage of high-quality outputs has emerged as an important property for efficient posttraining and test-time scaling (Foster et al., 2025; Chen et al., 2026a). Modern generative robot policies (Chi et al., 2025; Kim et al., 2024; Black et al., 2024) are likewise followed by supervised finetuning, RL, or inference-time steering (Kim et al., 2025; Intelligence et al., 2025a; 2026), yet few works study how properties acquired during pretraining shape these later stages. PostBC and TMRL explicitly modify pretraining to improve subsequent RL by preserving action coverage or exploration (Wagenmaker et al., 2025a; Hong et al., 2026). We instead study how coverage and reliability change during standard supervised pretraining itself and how they affect posttraining. Mirroring work showing that diffusion models can transition from generalization toward finite-sample fitting over training (Bonnaire et al., 2026; Favero et al., 2025), we characterize this learning dynamic in robot policies and connect it to closed-loop coverage and posttraining.

7 Conclusion

Our work studies how the coverage–reliability profile learned during supervised robot-policy pretraining shapes posttraining performance, and provides practical methods to measure, select, and improve this profile. Complementing our focus on preserving useful coverage during pretraining, an important open question is what posttraining data best enables efficient policy improvement. Future work could develop better pretraining objectives and optimization strategies that preserve both coverage and reliability, and build a stronger theoretical understanding of the interplay between robot policy pretraining and posttraining. It will also be important to test whether these dynamics persist in broader robot datasets, real-world settings, and a wider range of posttraining methods.

AI USE STATEMENT

We used generative AI tools to polish the writing, search for relevant literature, and help with parts of the code implementation. We did not use them to generate data, derive mathematical results, develop research ideas, or interpret experimental results. The authors checked all AI-assisted work: we tested the code, verified every cited paper against its source, and confirmed the text accurately reflects our contributions. We take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Appendix H describes implementation in detail, including the simulation environments and datasets, the implementation of our proposed method and all baselines), network architectures, hyperparameters, and the evaluation protocol. We will release the code, model checkpoints, and datasets used in this work in the very near future to support reproducibility and future research.

REFERENCES

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, et al. Scalable behavior cloning with open data, training, and evaluation. *arXiv preprint arXiv:2606.27375*, 2026.

Ricardo Baptista, Agnimitra Dasgupta, Nikola B Kovachki, Assad Oberai, and Andrew M Stuart. Memorization and regularization in generative diffusion models. *arXiv preprint arXiv:2501.15785*, 2025.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11 (113):eaea6201, 2026.

Quentin Bertrand, Anne Gagneux, Mathurin Massias, and Rémi Emonet. On the closed-form of flow matching: Generalization does not arise from target stochasticity. *Advances in neural information processing systems*, 38:8522–8549, 2026.

Giulio Biroli, Tony Bonnaire, Valentin De Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *Nature Communications*, 15(1):9957, 2024.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Adam Block, Ali Jadbabaie, Daniel Pfrommer, Max Simchowitz, and Russ Tedrake. Provable guarantees for generative behavior cloning: Bridging low-level stability and high-level behavior. *Advances in Neural Information Processing Systems*, 36:48534–48547, 2023.

Tony Bonnaire, Raphaël Urfin, Giulio Biroli, and Marc Mézard. Why diffusion models don't memorize: The role of implicit dynamical regularization in training. *Advances in Neural Information Processing Systems*, 38:141266–141286, 2026.

Fan Chen, Audrey Huang, Noah Golowich, Sadhika Malladi, Adam Block, Jordan Ash, Akshay Krishnamurthy, and Dylan Foster. The coverage principle: How pre-training enables post-training. In *International Conference on Learning Representations*, volume 2026, pp. 112225–112293, 2026a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

William Chen, Jagdeep Singh Bhatia, Catherine Glossop, Nikhil Mathihalli, Ria Doshi, Andy Tang, Danny Driess, Karl Pertsch, and Sergey Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control. *arXiv preprint arXiv:2602.13193*, 2026b.

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. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Giannis Daras, Kulin Shah, Yuval Dagan, Aravind Gollakota, Alex Dimakis, and Adam Klivans. Ambient diffusion: Learning clean distributions from corrupted data. *Advances in Neural Information Processing Systems*, 36:288–313, 2023.

Alessandro Favero, Antonio Sclocchi, and Matthieu Wyart. Bigger isn't always memorizing: Early stopping overparameterized diffusion models. *ArXiv*, abs/2505.16959, 2025. URL `https://api.semanticscholar.org/CorpusID:278789454`.

Dylan J Foster, Zakaria Mhammedi, and Dhruv Rohatgi. Is a good foundation necessary for efficient reinforcement learning? the computational role of the base model in exploration. *arXiv preprint arXiv:2503.07453*, 2025.

Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. On memorization in diffusion models. *arXiv preprint arXiv:2310.02664*, 2023.

Chengyang He, Xu Liu, Gadiel Mark Sznaier Camps, Joseph Bruno, Guillaume Adrien Sartoretti, and Mac Schwager. Demystifying robot diffusion policies: Action memorization and a simple lookup table alternative. In *The Fourteenth International Conference on Learning Representations*, 2026.

Matthew M Hong, Jesse Zhang, Anusha Nagabandi, and Abhishek Gupta. Tmrl: Diffusion timestep-modulated pretraining enables exploration for efficient policy finetuning. *arXiv preprint arXiv:2605.12236*, 2026.

Haoxu Huang, Tongsam Zheng, Yifan Chen, Jiacheng You, and Yang Gao. Critical interval mse: Toward reliable offline validation for robot manipulation policies. *arXiv preprint arXiv:2606.29898*, 2026.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, et al. $\pi_{0.7}$: a steerable generalist robotic foundation model with emergent capabilities. *arXiv preprint arXiv:2604.15483*, 2026.

Feiyang Kang, Michael Kuchnik, Karthik Padthe, Marin Vlastelica, Ruoxi Jia, Carole-Jean Wu, and Newsha Ardalani. Quagmires in sft-rl post-training: When high sft scores mislead and what to use instead. In *International Conference on Learning Representations*, volume 2026, pp. 56876–56918, 2026.

Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 6185–6191. IEEE, 2021.

Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. Ccil: Continuity-based data augmentation for corrective imitation learning. In *International Conference on Learning Representations*, volume 2024, pp. 35813–35831, 2024.

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

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

Hadas Kress-Gazit, Kunimatsu Hashimoto, Naveen Kuppuswamy, Paarth Shah, Phoebe Horgan, Gordon Richardson, Siyuan Feng, and Benjamin Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation. *arXiv preprint arXiv:2409.09491*, 2024.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pp. 143–156. PMLR, 2017.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *International Conference on Machine Learning*, pp. 22188–22214. PMLR, 2023.

Sadhika Malladi, Samy Jelassi, Dylan Foster, Jordan T Ash, and Akshay Krishnamurthy. Tailsft: Filtered fine-tuning improves post-training performance. *arXiv preprint arXiv:2608.25756*, 2026.

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. *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. *arXiv preprint arXiv:2310.17596*, 2023.

Suvir Mirchandani, Suneel Belkhale, Joey Hejna, Evelyn Choi, Md Sazzad Islam, and Dorsa Sadigh. So you think you can scale up autonomous robot data collection? *arXiv preprint arXiv:2411.01813*, 2024.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

Chaoyi Pan, Giridharan Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. In *International Conference on Learning Representations*, volume 2026, pp. 90575–90614, 2026.

Kulin Shah, Alkis Kalavasis, Adam R Klivans, and Giannis Daras. Does generation require memorization? creative diffusion models using ambient diffusion. *arXiv preprint arXiv:2502.21278*, 2025.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The pitfalls of imitation learning when actions are continuous. *arXiv preprint arXiv:2503.09722*, 2025.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

David Snyder, Asher James Hancock, Apurva Badithela, Emma Dixon, Patrick Miller, Rares Andrei Ambrus, Anirudha Majumdar, Masha Itkina, and Haruki Nishimura. Is your imitation learning policy better than mine? policy comparison with near-optimal stopping. *arXiv preprint arXiv:2503.10966*, 2025.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Understanding and mitigating copying in diffusion models. *Advances in Neural Information Processing Systems*, 36:47783–47803, 2023.

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. *arXiv preprint arXiv:2503.19206*, 2025.

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive rl finetuning. *arXiv preprint arXiv:2603.10263*, 2026.

Andrew Wagenmaker, Perry Dong, Raymond Tsao, Chelsea Finn, and Sergey Levine. Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning. *arXiv preprint arXiv:2512.16911*, 2025a.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025b.

Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Pérez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15626–15633. IEEE, 2025.

Zeqi Ye, Qijie Zhu, Molei Tao, and Minshuo Chen. Provable separations between memorization and generalization in diffusion models. In *International Conference on Learning Representations*, volume 2026, pp. 143375–143425, 2026.

Dylan Zhang, Yufeng Xu, Haojin Wang, Qingzhi Chen, and Hao Peng. Good sft optimizes for sft, better sft prepares for reinforcement learning. *arXiv preprint arXiv:2602.01058*, 2026.

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

APPENDIX CONTENTS

A Extended Related Work **15**

A.1 Pretraining–posttraining Interactions in Robot Learning 15

A.2 Generalization and Memorization in Diffusion Models 15

A.3 Robot Policy Evaluation and Model Selection 15

A.4 Noise Injection in Robot Policies 16

B Success Polarization Across Evaluation Conditions **17**

C Additional Results on Residual RL Finetuning Across the Coverage–Reliability Trade-off **17**

D Diagnosing Overfitting in Diffusion Policies **18**

E Full Coverage–Reliability Results and Checkpoint Selection **22**

F Additional Results on `JointAug` **24**

F.1 Additional Task 24

F.2 Robustness to Generator Checkpoint Selection 24

F.3 Visualizing How `JointAug` Changes the Action Distribution 25

F.4 Additional Baseline - Context-Smoothed Pre-training in `TMRL` 26

G Additional Results and Visualizations **27**

G.1 Sampling Configuration Ablations 27

G.2 Denoising Network Size and EMA Ablations 27

G.3 Isolating Coverage Loss to the Denoising Network 29

G.4 Best-of-k Off-Manifold Error Reveals the Same Coverage–Reliability Tradeoff . . 31

G.5 How Policy Coverage Affects World-Model Learning 31

H Implementation Details **32**

H.1 Tasks Datasets, and Evaluation 32

H.2 Baseline Single Task Diffusion Policy 32

H.3 `JointAug` Implementation 33

H.4 `PostBC` Implementation 34

H.5 Context-Smoothed Pre-training in `TMRL` Implementation 35

H.6 Residual RL Implementation 35

H.7 Multitask Training on Robocasa 35

A Extended Related Work

Pretraining–posttraining Interactions in Robot Learning. Language-model research provides strong motivation for studying pretraining and posttraining jointly. Recent work shows that performance within one stage in isolation can poorly predict the outcomes after the next. Models with similar pretraining validation loss can differ substantially in thei finetuning performance (Liu et al., 2023), and prolonged pretraining can reduce subsequent finetuning performance (Springer et al., 2025). In SFT–RL pipeline, higher SFT performance does not always predict stronger post-RL performance; stronger SFT checkpoints can even perform worse after the same RL procedure (Kang et al., 2026; Zhang et al., 2026). Recent theory instead highlights coverage, the probability mass assigned to high-quality outputs, as an important property for test-time sampling (Foster et al., 2025; Chen et al., 2026a). Empirically, large-k pass@k can predict post-RL performance better than SFT task performance (Kang et al., 2026). TailSFT similarly improves posttraining by modifying SFT to preserve broader output coverage (Malladi et al., 2026). Robot learning increasingly follows a similar staged pipeline: BC or pretraining (Chi et al., 2025; Kim et al., 2024; Black et al., 2024) followed by SFT (Kim et al., 2025; Bjorck et al., 2025; Nasiriany et al., 2026), RL (Intelligence et al., 2025a; Wagenmaker et al., 2025b), or inference-time steering (Intelligence et al., 2026; Wang et al., 2025; Chen et al., 2026b). Yet few works study how properties learned during pretraining affect posttraining. PostBC proves that standard BC can fail to guarantee coverage of the demonstrator's actions, which is necessary for RL finetuning to reach the demonstrator's performance. It instead models the posterior over demonstrator action to guarantee such coverage and improve RL finetuning (Wagenmaker et al., 2025a). TMRL broadens coverage during pretraining by adding noise to the policy context, then lets RL control the noise level during finetuning (Hong et al., 2026). Both methods redesign pretraining with noise injection specifically for downstream RL finetunig. We instead study how a policy's finetunability changes over the course of standard supervised pretraining. We identify an evolving coverage–reliability tradeoff, show that it affects multiple forms of posttraining, and use this to design offline checkpoint selection criterion and training intervention.

Generalization and Memorization in Diffusion Models. Finite-data diffusion objectives admit closed form solutions that reproduce the training set, and several works characterize how model capacity, dataset size, conditioning, and optimization determine whether trained models approach these memorizing solutions (Gu et al., 2023; Baptista et al., 2025; Ye et al., 2026; Biroli et al., 2024). Recent papers further show a temporal separation in training dynamics: models first learn population-level structure that generalizes beyond the training set, while prolonged optimization can increasingly fit individual training examples (Bonnaire et al., 2026; Favero et al., 2025). Related work on flow matching likewise studies why learned models generalize despite an empirical optimum that memorizes the data (Bertrand et al., 2026). These findings have motivated interventions including early stopping, limiting model capacity (Baptista et al., 2025; Ye et al., 2026; Favero et al., 2025), training from corrupted examples (Daras et al., 2023; Shah et al., 2025), and perturbing conditioning input (Somepalli et al., 2023). Most of this literature studies unconditional, class-conditioned, or text-conditioned generation of independent samples. Robot policies instead condition action generation on dense, high-dimensional observations and repeatedly sample the policy for multi-turn closed-loop interaction. He et al. (2026) adopt a static view of trained diffusion policies and argue that, on small robot datasets, they largely retrieve actions from nearby training examples. We instead study how the policy distribution dynamically evolves over optimization, how this evolution changes closed-loop reliability and coverage and posttraining.

Robot Policy Evaluation and Model Selection. Closed-loop rollouts remain the standard for evaluating robot policies, but reliable comparisons require many trials and careful statistical treatment (Kress-Gazit et al., 2024; Barreiros et al., 2026; Snyder et al., 2025). This cost motivates offline metrics for policy development and checkpoint selection. Mandlekar et al. (2021) showed that validation loss can select substantially worse BC checkpoints than rollout evaluation. More recently, ABC found that diffusion validation loss was poorly correlated with real-world performance, whereas validation action error correlates more strongly with it (Allshire et al., 2026). Other work proposes manifold adherence (Pan et al., 2026) or task-critical action error (Huang et al., 2026) as more informative offline signals. Our setting differs because stochastic generative policies can have similar average error while differing substantially across repeated sampling. We therefore evalu-

ate the full distribution of held-out action errors and select checkpoints by balancing coverage and reliability with a special focus for on their use in downstream posttraining.

Noise Injection in Robot Policies. Noise injection and local data augmentation methods have long been used to reduce covariate shift in behavior cloning. DART injects action noise during expert collection so demonstrations cover states reached after action perturbations (Laskey et al., 2017). Other work perturbs existing demonstrations and synthesizes corrective labels (Ke et al., 2021), uses noise augmentation to improve closed-loop stability (Block et al., 2023), or generates corrective data under local continuity assumptions (Ke et al., 2024). Recent theory shows that expert action noise can mitigate exponential compounding error in continuous-control imitation learning (Zhang et al., 2025). PostBC (Wagenmaker et al., 2025a) and TMRL (Hong et al., 2026) similarly inject noise into actions or observation conditioning to broaden coverage for downstream RL. Our joint (o,a)-augmentation instead learns the joint observation–action distribution from the demonstrations and generates new (o, a) pairs together. Thus, the dataset itself determines both the observation variation and its compatible action, without additional expert collection, a hand-designed external noise distribution, or a local analytical dynamics model.

B Success Polarization Across Evaluation Conditions

Sec. 3 showed that pass@k evolves asynchronously during supervised pretraining, and attributed this behavior to a redistribution of success across initial evaluation conditions. Here, we examine this redistribution directly at the condition level and make explicit why it causes larger-k pass@k performance to decline first.

For the 200-demonstration Threading-D2 experiment, we evaluate every checkpoint on the same 500 evaluation conditions, with 32 rollout attempts per condition (pass@32). For each condition, we count the number of successful rollouts and track how the distribution of these counts changes over the course of supervised training.

Late training polarizes success across conditions. Fig. 7 makes the redistribution described in Sec. 3 explicit. Early in training, the improvement is largely uniform: the fraction of conditions with zero successes decreases and more conditions achieve higher success counts. Later, however, this trend reverses for the hardest conditions. Later in training, both the fraction of conditions with zero successes and the fraction with high success counts increase. Therefore, late training *polarizes* condition-wise success, improving reliability on a subset of conditions while reducing coverage on others.

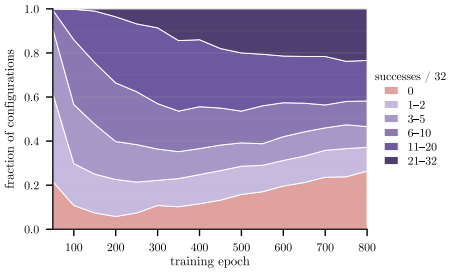


Figure 7: **Distribution of condition-level success counts over supervised pretraining.** For each checkpoint, we run pass@32 on each of 500 evaluation conditions and bin each condition by its number of successful rollouts within 32 trials. The stacked bands show the fraction of conditions in each success-count bin; each vertical slice sums to one. During late training, the fraction of conditions with zero successes increases, while the fraction with high success counts also increases.

Why polarization produces asynchronous pass@k profiles. Let $q_\tau(c)$ denote the one-rollout success probability for condition c at checkpoint τ. Then

$$P_k(\tau) = \mathbb{E}_c \left[1 - \big(1 - q_\tau(c)\big)^k \right].$$

Pass@1 averages $q_\tau(c)$ linearly. For $k > 1$, $f_k(q) = 1 - (1 - q)^k$ is concave. Thus, shifting success probability from a lower-q condition to a higher-q condition while holding mean success fixed leaves pass@1 unchanged but lowers pass@k.

$$\frac{\partial P_k}{\partial \tau} = \mathbb{E}_c \left[k(1 - q_\tau(c))^{k-1} \frac{\partial q_\tau(c)}{\partial \tau} \right].$$

As k increases, the factor $(1 - q_\tau(c))^{k-1}$ suppresses changes on high-success conditions more strongly. Large-k pass@k therefore depends more on whether low-success conditions remain solvable. Once those conditions begin to degrade, large-k pass@k begins to decrease, even if small-k pass@k continue to improve from improvements on already-reliable conditions.

C Additional Results on Residual RL Finetuning Across the Coverage–Reliability Tradeoff

To examine more broadly how pretraining coverage and reliability affect residual RL finetuning (Sec. 3), we additionally finetune a substantially earlier checkpoint that retains more coverage but has lower pass@1 than the checkpoint selected by our criterion (Sec. 4).

Fig. 8 shows that the relative advantage of this much earlier checkpoint depends on the available online budget. Its lower pass@1 can hurt early residual RL performance, while its greater coverage can become beneficial with more online budget; the exact tradeoff can be task dependent. While we do not claim that our criterion selects *the* optimal checkpoint for any online RL budget, it instead selects a checkpoint with substantially higher large-k pass@k than the final checkpoint while maintaining comparable pass@1; and across the evaluated budgets, this checkpoint consistently finetunes better than the final checkpoint from the same supervised run.

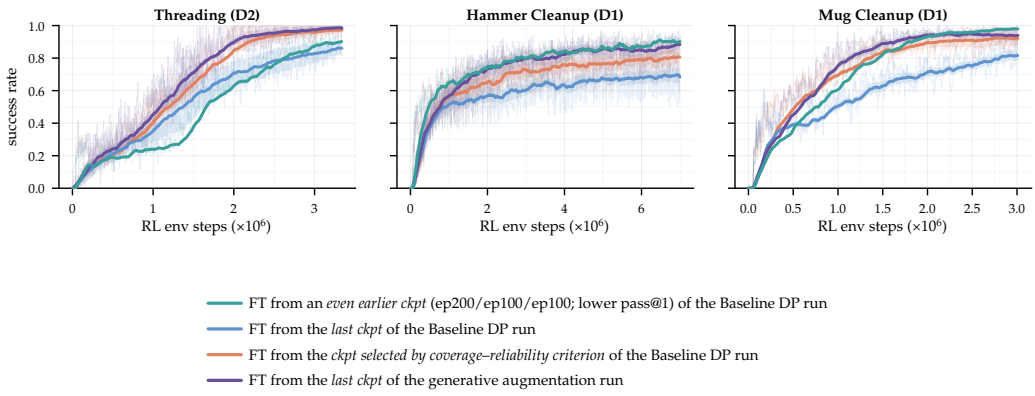


Figure 8: **Residual RL finetuning across the coverage–reliability tradeoff.** We additionally finetune an earlier baseline checkpoint (ep.200/100/100 for `Threading`/`Hammer Cleanup`/`Mug Cleanup`) with lower pass@1 and higher coverage than the checkpoint selected by our criterion. Its performance relative to later checkpoints depends on the available online budget.

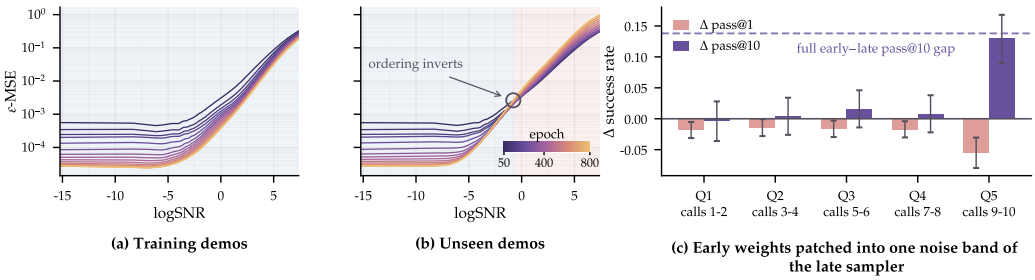


Figure 9: **Overfitting concentrates at low diffusion noise, and the same region largely controls the pass@k profile.** (a) On training data, denoising loss decreases throughout training across the full noise range. (b) On unseen data, later checkpoints have higher dsm loss near the low-noise end despite lower training loss. (c) In a 10-step DDIM sampler, replacing one two-step band of the late checkpoint with the corresponding early weights has little effect for Q1–Q4; replacing Q5, the final two denoising steps, recovers most of the early–late pass@10 gap and reduces pass@1, indicating the closed loop pass@k profile is largely determined by denoising calls at low noise level.

The exact ordering of checkpoints after RL finetuning can depend on both the RL algorithm and the task dynamics. Different RL methods explore and use online experience differently, and task dynamics affect how quickly differences between pretrained policies are amplified or corrected during closed-loop interaction. Our experiments establish the online rollout budget-dependent ordering of checkpoints for DICE-RL; we believe similar effects could also arise in other residual- or steering-based RL methods. Large-k pass@k should therefore be interpreted as an offline measure of how broadly a pretrained policy succeeds before finetuning, rather than as a universal predictor of checkpoint ordering after arbitrary RL finetuning procedures.

D Diagnosing Overfitting in Diffusion Policies

The main text establishes that extended supervised pretraining can improve pass@1 (reliability) while reducing large-k pass@k (coverage). Here we examine the associated finite-data overfitting. We test where it appears along diffusion noise levels and how it depends on local observation support.

Overfitting concentrates at low noise. Fig. 9(a,b) show denoising loss on the training and unseen data, respectively, across diffusion noise levels. On training data, loss decreases throughout supervised pretraining at every noise level. On unseen data, later checkpoints have higher denoising error near the low-noise end while remaining comparable better at high noise. Training loss therefore continues to improve while held-out loss already starts to worsen at low noise. We refer to this train–held-out divergence as overfitting; and it is concentrated near the low-noise end.

Independent data splits show fitting to the training split. Here we explain the data split experiment in main text Sec 3 in full details. We train two policies, $\pi_\tau^{(1)}$ and $\pi_\tau^{(2)}$, on disjoint demonstration sets D_1 and D_2, and evaluate both on the same held-out set D_3 with matched initial diffusion noise. Their prediction divergence on D_3 first decreases, then increases later in training. To test whether this late divergence reflects each policy moving toward its own training split, define the nearest-neighbor action distance

$$d_i(\bar{a}) = \min_{\bar{a}' \in D_i} \|\bar{a} - \bar{a}'\|_2 \tag{D.1}$$

and the normalized ratio

$$R_i(\tau) = \frac{\mathbb{E}_{o\sim D_3}\mathbb{E}_z\left[d_i\left(\bar{a}_\tau^{(i)}(o,z)\right)\right]}{\mathbb{E}_{(o,a^\star)\sim D_3}[d_i(a^\star)]}. \tag{D.2}$$

Here $\bar{a}_\tau^{(i)}(o,z)$ is an action chunk sampled from policy $\pi_\tau^{(i)}$ for held-out observation o and diffusion noise z. The numerator measures its distance to the policy's own training split D_i; the denominator measures the same distance for held-out expert actions. Thus $R_i < 1$ means policy samples are closer to D_i than held-out expert actions are, and a decrease in R_i indicates increasing bias toward that training split. This provides a second measure of finite-trainingdata overfitting.

Overfitting is stronger where observation support is sparse. We next test whether low-noise overfitting depends on how many training demonstrations are nearby in observation space. The policy is image-conditioned, making local support hard to measure directly; therefore, we use ground-truth simulator state to measure local training support. For a held-out query state s and a training episode e, define

$$d(s,e) = \min_{s' \in e} \|s - s'\|_2. \tag{D.3}$$

Let $r_{10}(s)$ be the 10th-smallest $d(s,e)$ across the 200 training episodes. A small r_{10} means that states near s occur in many distinct training episodes; a large r_{10} means fewer training episodes contain nearby states. We rank held-out queries by r_{10} and split them into equal thirds, giving the dense, medium, and sparse groups in Fig. 10.

For each query, noise level, and noise draw, let

$$\ell_\tau(o,t,\epsilon) = \|\epsilon - \epsilon_\tau(a_t,o,t)\|_2^2 \tag{D.4}$$

be the denoising loss at checkpoint τ. We evaluate checkpoints 200 and 800 on the same noisy input and report

$$\Delta L(o,t,\epsilon) = \ell_{800}(o,t,\epsilon) - \ell_{200}(o,t,\epsilon), \tag{D.5}$$

so $\Delta L > 0$ means that the later checkpoint has larger held-out denoising loss on that query.

Fig. 10 shows two effects. *First*, within every support group, ΔL is near zero at high noise and increases toward the clean end. *Second*, at the same noise level, sparse queries have larger ΔL than medium and dense queries, with the largest difference at low noise levels. Overfitting is therefore strongest when diffusion noise is low and local observation support is sparse.

Local-support analysis. We now show how the two factors above affect the number of training examples near a denoising query. This is a local finite-sample model, not a model of SGD or of the network's internal capability.

Assume the training windows $\mathcal{D}_n = \{(Z_i, A_i)\}_{i=1}^n$ are samples from a joint density $p_{Z,A}$ that is continuous near the held-out query. Here $Z_i \in \mathbb{R}^{d_z}$ is a conditioning representation and $A_i \in \mathbb{R}^{d_a}$ is the clean expert action. Take an independent held-out pair $(Z^\star, A^\star) = (z^\star, a^\star)$ and one initial diffusion noise draw ϵ. At diffusion time t,

$$x_t = \alpha_t a^\star + \sigma_t \epsilon, \qquad r_t := \frac{\sigma_t}{\alpha_t}, \qquad \widetilde{a}_t := \frac{x_t}{\alpha_t} = a^\star + r_t \epsilon. \tag{D.6}$$

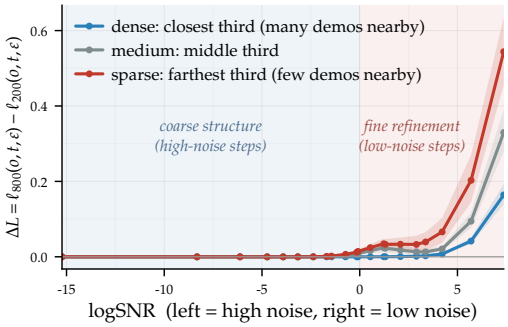


Figure 10: **Overfitting increases as observation support becomes sparser.** Late-minus-early denoising loss $\Delta L = \ell_{800}(o,t,\epsilon) - \ell_{200}(o,t,\epsilon)$ on held-out queries, grouped into equal thirds by local training support. Within every group, ΔL is near zero at high noise and increases toward low noise. At the same low-noise level, ΔL follows sparse $>$ medium $>$ dense, with the largest gap near the clean end.

For a candidate clean training action A_i, the forward diffusion process gives

$$p(x_t \mid A_i) \propto \exp\left(-\frac{\|x_t - \alpha_t A_i\|^2}{2\sigma_t^2}\right) \tag{D.7}$$

$$= \exp\left(-\frac{\|A_i - \widetilde{a}_t\|^2}{2r_t^2}\right). \tag{D.8}$$

Eq. D.8 shows that the likelihood depends on $\|A_i \widetilde{a}_t\|_2 / r_t$. An action at distance cr_t from $\widetilde{a}_t$ has relative likelihood $e^{-c^2/2}$. Therefore, for any fixed relative-likelihood threshold, the radius of the corresponding set of clean actions is proportional to r_t. As r_t decreases, this set shrinks around $\widetilde{a}_t$.

We now introduce a simple local-support model. Fix a condition-side radius h, and replace the smooth Gaussian weighting in Eq. D.8 with a hard action-space cutoff at r_t. We then count training examples that satisfy both conditions:

$$N_{\mathrm{loc}}(t) = \sum_{i=1}^{n} \mathbf{1}\{\|Z_i - z^\star\|_2 \le h\}\mathbf{1}\{\|A_i - \widetilde{a}_t\|_2 \le r_t\}. \tag{D.9}$$

Thus, an example is counted if its conditioning representation lies within distance h of $z^\star$ and its clean action lies within distance r_t of $\widetilde{a}_t$. Both hard cutoffs are analytical simplifications rather than claims about the model's capability.

Let q_t be the probability that one i.i.d. training pair is counted by Eq. D.9. Then

$$q_t = \Pr(\|Z - z^\star\| \le h, \|A - \widetilde{a}_t\| \le r_t) \tag{D.10}$$

$$= \int_{\|z - z^\star\| \le h} \int_{\|a - \widetilde{a}_t\| \le r_t} p_{Z,A}(z,a)\, da\, dz. \tag{D.11}$$

i.e. $N_{\mathrm{loc}}(t) \sim \mathrm{Binomial}(n, q_t)$ and $\mathbb{E}[N_{\mathrm{loc}}(t)] = nq_t$.

To evaluate q_t, substitute $z = z^\star + hu$ and $a = \widetilde{a}_t + r_t v$:

$$q_t = h^{d_z} r_t^{d_a} \int_{\|u\| \le 1} \int_{\|v\| \le 1} p_{Z,A}(z^\star + hu, \widetilde{a}_t + r_t v)\, dv\, du. \tag{D.12}$$

Near the low-noise end, $r_t \to 0$ and $\widetilde{a}_t = a^\star + r_t \epsilon \to a^\star$. For small h and r_t, continuity of $p_{Z,A}$ at $(z^\star, a^\star)$ therefore gives

$$\mathbb{E}[N_{\mathrm{loc}}(t)] = V_{d_z} V_{d_a}\, n p_{Z,A}(z^\star, a^\star) h^{d_z} r_t^{d_a} [1 + o(1)], \tag{D.13}$$

where V_d is the volume of the unit ball in $\mathbb{R}^d$.

Factoring $p_{Z,A}(z^\star, a^\star) = p_Z(z^\star)p_{A|Z}(a^\star \mid z^\star)$ gives

$$\mathbb{E}[N_{\mathrm{loc}}(t)] \approx Cn \underbrace{p_Z(z^\star)h^{d_z}}_{\text{local condition mass}} \underbrace{p_{A|Z}(a^\star \mid z^\star)r_t^{d_a}}_{\text{local action mass at noise scale } r_t}, \tag{D.14}$$

with $C = V_{d_z}V_{d_a}$. Sparse conditioning support reduces the first factor. Lower noise reduces $r_t = \sigma_t/\alpha_t$, so the second factor decreases as $r_t^{d_a}$.

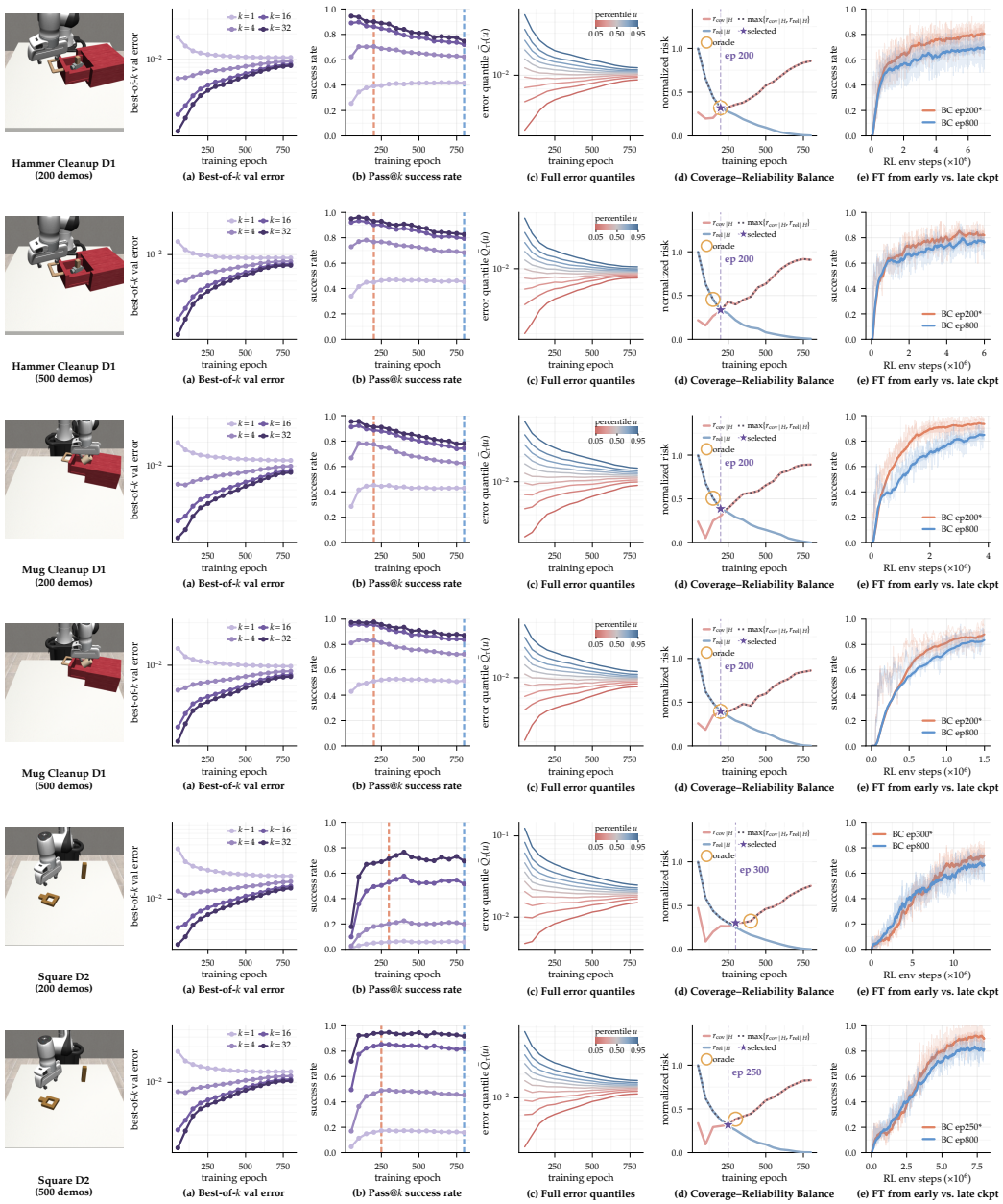
The r_{10} measure used in Fig. 10 is a finite-sample proxy for the first factor: under a locally smooth state density, a larger nearest-neighbor radius means lower local density.

Low-noise changes affect closed-loop pass@k. The analyses above are offline measurements. To test whether the same low-noise region changes closed-loop behavior, we partition the 10-step DDIM denoising calls used for inference into five consecutive two-step bands, Q1–Q5, and use the earlier checkpoint's weights for only one band at a time. Fig 9(c) shows that patching Q1–Q4 has little effect on pass@k profile, whereas replacing only Q5 restores most of the earlier checkpoint's pass@10 advantage and reduces pass@1. A change applied only to the final low-noise denoising calls is therefore able to shift the closed-loop reliability–coverage profile to a large extent.

In summary, the low noise queries have the narrowest compatible-action neighborhood, and sparse observation support also has fewer nearby training examples. The Q5 intervention shows that changes in low-noise band affect closed-loop pass@k.

E Full Coverage–Reliability Results and Checkpoint Selection

We report the complete results across five MimicGen tasks with 200 and 500 demonstrations, together with multitask RoboCasa pretraining. For each setting, we show best-of-k validation loss, closed-loop pass@k, the full error-quantile profile, coverage–reliability checkpoint selection, and downstream residual RL or supervised finetuning.



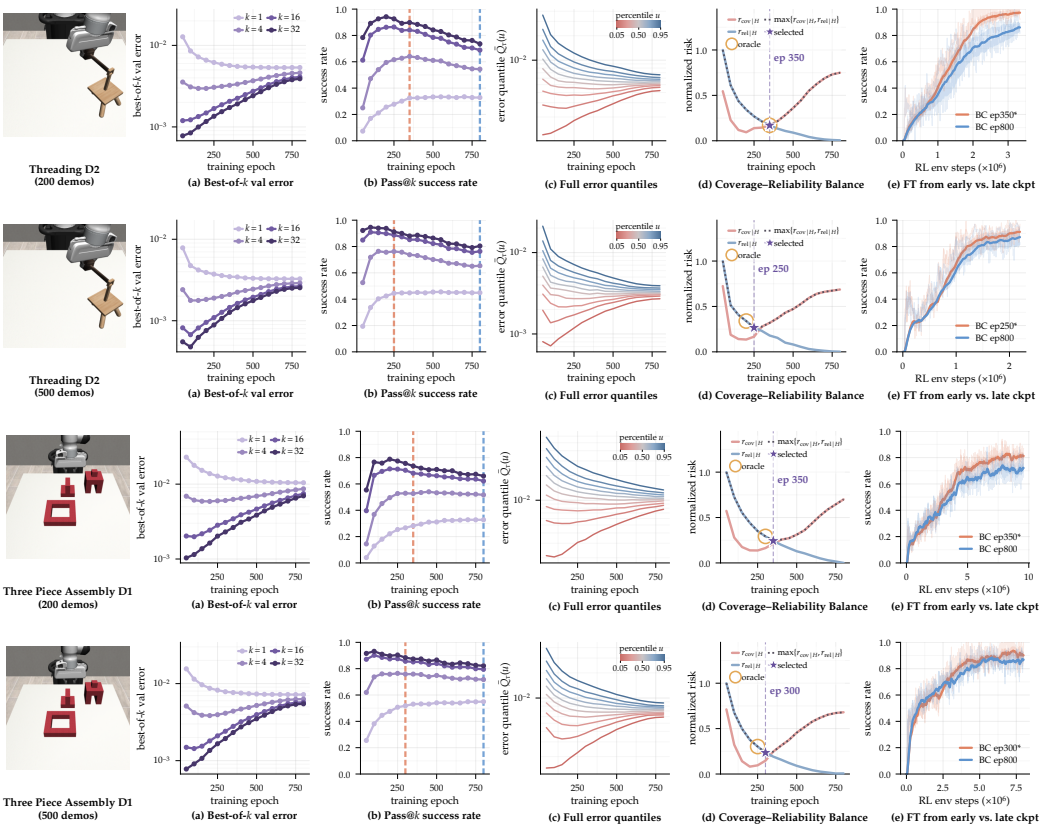


Figure 11: **Full MimicGen results across tasks and dataset sizes.** Results for all five tasks with 200 and 500 demonstrations. Each row reports best-of-k validation loss, closed-loop pass@k, the full error-quantile profile, coverage–reliability checkpoint selection, and downstream residual RL finetuning.

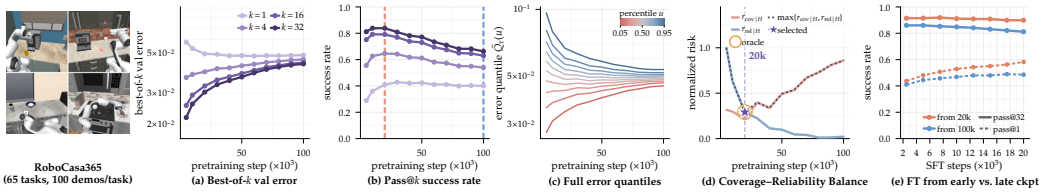


Figure 12: **Full RoboCasa multitask pretraining results.** Best-of-k validation loss, pass@k, error quantiles, coverage–reliability checkpoint selection, and downstream SFT.

The same learning dynamics hold across tasks and dataset sizes. Figures 11 and 12 show a consistent pattern across all settings. Best-of-k loss and pass@k favor increasingly earlier checkpoints as k increases, while the error-quantile curves exhibit the same tradeoff between favorable-tail coverage and average-case reliability as described in Sec. 4. The pattern holds for both single-task MimicGen BC and multitask RoboCasa pretraining. Importantly, pass@1 and k=1 best-of-k loss (mean held-out action MSE) generally favor later checkpoints after large-k pass@k has begun to decline.

Increasing the MimicGen dataset from 200 to 500 demonstrations generally attenuates, and in several cases delays, the late-stage loss of coverage. While two discrete dataset sizes are insufficient for a scaling-law claim, this trend is consistent with the finite-data overfitting diagnosis in Appendix. D.

Coverage–reliability balance criterion agrees with a closed-loop oracle. We apply the coverage–reliability criterion from Sec. 4 to every setting. For comparison, we define the closed-

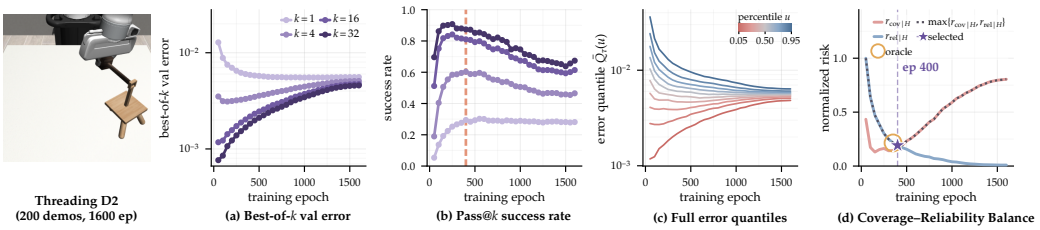


Figure 13: **Extended training on Threading D2.** The coverage–reliability dynamics persist to 1600 epochs, and the offline criterion continues to select an intermediate checkpoint.

loop oracle directly from the full pass@k table. Here we describe how we compute the checkpoint selected from oracle closed-loop pass@k runs. For each k, we normalize pass@k across checkpoints as

$$g_\tau(k) = \frac{\max_{\tau'} P_k(\tau') - P_k(\tau)}{\max_{\tau'} P_k(\tau') - \min_{\tau'} P_k(\tau')}, \tag{E.1}$$

where $P_k(\tau)$ denotes pass@k at checkpoint τ. We then select

$$\tau_{\text{oracle}} = \arg\min_{\tau} \max_{k} g_\tau(k). \tag{E.2}$$

This oracle selects the checkpoint whose worst normalized gap from the best checkpoint is smallest across the full pass@k profile. Across tasks and dataset sizes, the checkpoint selected by our proposed offline criterion lies near the closed-loop oracle.

Selected checkpoints also support stronger finetuning. For all MimicGen settings, residual RL initialized from the selected checkpoint outperforms RL from the final BC checkpoint. RoboCasa shows the same pattern under SFT: initialization from the selected pretraining checkpoint yields stronger pass@1 and pass@32 throughout finetuning. These results extend the main-text finding that the coverage–reliability balance retained during supervised training leads to stronger finetuning performance compared to the final checkpoint from the same run.

Selection is robust to training horizon. We additionally extend the 200-demonstration Threading D2 run from 800 to 1600 epochs. Figure 13 shows that the same pass@k and error-quantile dynamics persist under substantially longer training, while the criterion continues to select an intermediate checkpoint.

We further recompute both selectors using only checkpoints available up to each training horizon H. Figure 14 shows that the offline-selected checkpoint closely tracks the corresponding closed-loop oracle as additional checkpoints become available, rather than simply moving toward the end of training. The selection criterion therefore does not depend sensitively on the chosen training horizon.

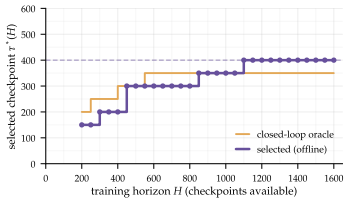


Figure 14: **Robustness to training horizon.** Offline selection tracks the closed-loop oracle as H increases.

F Additional Results on JointAug

Additional Task.

We additionally evaluate **JointAug** on 3-Piece Assembly (D2) with 500 demonstrations. This task provides a more extreme example of the effect of pretraining coverage on residual RL finetuning. As shown in Fig. 15, **JointAug** raises pass@10 to roughly 0.8 during BC pretraining, compared with below 0.1 for the baseline. Moreover, residual RL from the final **JointAug** checkpoint then reaches over 0.8 success, while finetuning from the baseline checkpoint remains near zero throughout the evaluated interaction budget. Thus, on this task, the pretraining difference changes residual RL from not working to successful finetuning.

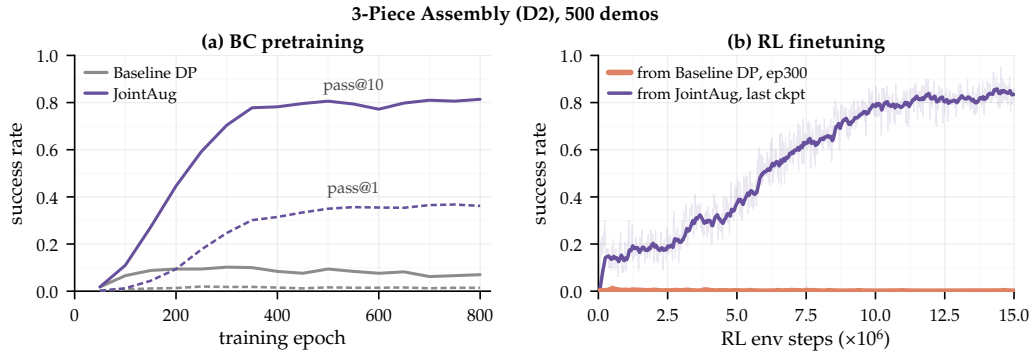


Figure 15: **JointAug enables residual RL finetuning on 3-Piece Assembly (D2).** *Left:* **JointAug** substantially improves pass@1 and pass@10 during BC pretraining. *Right:* residual RL from the final **JointAug** checkpoint reaches over 0.8 success, while finetuning from the baseline checkpoint remains near zero.

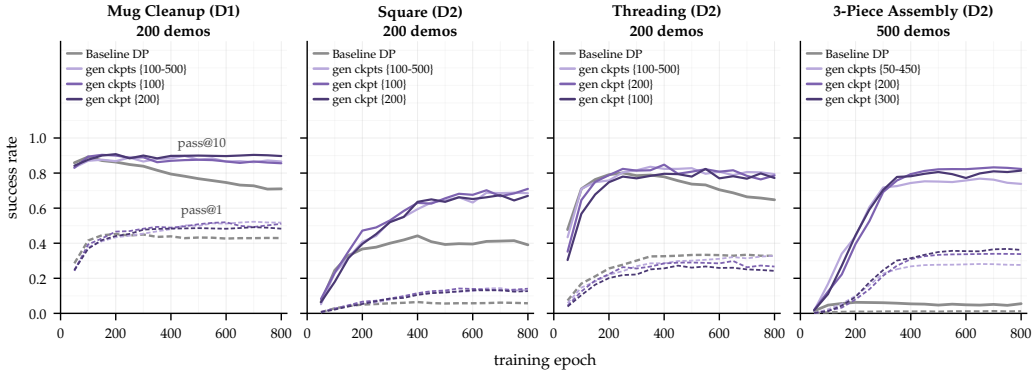


Figure 16: **JointAug is relatively robust to generator checkpoint selection.** We compare the five-checkpoint mixture used in our main experiments against individual generator checkpoints from the same run . Across four tasks, single-checkpoint variants have similar pass@10 gains over the baseline, while the mixture performs similarly overall and slightly improves pass@1 on `Mug Cleanup` and `Threading`.

Robustness to Generator Checkpoint Selection.

By default, **JointAug** uses a a uniform mixture of five generator checkpoints, spaced 100 epochs apart, around the transition identified by our train–held-out nearest-neighbor ratio (Appendix H.3). We use this short checkpoint window to average supervision across the generator's transition from population-level generalization toward increasing training-data bias.

To test how sensitive **JointAug** is to this heuristic choice, we compare the five-checkpoint mixture against distilling from individual generator checkpoints from the same region. Fig. 16 shows that the resulting policy curves are similar across these choices on all four tasks. In particular, single generator checkpoints preserve most of the pass@10 improvement over the baseline. The five-checkpoint mixture performs similarly overall and gives slightly higher pass@1 on `Mug Cleanup` and `Threading`. Together, these results show that **JointAug** does not depend strongly on selecting one particular generator checkpoint or checkpoint mixture. Mixing several nearby checkpoints provides a simple way to reduce checkpoint sensitivity.

Visualizing How JointAug Changes the Action Distribution.

To understand why **JointAug** improves both coverage and reliability, we inspect the action distribution at held-out observations. For each state, we sample 64 action chunks from an early base-



Figure 17: **JointAug retains favorable-tail coverage while improving typical action accuracy.** For four held-out `Mug Cleanup` states, we jointly project 64 action samples from baseline ep50, baseline ep800, and **JointAug** ep800 with PCA. Stars mark expert actions and rings mark the minimum-MSE sample. The baseline ep50 is dispersed with low best-of-64 but high mean MSE; baseline baseline ep800 contracts and lowers mean MSE but worsens best-of-64 MSE; **JointAug** ep800 achieves best of both worlds.

line checkpoint (ep50), the final baseline checkpoint (ep800), and the final **JointAug** checkpoint (ep800), and jointly project the samples with PCA. We report both the mean action MSE and the best-of-64 MSE to the expert action. The former measures the accuracy of a typical sample, while the latter measures whether the sampled distribution still contains an action close to the expert.

Fig. 17 shows that the early baseline checkpoint produces a broad action distribution: across the four states, best-of-64 MSE is very small, but mean MSE can go up to 2.32×10^{-3}. Thus, lucky draw samples lie very close to the expert, while the majority of samples are much farther away. This provides coverage but poor reliability for stable closed-loop execution.

By ep800, the baseline distribution contracts substantially. Mean MSE improves a lot from the early checkpoint, but best-of-64 MSE worsens on all four states. The later baseline therefore produces more consistent samples, but its best samples are farther from the expert, thus losing coverage.

The final **JointAug** policy checkpoint combines the favorable properties of both checkpoints: its action clouds remain broader than the final baseline while being centered closer to the expert. Mean MSE falls further to 0.18–0.64×10^{-3}, and best-of-64 MSE decreases to 0.06–0.23×10^{-3}. Thus, **JointAug** achieves lower mean and best-of-64 error than the final baseline on every state, while nearly matching or improving the favorable tail of the early baseline.

Together, these visualizations provide a post-hoc explanation for the pass@1 and large k pass@k improvement from **JointAug**. Jointly generated (o, a) pairs expand supervision beyond the finite demonstrations while preserving observation–action correspondence. The resulting policy retains a broader set of low-error actions without the large average errors of the early baseline.

Additional Baseline - Context-Smoothed Pre-training in TMRL.

We additionally compare against the context-smoothed pretraining part from **TMRL** (Hong et al., 2026), which is designed to increase action coverage by reducing the amount of information available in the policy context. During pretraining, **TMRL** applies a forward diffusion noise to the obser-

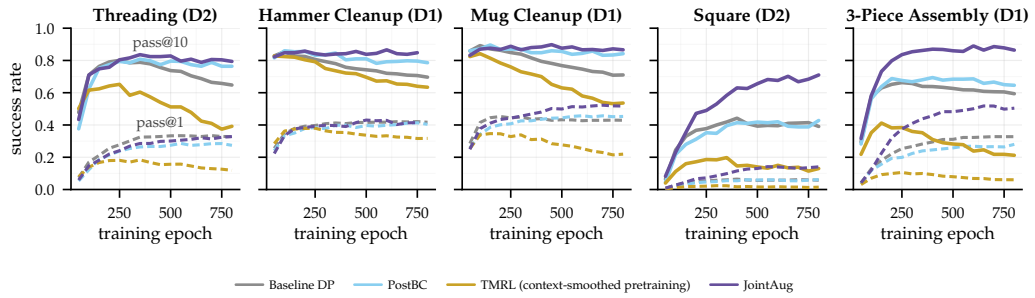


Figure 18: **Context-Smoothed Pre-training from TMRL does not preserve coverage under prolonged training.** Across five MimicGen tasks, TMRL generally underperforms JointAug in pass@10 and often loses both pass@10 and pass@1 at later checkpoints.

vation context,

$$q(\tilde{c}_t \mid c) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\, c,\ (1-\bar{\alpha}_t)I\right),$$

then conditions the policy on both the corrupted context $\tilde{c}_t$ and its corresponding noise level t. The intuition is that increasing t aliases a wider neighborhood of contexts, so the corresponding action distribution can borrow actions from other demonstrations rather than fitting only the conditional distribution at the original observation. We tune the maximum context-noise level for each task and report the best-performing setting; full implementation details are in Appendix H.5.

Fig. 18 shows that context smoothing does not preserve coverage well in our setting. Across tasks, its pass@10 remains below JointAug and often declines substantially with further training; its pass@1 also tends to degrade

One likely reason is that context smoothing relies on local structure in the latent space where noise is applied. Isotropic Gaussian perturbations are useful when nearby context representations correspond to observations with compatible actions, since smoothing can then borrow coherent behaviors from neighboring contexts. In our image-based diffusion policies, however, this assumption may not hold. The observation encoder is trained jointly from task demonstrations and is not explicitly regularized so that Euclidean neighborhoods in its latent space preserve action similarity. Context corruption can therefore mix observations that require different actions. We note that differs from the image-based experiments in Hong et al. (2026), which smooth pretrained VLM embeddings.

More generally, context smoothing increases overlap between conditional action distributions, but by design does not directly constrain the favorable tail of the action distribution as supervised training proceeds. It therefore does not guarantee preservation of large-k pass@k.

G Additional Results and Visualizations

Sampling Configuration Ablations.

Our main experiments use DDIM sampling with 10 denoising steps and $\eta = 0$. The coverage–reliability tradeoff develops during diffusion-policy training rather than being induced by a particular sampling procedure. Fig. 19 compares DDIM with 10 or 100 denoising steps, $\eta \in \{0, 0.5, 1\}$, and DDPM with 100 steps. Across all settings, pass@10 peaks earlier and declines with further training, while pass@1 continues to improve or remains stable. The same trend across sampling configurations shows that the late-training coverage loss is a property of the trained denoising model rather than the sampling configuration used for evaluation.

Denoising Network Size and EMA Ablations. Implementation choices could affect the observed learning dynamics between coverage and reliability, here we examine two: denoiser capacity and the EMA schedule. Across a 40× range of denoiser parameter counts and 3 EMA schedules, large-k pass@k shows the same early-peak-then-decline trend. These choices affect the timing and magnitude of the curves, but do not fundamentally remove the coverage–reliability tradeoff.

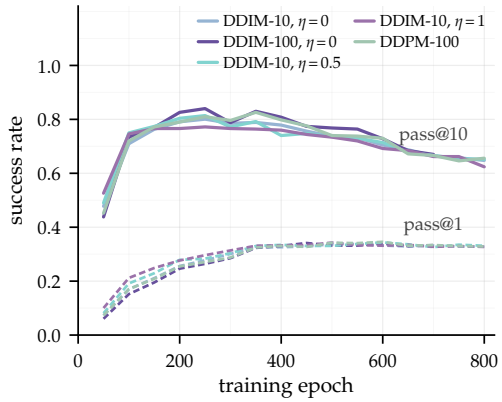


Figure 19: **Pass@k training dynamics are stable across sampling configurations.** Pass@1 and pass@10 are evaluated using DDIM with different numbers of denoising steps and η, and using DDPM. All settings show the same late-training decline in pass@10.

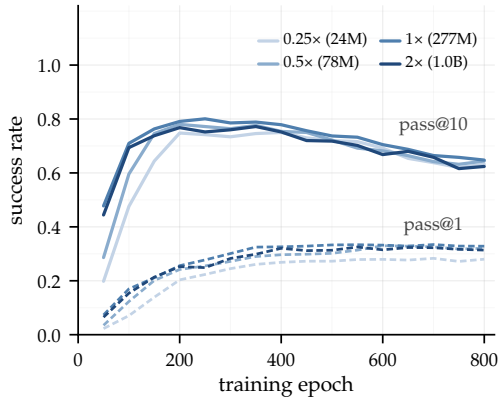


Figure 20: **The late-training coverage loss persists across denoiser widths.** We vary the U-Net from 24M to 1.0B parameters while keeping the encoder, data, optimizer, learning-rate schedule, and sampler fixed. All four models show a similar peak and subsequent decline in pass@10.

Denoising network width. We vary the U-Net channel widths while keeping the visual encoder, data, optimizer, learning-rate schedule, and sampler unchanged. We test four denoiser capacities contain 24M, 78M, 277M, and 1.0B parameters. As shown in Fig. 20, pass@10 peaks between roughly epochs 200–300 for every U-Net model size and then eventually declines to a similar level by epoch 800. Therefore, the loss of coverage during late training is not specific to the capacity of the denoising network.

EMA schedule. EMA changes how quickly the network weights used for evaluation track the current denoiser weights. A possible explanation for the late-training drop in large k pass@k is therefore that the EMA schedule forgets the earlier, higher-coverage weights too quickly. If so, increasing the EMA memory should substantially reduce or remove the decline. Let θ_t denote the current model parameters and $\bar{\theta}_t$ the EMA parameters. The EMA weights are updated as

$$\bar{\theta}_t = \beta_t \bar{\theta}_{t-1} + (1 - \beta_t)\theta_t,$$

where the decay follows the Diffusion Policy warmup schedule

$$\beta_t = \text{clip}\left(1 - \left(1 + \frac{s_t}{\gamma^{-1}}\right)^{-p}, \beta_{\min}, \beta_{\max}\right), \qquad s_t = \max(0, t - t_0 - 1).$$

Our default uses $\gamma^{-1} = 1$, $p = 0.75$, $t_0 = 0$, $\beta_{\min} = 0$, and $\beta_{\max} = 0.9999$.

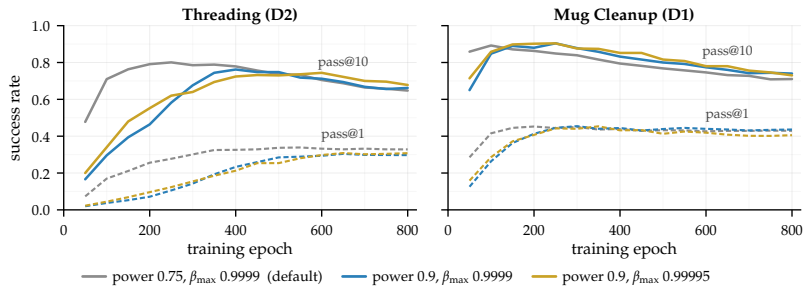


Figure 21: **Longer EMA memory delays and smooths the pass@10 decline but does not fundamentally remove it.** Holding $\beta_{\max} = 0.9999$ fixed, increasing p from 0.75 to 0.9 gives the EMA a longer memory earlier in training, causing pass@10 to peak and decline later. Holding $p = 0.9$ fixed, increasing $\beta_{\max}$ from 0.9999 to 0.99995 allows the EMA memory to continue growing later in training and further smooths the decline. Pass@10 still decreases under all schedules.

We ablate over two key hyperparameters, the power p and $\beta_{\max}$, each controling different parts of the EMA memory schedule. Increasing p makes $(1 + s_t)^{-p}$ decay faster, so β_t approaches one earlier in training. The EMA therefore puts less weight on the current network at each update and retains older weights for longer. In other words, larger p gives the EMA a longer memory earlier in training. By contrast, $\beta_{\max}$ becomes to matter once this growth reaches the cap value. Increasing $\beta_{\max}$ allows β_t to move even closer to one before being clipped, giving the EMA a longer maximum memory later in training.

We compare the default setting $(p = 0.75, \beta_{\max} = 0.9999)$ against $(0.9, 0.9999)$ and $(0.9, 0.99995)$. The first comparison increases the EMA memory earlier in training. The second keeps p fixed and increases the maximum memory later in training; for $p = 0.9$, the 0.9999 cap is reached around epoch 309, while the 0.99995 cap is reached around epoch 668.

Fig. 21 shows comparisons between these different temporal averaging schedules. Holding $\beta_{\max} = 0.9999$ fixed, increasing p from 0.75 to 0.9 gives the EMA a longer memory earlier in training. Accordingly, the $p = 0.9$ policy lags more while the underlying current model is improving: pass@10 rises more slowly and peaks later. Its decline after the peak is also more gradual because the EMA continues to place more weight on parameters from earlier checkpoints.

Holding $p = 0.9$ fixed isolates the effect of $\beta_{\max}$. The two schedules are identical until the $\beta_{\max} = 0.9999$ run reaches its cap; after that, $\beta_{\max} = 0.99995$ allows the EMA memory to keep growing and therefore retains earlier parameters for longer. This further smooths the late-training change in pass@10 for a little bit, but does not fundamentally prevent its decline. Thus, increasing either the early growth or the maximum length of the EMA memory can delay and smooth the loss of coverage, but neither prevents it.

Isolating Coverage Loss to the Denoising Network. To determine how much of the late-training coverage loss can be attributed to the visual encoder or the denoising network, we run three complementary experiments. (1) *Freeze the encoder*: we freeze the visual encoder at epoch 150 and continue training only the denoiser, testing whether removing further encoder updates changes the subsequent coverage decline. (2) *Fresh denoiser with a fixed encoder*: we train a new denoiser from scratch while keeping either the epoch-150 or epoch-800 encoder fixed throughout training. This isolates whether denoiser optimization alone can reproduce the peak-then-decline pattern under a fixed representation. (3) *Low-dimensional observations*: we train diffusion policies directly from state observations, removing the visual encoder entirely. This tests whether the same coverage–reliability tradeoff arises without any learned visual representations.

Freezing the encoder does not stop the coverage decline. We choose epoch 150 according to a split-data representation comparison shows that the nearest-neighbor structure of independently trained encoders has stabilized by this point. Starting from the epoch-150 checkpoint, we replace the encoder with its EMA copy, freeze all encoder parameters, and continue training the denoiser till the end of the training horizon. Fig. 22 shows that pass@10 still declines at later epochs, with a

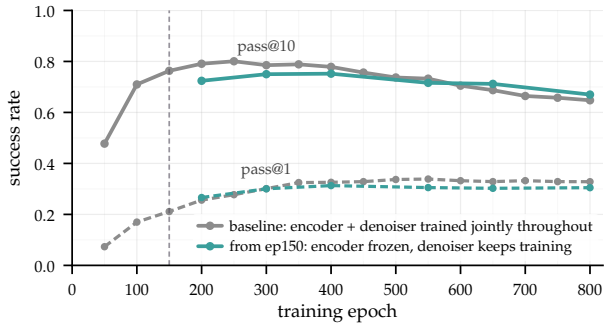


Figure 22: **Freezing the encoder does not prevent the late-training large k pass@k decline.** From epoch 150 onward, the encoder is fixed and only the denoiser is trained. Pass@10 still decreases at later epochs. The overall training dynamics is largely the same as baseline run.

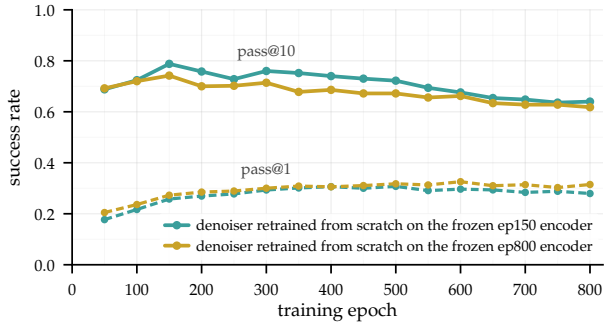


Figure 23: **Fresh denoisers reproduce the pass@10 decline with fixed encoder representations.** New denoisers are trained from scratch with either the epoch-150 or epoch-800 encoder frozen throughout training. Both show a later decline in large k pass@k.

similar pass@1 trend to the original run. Continued encoder updates are therefore not responsible for the coverage loss.

A fresh denoiser reproduces the decline with a frozen learned visual representation. We next randomly initialize a new denoiser and train it from scratch for 800 epochs while keeping the encoder fixed throughout training. We run this experiment with both the epoch-150 and epoch-800 encoders, using the same denoiser initialization. As shown in Fig. 23, pass@10 first increases and then declines for both frozen representatioons. The decline therefore reappears even when the encoder never changes and the denoiser is trained entirely from scratch.

The same trend appears without a visual encoder. Finally, we train diffusion policies directly from low-dimensional state observations. Across all five tasks in Fig. 24, baseline pass@10 again declines at later checkpoints, while pass@1 remains stable or improves. Thus, the late-training coverage loss happens in the absence of any visual encoder.

Together, these experiments show that visual-encoder overfitting cannot account for most of the late training coverage loss. Freezing the encoder does not eliminate the decline, a newly initialized denoiser reproduces it under a frozen representation, and the same trend appears in low dimensional state-based policies with no visual encoder at all. The visual encoder can still affect the magnitude of the effect, but the peak-then-decline behavior arises from denoiser training itself.

This result matches the generalization–memorization mechanism identified in prior work on diffusion models. With finite training data, the empirical denoising objective is minimized by the empirical score rather than the population score; and prolonged optimization can drive the denoiser toward memorizing training examples(Gu et al., 2023; Baptista et al., 2025; Ye et al., 2026). Network training initially favors the smoother, population-like components of the score and fits the

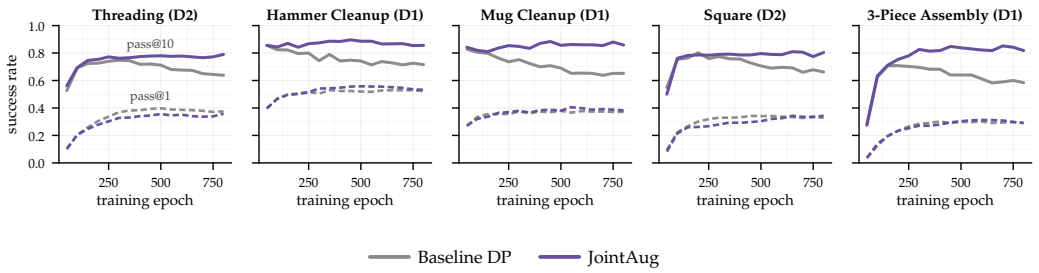


Figure 24: **Late-training coverage loss also occurs with *low-dimensional state observations*.** Across five tasks, baseline diffusion policies lose pass@10 at later checkpoints despite using no visual encoder. Joint (o, a) augmentation preserves substantially more pass@10.

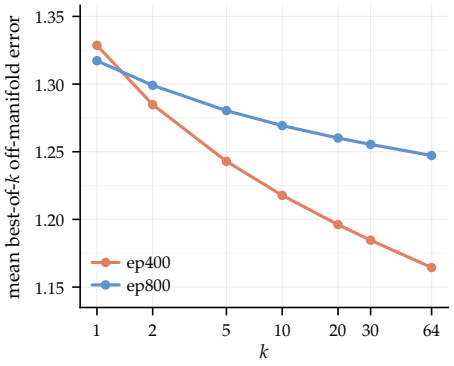


Figure 25: **Best-of-k off-manifold error reveals the same coverage–reliability tradeoff.** At $k = 1$, ep800 has slightly lower off-manifold error; from $k = 2$ onward, ep400 is better and the gap grows with k.

higher frequency, dataset-specific components later, particularly at low noise (Bonnaire et al., 2026; Favero et al., 2025). Our results show the same transition in conditional diffusion policies and it persists after removing visual-representation learning and is concentrated in the denoising network.

Best-of-k Off-Manifold Error Reveals the Same Coverage–Reliability Tradeoff.

We additionally ask whether the coverage–reliability tradeoff appears under a different offline metric of policy quality. Following the off-manifold metric of Pan et al. (2026), we perturb expert actions at high-ambiguity training states, execute the perturbed actions to obtain OOD observations, and measure the residual of each predicted action from the span of expert actions at nearby training states. We extend from the one policy prediction sample to best-of-k over stochastic policy samples.

We compare epochs 400 and 800 of `3-Piece Assembly` run. The later checkpoint has higher pass@1 (0.328 vs. 0.2988) but lower pass@10 (0.591 vs. 0.639). For each of 400 critical states, we generate 16 action-perturbed OOD observations. At each observation, we draw 64 policy action chunks and compute the expected minimum off-manifold error among k samples; $k = 1$ recovers the single-sample metric originally introduced by the paper.

Fig. 25 shows the same tradeoff in this metric. At $k = 1$, the later checkpoint has slightly lower off-manifold error. However, the ordering reverses at $k = 2$, and the gap increases with k. Thus, the later checkpoint produces a slightly better typical sample under the original single-sample metric, whereas the earlier checkpoint enjoys a substantially better favorable tail. Best-of-k off-manifold error therefore reveals the same coverage–reliability tradeoff observed with held-out action error and closed-loop pass@k.

How Policy Coverage Affects World-Model Learning.

Threading (D2)			
Training rollouts	$h = 1$	$h = 2$	$h = 3$
Selected	**0.0800**	**0.1033**	**0.1186**
Final	0.0815	0.1059	0.1219

Mug Cleanup (D1)			
Training rollouts	$h = 1$	$h = 2$	$h = 3$
Selected	**0.0992**	**0.1275**	**0.1458**
Final	0.1003	0.1291	0.1484

Table 3: **Higher-coverage rollout data improves held-out world-model prediction.** We report open-loop latent token MSE; lower is better. Each model uses the same 200 demonstrations plus 1,000 unfiltered rollouts from either the earlier selected or final policy checkpoint.

We additionally test whether policy coverage affects world-model learning from policy rollouts. For each task, we construct two training sets that share the same 200 expert demonstrations and differ only in the policy used to collect 1,000 additional unfiltered rollout episodes. One uses an earlier, higher-coverage checkpoint selected by the coverage-reliability criterion and the other uses the final, lower-coverage checkpoint. We keep failed rollouts because they provide additional state–action–next-state outcomes for learning dynamics and counterfactual reasoning on unseen actions.

We train the same latent dynamics model on each dataset where images are encoded once with the same frozen DINOv3 ViT-B/16 encoder (Siméoni et al., 2025), so the comparison is not affected by representation learning. Each model receives two frames of history and predicts the next three latent states open loop using the predicted actions. We evaluate both models on the same held-out episodes collected across the full checkpoint family.

Higher-coverage rollout data produces better held-out prediction. Tab. 3 shows lower prediction error for the early-rollout model at every horizon on both tasks. These results mirror the critic-learning experiment. Under a fixed rollout budget, broader policy coverage provides a wider range of state–action outcomes for learning. Because the rollouts are unfiltered, this includes both successful and failed behavior; both are useful for learning how different actions lead to different future states.

H Implementation Details

Tasks Datasets, and Evaluation. Across both MimicGen and RoboCasa365, we use the demonstration datasets released with the official repositories. Unless otherwise noted, experiments in the main paper use image observations as policy inputs.

For MimicGen, we train policies on 200 demonstration trajectories for the main-text experiments. We additionally report results with 500 demonstrations in Appendix E. For RoboCasa pretraining, we use 65 atomic tasks with 100 human demonstrations per task. For SFT, we use 100 demonstrations per target task.

For MimicGen BC evaluation, pass@k is computed over 500 held-out task configurations. For residual RL finetuning, we report the training success rate measured as the fraction of successful episodes among the 100 most recently completed episodes.

For RoboCasa evaluation, pass@k is computed over 50 held-out task configurations for each evaluated task.

For both task suites, we use the default task horizon specified by the official repos.

Baseline Single Task Diffusion Policy.

Our baseline single-task BC policy is a CNN-based diffusion policy following the implementation of Chi et al. (2025): a 1D temporal U-Net denoiser conditioned on a global observation embedding via FiLM, with a ResNet-18 visual encoder trained from scratch jointly with the denoiser network. The policy consumes a single observation step (two 84×84 RGB views, stacked as a 6-channel input, plus the 9-d proprioceptive state) and predicts an 8-step action chunk with 6-d rotation representation; all 8 predicted actions are executed open-loop before the next inference. Training uses the standard denoising objective: a uniformly sampled diffusion step $t \in \{0, \dots, 99\}$, forward noising with a squared-cosine β schedule, and an MSE loss on the predicted noise (ϵ-prediction). At

inference we sample with 10 DDIM steps ($\eta = 0$). All six MimicGen tasks share the identical architecture and optimization settings listed in Tab. 4.

Table 4: Baseline diffusion-policy configuration. All settings are shared across the six MimicGen tasks unless noted.

Hyperparameter	**Value**
Observation steps	1
RGB inputs	2 views, 84×84 (6-channel stack)
Proprioceptive state	9-d
Predicted action chunk size	8
Executed action chunk size	8
Visual encoder	
Backbone	ResNet-18
Normalization	GroupNorm
Encoder sharing	one encoder per camera (not shared)
Image augmentation	random crop to 76×76 (center crop at inference)
Denoiser	
Architecture	1D conditional U-Net
Channel dimensions	$[512, 1024, 2048]$
Kernel size / GroupNorm groups	5 / 8
Diffusion step embedding dim	128
Diffusion process	
Training diffusion steps	100
β schedule	squared-cosine ($\beta \in [10^{-4}, 0.02]$)
Prediction type / loss	ϵ-prediction
Inference sampler	DDIM, 10 steps, $\eta = 0$
Optimization	
Optimizer	AdamW ($\beta = (0.95, 0.999)$, $\epsilon = 10^{-8}$)
Learning rate	1×10^{-4}
Weight decay	1×10^{-6}
LR schedule	cosine, 500 warmup steps
Batch size	512
Training epochs	800
Demonstrations	200 (500 for Three-Piece Assembly D2 only)
EMA	inv_gamma 1.0, power 0.75, max decay 0.9999
Evaluation weights	EMA

`JointAug` Implementation.

`JointAug` augments policy training with samples from a joint generative model of the full observation–action tuple (o, a), trained on the same demonstrations as the policy. The generator is a pixel-space, dual-output stream diffusion model that denoises the $6\times84\times84$ two-camera frame, the 9-d proprioceptive states together with the 8×10 action chunk under a single shared diffusion process. Image tokens and the low-dimensional tokens are concatenated, jointly self-attend, and are split again, so every modality is generated conditioned on the others rather than independently. All three modalities are noised at the same timestep and the model predicts the v-target for each; the loss is a weighted sum of per-modality mean squared errors with equal weights, which prevents the image from drowning out the 80-d action.

Inference. Samples are drawn with the generator's EMA weights using a DDIM sampler with 100 steps. Training uses 1000-step squared-cosine schedule.

Selecting the generator checkpoint window. We save generator checkpoints every 50 epochs. For each saved checkpoint G_γ, we draw 512 joint samples $(\tilde{o}, \tilde{a}) \sim G_\gamma$ and compare each generated action chunk with two equally sized sets of expert action chunks: the demonstrations used to train the generator, $\mathcal{D}_{\text{train}}$, and a held-out set $\mathcal{D}_{\text{held}}$ of the same task that are used to train neither the generator nor the policy. Let $d(\tilde{a}, \mathcal{D}) = \min_{a' \in \mathcal{D}} \|\tilde{a} - a'\|_2$ be the nearest-neighbor distance on

normalized action chunks. We define

$$\rho(\gamma) = \frac{\mathbb{E}_{\tilde{a}\sim G_\gamma}[d(\tilde{a}, \mathcal{D}_{\text{train}})]}{\mathbb{E}_{\tilde{a}\sim G_\gamma}[d(\tilde{a}, \mathcal{D}_{\text{held}})]}. \tag{H.1}$$

$\rho(\gamma) = 1$ means generated actions are equally close to training and held-out demonstrations. We set γ_0 to the first saved checkpoint with $\rho(\gamma) < 1$, when generated samples first become closer to the training data than to held-out data, and sample from γ_0 and the four subsequent checkpoints spaced 100 epochs apart. Table 6 lists the resulting checkpoints for each task.

Distillation into policy training. During policy training, 50% of minibatch's rows is replaced by freshly generated (o, a) tuples; replaced and dataset rows are then trained with the policy's ordinary behavior-cloning loss, so there's no change to the policy architecture of Appendix H.2.

Table 5: Joint (o, a) generator configuration. All settings are shared across the six MimicGen tasks unless noted.

Hyperparameter	**Value**
Image	2 views, 84×84 (6-channel stack)
Proprioceptive state	9-d
Action chunk	8 steps, absolute pose, 6-d rotation (10-d)
Observation steps	1
Image stream	
Architecture	conv U-Net
Channel dimensions	$[96, 192, 384]$
Bottleneck	self-attention at 21×21
Low-dimensional stream	
Architecture	DiT over 9 tokens (1 state + 8 action)
Model width / layers / heads	384 / 6 / 6
Dropout	0.1
Cross-modal coupling	
Bridge	MM-DiT block at the bottleneck (joint self-attention)
Time embedding dim	256
Diffusion process	
Training diffusion steps	1000
β schedule	squared-cosine ($\beta \in [10^{-4}, 0.02]$)
Prediction type	v-prediction
Loss	per-modality mean MSE, weights 1:1:1 (image:state:action)
Sampler	DDIM, 100 steps
Sampling weights	EMA
Optimization	
Optimizer	AdamW ($\beta = (0.95, 0.999)$, $\epsilon = 10^{-8}$)
Learning rate	1×10^{-4}
Weight decay	1×10^{-4}
LR schedule	cosine
Batch size	256
EMA	inv_gamma 1.0, power 0.75, max decay 0.9999
Demonstrations	200 (500 for Three-Piece Assembly D2)

PostBC Implementation. We follow the image instantiation of Algorithms 1 and 4 (Appendix D of Wagenmaker et al. (2025a)). An ensemble of K members is first fit to the policy's own training split; each member f_l is a freshly initialized copy of the policy's observation encoder followed by a 3-layer MLP head that regresses the flattened 8×10 action chunk by mean squared error, and each is fit on its own bootstrap resample of that split. The ensemble is trained once per task, before and independently of policy training, with the settings in Table 7.

Posterior uncertainty is estimated from the disagreement of the ensemble predictions, cached per training window offline. At every fetch we draw a fresh noise vector from the diagonal of that covariance, scale it by α, and add it to the clean target action with the gripper dimension zeroed;

Table 6: Per-task **JointAug** settings. Generator checkpoints are indexed by their training epoch; rows of each minibatch are drawn equally from the listed checkpoints

	Mug D1	Square D2	Threading D2	Hammer D1	3-Piece D1	3-Piece D2
Demonstrations	200	200	200	200	200	500
Generator checkpoints (training epoch)	100, 200, 300, 400, 500	100, 200, 300, 400 500	100, 200, 300, 400, 500	100, 200, 300, 400, 500	100, 200, 300, 400, 500	50, 150, 250, 350, 450

policy training is otherwise unchanged from Appendix H.2 or H.7. For MimicGen tasks, we use $\alpha = 2.0$. For RoboCasa, we use $\alpha = 1.0$.

Table 7: **PostBC** posterior-ensemble config and noise injection. All settings are shared across the five MimicGen tasks for which we report **PostBC**.

Hyperparameter	Value
Ensemble members	
Ensemble size K	20
Member architecture	policy observation encoder + MLP head
Head width / layers	512 / 3
Member output	flattened action chunk ($8 \times 10 = 80$-d)
Ensemble optimization	
Optimizer	Adam
Learning rate	3×10^{-4}
Batch size (per member)	128
Training epochs	200
Checkpoing used	40
Noise injection	
Noise strength α	2.0
Excluded dimensions	gripper (final action dimension, scale 0)

Context-Smoothed Pre-training in TMRL Implementation. Context-smoothed pre-training Hong et al. (2026) widens a BC policy's support by corrupting its conditioning during pretraining, so that one network models $p(a \mid \tilde{c}, t_c)$ from conditional action distribution at $t_c = 0$ to the marginal action distribution at $t_c = 1$. We keep the paper's corruption kernel and noise schedule: a discrete linear β schedule over $[10^{-4}, 0.02]$ with $T_c = 100$ context steps, the variance-preserving marginal $\tilde{c} = \sqrt{\bar{\alpha}_{t_c}}\, c + \sqrt{1 - \bar{\alpha}_{t_c}}\, \epsilon$, and $t_c \sim \mathrm{Unif}(0, 1)$ drawn independently of the action timestep t_a. We evaluate with a clean context, $t_c = 0$.

Residual RL Implementation.

We use DICE-RL (Sun & Song, 2026) as an instantiation of residual RL finetuning. In DICE-RL, a frozen pretrained diffusion policy is finetuned by additive correction from a residual actor. Let s be the observation embedding of the frozen policy and z the initial DDIM noise; the base policy deterministically maps (s, z) to a chunk $\pi_{\text{pre}}(s, z)$ of $H = 8$ actions, and the executed chunk is $a = \pi_{\text{pre}}(s, z) + r_\theta(s, z)$, where only the residual MLP r_θ is trained. Since the chunk is executed open-loop, one transition spans the entire chunk and the critic $Q_\phi(s, a)$ scores the whole chunk. Because image features learned by BC on MimicGen tasks carry a weak value signal, we parameterize the residual actor and the critic with the low-dimensional state; the frozen pretrained policy still remains image-based. Episodes end at the first success step or at the task horizon, with truncation treated as terminal. Hyperparameters are identical across tasks (Table 8). For each residual RL experiment, we run with three random seeds.

Multitask Training on Robocasa.

We use the RoboCasa release of `openpi` and follow the RoboCasa $\pi_{0.5}$ benchmark recipe unchanged, except for the pretraining learning rate (Table 9).

Table 8: Residual RL finetuning configuration, identical across tasks; only the frozen base checkpoint and the environment horizon change.

Hyperparameter	Value
Frozen base policy	EMA weights, DDIM 10 steps ($\eta = 0$), chunk $H = 8$
Residual actor	MLP $[1024, 1024, 1024]$, GELU, LayerNorm
Critic ensemble	$10\times$ MLP $[1024, 1024, 1024]$, GELU, LayerNorm
BC weight λ_{BC}	200
TD loss / n-step	MSE / $n = 3$ chunks
Discount γ / target update τ	0.99 / 0.01
Parallel environments	64
Gradient steps per iteration	40
Batch size	256
RLPD expert ratio	$0.5 \rightarrow 0.1$ over the first 10k training steps
Optimizer	Adam, lr 1×10^{-4}, weight decay 1×10^{-5}
Actor update frequency	2
BoN Exploration	$\arg\max_k \min_e Q$ over $K = 16$ samples

Pretraining. Starting from `pi05_base`, the language model is frozen, the vision encoder is trained at $0.1\times$ the base learning rate, and the action expert at the base learning rate, for 100k steps at batch size 256 with AdamW, a 1,000-step warm-up, and cosine decay to $0.1\times$ the peak. The benchmark uses a peak learning rate of 2.5×10^{-5}; we instead uses 1×10^{-4} ($4\times$ higher, decaying to 1×10^{-5}), which we find to work better, with the vision-encoder scaling unchanged.

Supervised fine-tuning. Each fine-tuning run initializes all weights from a pretraining checkpoint and uses the benchmark fine-tuning recipe as is: peak learning rate 2.5×10^{-5} with the same warm-up and cosine schedule (50k steps), batch size 256, the same frozen/scaled parameter groups, and normalization statistics recomputed on the SFT data. Runs differ only in their initialization; checkpoints are saved every 2.5k steps and evaluated on the SFT tasks up to 20k steps.

Table 9: $\pi_{0.5}$ training configuration (pretraining / fine-tuning). All values follow the RoboCasa $\pi_{0.5}$ benchmark except the pretraining peak learning rate (benchmark: 2.5×10^{-5}).

Hyperparameter	Value
Peak learning rate	$\mathbf{1 \times 10^{-4}}$ / 2.5×10^{-5}
Schedule	1k-step warm-up, cosine decay to $0.1\times$ peak
Training steps	100k / 50k
Global batch size	256
Optimizer	AdamW (β_1=0.9, β_2=0.95, wd 10^{-10}), grad clip 1.0
EMA decay (evaluated weights)	0.99
Action chunk / cameras / image size	50 steps / 3 / 224×224