

The Gaussian Is Enough: Flow-Matching Priors Do Not Help When Fine-Tuning Large Behavior Models

Chen Xu¹, Rishi Shah², Hadas Kress-Gazit³, Haruki Nishimura¹, and Masha Itkina¹

¹Toyota Research Institute ²Woven by Toyota ³Cornell University

Abstract—Modern robot imitation learning increasingly relies on generative policies based on diffusion or flow-matching models, which generate actions by transforming samples from a prior distribution. A key question is whether the choice of prior matters. Replacing the standard Gaussian with a closer-to-target, non-Gaussian prior has been shown to substantially improve performance when *training from scratch*. A natural next step is to ask whether these gains transfer to *fine-tuning* pretrained Large Behavior Models (LBMs) such as LBM 1.0, $\pi_{0.5}$, and GR00T N1.5, where one might expect even larger gains. Surprisingly, we find that this is not the case, except possibly at very low fine-tuning data fractions. Across over 100K simulation rollouts spanning all three aforementioned LBMs on 40+ tasks in two simulation platforms, and 1250 hardware rollouts on five bimanual manipulation tasks, non-Gaussian priors that are demonstrably closer to the target yield statistically indistinguishable or worse fine-tuning performance than a standard Gaussian prior. Diagnostic analyses suggest why: fine-tuned imitation learning policies converge to similar action predictions across priors, despite their fine-tuned *encoder embeddings* diverging substantially from the pretrained embeddings and each other. A learning-rate ablation further confirms that encoder training is the dominant factor in fine-tuning performance, substantially outweighing the effect of prior choice. We conclude with concrete directions for future research on when and why learned priors might still matter in fine-tuning. Project page: https://cxu-tri.github.io/non_gaussian_FT/.

Index Terms—Imitation learning, flow-matching & diffusion policies, action priors, large behavior models, fine-tuning.

I. INTRODUCTION

IMITATION learning for robot manipulation increasingly employs generative models, such as diffusion and flow-matching policies [1], [2], which transport samples from a prior distribution p_0 to a target action distribution conditioned on observations. Large Behavior Models (LBMs) scale this approach: pretrain once on diverse vision, language, and robot action data, then fine-tune on a modest set of task-specific demonstrations. Recent examples include LBM 1.0 [3], $\pi_{0.5}$ [4], and GR00T N1.5 [5]. By default, these LBMs use an isotropic Gaussian $\mathcal{Z} \sim \mathcal{N}(0, I)$ as the prior, though flow matching [6] is agnostic to p_0 by construction. In other words, the Gaussian is a convention, not a requirement. A line of recent work [7]–[10] has exploited this flexibility when *training from scratch*: learning single-task policies without loading weights pretrained on large-scale data. Replacing the Gaussian with a closer-to-target prior substantially improves success rates on common benchmarks such as LIBERO [11], MetaWorld [12], and Adroit [13]. It remains unknown whether this benefit carries over to *fine-tuning* pretrained LBMs, where

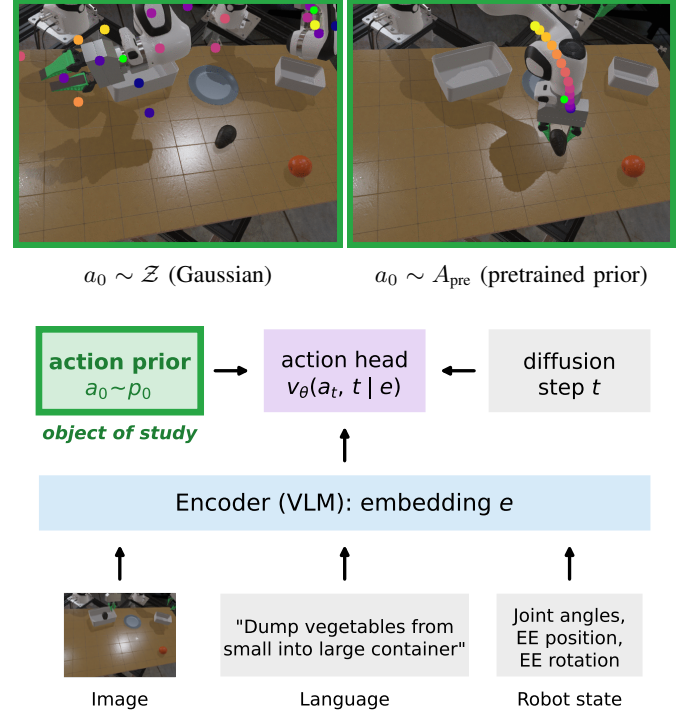


Fig. 1. **Prior definition and visualization.** **Bottom:** schematic of an example flow-matching policy. The action head v_θ is conditioned on the encoder embedding e and takes in diffusion step t as well as the action prior p_0 . Only the prior varies across our experiments. **Top:** samples drawn *directly from two examples of the prior p_0* on the simulated DumpVegetablesIntoContainer task from LBM Eval [3], with overlaid colored dots showing both arms' predicted end-effector positions over the action chunk horizon. Color gradient indicates temporal ordering within the chunk, with lighter dots representing predictions further into the future. The Gaussian $\mathcal{Z}$ (left) produces incoherent, random motion, while A_{pre} (right) already makes partial task progress by picking up the avocado and moving it to the bin, reflecting a prior close to the target action. We hypothesize that the close-to-target prior A_{pre} would outperform $\mathcal{Z}$ after policy fine-tuning, but we empirically find that this is not the case.

policies are initialized from pretrained weights rather than learned from scratch.

A natural question arises: *Can we fine-tune more capable policies by starting from an informed action prior rather than a standard Gaussian?* Swapping the Gaussian for another distribution, such as actions sampled from the pretrained LBM itself, requires no change to the training pipeline. The intuition is appealing: if the prior is already close to the target, the policy only needs to traverse a short transport distance in action space, and doing so should facilitate fine-tuning and improve policy performance (Fig. 1).

We therefore evaluate whether the from-scratch-training benefits of non-Gaussian priors persist after LBM pretraining.

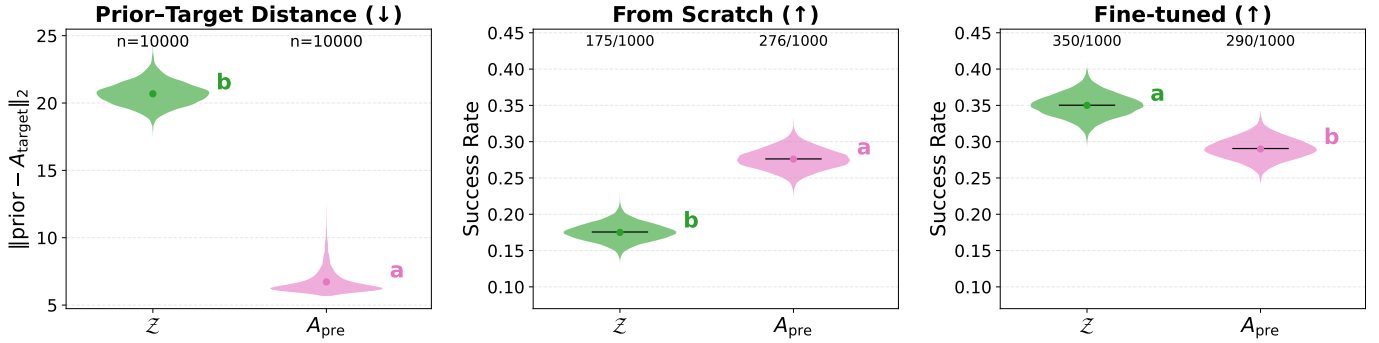


Fig. 2. Action priors help when training from scratch but not when fine-tuning (Setting A, 5 LBM Eval kitchen tasks [3], 1000 rollouts per prior; see Section VI-A for details). All fine-tuning prior variants start from the same pretrained LBM 1.0 checkpoint. A_{pre} denotes the action prior sampled from the pretrained policy. **Left:** ℓ_2 distance between prior samples and ground-truth actions, in un-normalized action coordinates; A_{pre} is $3\times$ closer to the target data than the Gaussian. **Middle:** from-scratch training; A_{pre} significantly outperforms the Gaussian. **Right:** after fine-tuning, the Gaussian prior matches or, in this case, exceeds A_{pre} in success rate. The full comparison across all seven prior variants (Fig. 3) confirms that most priors we consider yield statistically indistinguishable performance after fine-tuning, and, thus, provide no benefit to the policy.

If so, learned priors would compound the gains from pretraining with no change to the deployment pipeline. Surprisingly, we consistently obtain negative results to this question based on average success rates and average task progress across tasks (Fig. 2). The priors that help when training from scratch almost always do not improve fine-tuning performance, and in several regimes they do worse than the default Gaussian prior.

To our knowledge, this paper is the first large-scale study of non-Gaussian action priors for LBM fine-tuning. We make three contributions:

- *A negative finding at scale:* Non-Gaussian priors that are demonstrably closer to the target yield fine-tuning performance that is statistically indistinguishable from, or worse than, a standard Gaussian. This holds both on average and for the majority of individual tasks.
- *Consistency across the fine-tuning design space:* The finding is consistent across seven prior constructions, most data fractions (25%–100% of fine-tuning data), three LBM architectures (LBM 1.0, $\pi_{0.5}$, and GR00T N1.5), as well as both simulation and robot hardware settings. One exception is the very-low-data regime (5% fraction), which we discuss in Section VIII.
- *An introspective explanation:* Across priors, three diagnostics reveal that fine-tuned encoder embeddings diverge from pretrained embeddings and each other, yet yield indistinguishable conditional action likelihoods, and the resulting policy head outputs comparable action predictions. A learning-rate ablation confirms that encoder training matters far more than prior choice for policy performance, pointing to the pretrained encoder, not the prior, as the dominant factor. In short, action priors shape fine-tuning training dynamics, particularly in the observation embedding space, but do not affect policy performance.

II. RELATED WORK

A. Flow-Matching and Diffusion Policies for Robotics

Diffusion and flow-matching policies have become prominent approaches for visuomotor imitation learning. At their core, both families define time-dependent transport between

a prior and target distribution in action space. Single-task diffusion policy [1] introduced the formulation, which was then extended in BESO [2] to goal-conditioned settings, and scaled in LBMs [3]–[5], [14]. Compared to DDPM-based diffusion models [15], flow-matching variants [6] take a more direct path from the prior to the target distribution and make no distributional assumption on the prior. These desirable properties are central to recent flow policies [4], [5] and to this paper. Within the generative policy paradigm, LBMs (e.g., LBM 1.0 [3], $\pi_{0.5}$ [4], and GR00T N1.5 [5]) pretrain on large-scale, diverse manipulation data before fine-tuning on downstream tasks, but by default use the Gaussian prior distribution in both pretraining and fine-tuning. Parallel empirical studies of fine-tuning design choices (action-space parameterization [16], data scaling [17], co-training [18]) have examined many axes of the recipe but hold the prior fixed. Our paper investigates the effect of action prior choice on policy fine-tuning performance.

B. Pretraining, Transfer Learning, and Introspection

Fine-tuning adapts a pretrained model’s learned representations to new tasks, a form of transfer learning that has proven highly effective. Prior works have shown that pretrained features improve performance in both computer vision [19], [20] and robotics: frozen pretrained encoders [21], [22], shared trunks [23], and full pretrained policies [3]–[5] all outperform training from scratch. Other works examine *what explains* whether a pretrained model transfers well to a new task. Training-free probes such as LogME [24] estimate the predictive power of pretrained features. Scaling-law studies [25], [26] and likelihood-based assessments [27] examine when pretraining metrics can be trusted as proxies for downstream policy performance. Our work applies these introspective tools to the choice of prior: we compare priors using downstream performance (Section VI) and training-free metrics, such as LogME, per-observation ℓ_2 distance, and cosine similarity in action and embedding spaces (Section VII).

C. Non-Gaussian Prior Distributions

A growing body of work has proposed structured non-Gaussian priors for policy learning. BRIDGER [8] trains a

conditional variational autoencoder (CVAE) to produce informative action priors. Cocos [9] uses a condition-dependent Gaussian whose parameters are learned from observations to prevent loss collapse. Retrieval [39]–[41] constructs the action prior from nearest-neighbor actions, with distances measured in the observation embedding space. Action-to-action flow matching [7] maps demonstration actions directly to target actions for single-step inference. Other variants include equivariance guarantees for non-Gaussian priors [10], cluster-based and residual-action priors [28], [29], and Schrödinger-bridge formulations [30], [31]. In each case, however, the proposed prior is evaluated by training a policy from scratch; none of these works investigates whether the benefit transfers to fine-tuning a pretrained LBM.

A complementary line of research modifies the prior without changing the policy’s weights. Frozen-policy steering methods operate on the latent noise space of a trained policy: DSRL [32] defines a reward function over the noise space to denoise high-reward noise samples, Golden Ticket [33] searches for optimal constant noise vectors, and Golden-Start [34] trains a Q-guided CVAE noise prior. Related inference-time warm-starting methods [35]–[38] adjust the prior at rollout time. These methods change sampling, not training, and apply to any pretrained policy. Since we focus on the effects of training-time priors in fine-tuned policies, these approaches are outside the scope of this work.

III. PROBLEM SETUP

Setting. We fine-tune a pretrained LBM π_{pre} on task-specific demonstration datasets and consider whether replacing the default Gaussian prior with a non-Gaussian prior improves the fine-tuned policy performance. The pretrained policy π_{pre} can be any flow-matching- or diffusion-based LBM (in our experiments: LBM 1.0 [3], $\pi_{0.5}$ [4], or GR00T N1.5 [5]).

Notation. We follow the diffusion policy notation [1]. The target a_1 is an action chunk (H steps of future actions) drawn from the demonstration dataset $\mathcal{D}$. The raw observation o (camera images, language instructions, proprioception) is passed through the LBM’s vision-language-action encoder to produce the observation embedding e , which conditions the action-head network v_θ . The action head is trained to transport prior samples $a_0 \sim p_0$ to targets a_1 . We write $a_0 \sim p_0$ for a generic prior sample; the specific prior constructions (e.g., $\mathcal{Z}$, A_{pre} , etc.) defined in Section IV each instantiate p_0 . We use “prior” throughout the paper to refer to p_0 .

Why flow matching is the right testbed. Flow matching (FM) [6] regresses a velocity field directly between prior and target samples. The conditional FM objective

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t, a_0, a_1} \|v_\theta(a_t, t \mid e) - (a_1 - a_0)\|^2, \quad (1)$$

$$a_t = (1 - t)a_0 + ta_1,$$

only requires paired samples (a_0, a_1) from the prior and target, so swapping p_0 from $\mathcal{N}(0, I)$ to any other distribution requires no change to the training or inference procedure. DDPM-style diffusion models [15], by contrast, are designed around a forward noising process whose stationary distribution is Gaussian. Replacing this distribution would require redesigning the

TABLE I
THE SEVEN PRIOR VARIANTS COMPARED IN THIS WORK. π_{pre} IS THE PRETRAINED LBM (LBM 1.0, $\pi_{0.5}$, OR GR00T N1.5). APPENDIX A GIVES FULL SAMPLING PROCEDURES AND AUXILIARY-TRAINING DETAILS.

Prior	Description
$\mathcal{Z}$	Isotropic Gaussian $\mathcal{N}(0, I)$
<i>Proposed priors (constructed from π_{pre})</i>	
A_{pre}	Actions generated by π_{pre}
$A_{\text{pre}} + \sigma_A \mathcal{Z}$	Actions from π_{pre} with added Gaussian noise (scale σ_A)
$E_{\text{pre}} + \sigma_E \mathcal{Z}$	Encoder embeddings with added Gaussian noise (scale σ_E)
<i>Baselines from the literature</i>	
Cocos	Condition-dependent Gaussian prior [9]
BRIDGER	CVAE-based action prior [8]
Retrieval	k -NN-based action prior [39]–[41]

noise schedule, objective, and sampling procedure, which are non-trivial changes.

IV. PRIOR VARIANTS

Our study compares seven types of priors, including the default Gaussian $\mathcal{Z}$, *three we propose* (built directly from the pretrained LBM with no auxiliary training), and *three baselines* from the literature. Table I summarizes the priors; Appendix A gives the exact sampling procedures.

Proposed priors. Our three proposed priors are constructed on the fly from π_{pre} , inheriting the structure the pretrained policy has already learned.

- A_{pre} (**pretrained actions**) is a partially denoised action chunk from a frozen copy of π_{pre} run for half of its usual denoising budget (“half” splits the denoising budget evenly between π_{pre} and the fine-tuned action head to avoid extra computational costs). The pretrained model already roughly outputs plausible actions, so the velocity field only needs to cover a shorter transport distance to reach the target (see Fig. 2 left against $\mathcal{Z}$).
- $A_{\text{pre}} + \sigma_A \mathcal{Z}$ (**noise-perturbed actions**) is A_{pre} with added isotropic Gaussian jitter at action-space scale σ_A . Unperturbed A_{pre} can collapse the prior too tightly around certain modes of the pretrained policy’s output; adding noise enriches the distribution of A_{pre} .
- $E_{\text{pre}} + \sigma_E \mathcal{Z}$ (**noise-perturbed embedding**) is the pretrained encoder output embedding e , injected with Gaussian noise at embedding-space scale σ_E *before* inputting into π_{pre} to produce a_0 . This perturbs the prior in embedding space rather than action space, exploring a different axis of prior variation.

Baselines from the literature. We include three representative external priors effective for policy learning.

- **Cocos** [9] is a condition-dependent Gaussian whose mean is produced by passing the observation embedding e through a small encoder into the action space. The encoder is jointly trained with a decoder that maps back to the embedding space of e .
- **BRIDGER** [8] is a CVAE-based action prior that maps a Gaussian latent to an action chunk, conditioned on the observation embedding e .

- **Retrieval (k -NN)** [39]–[41] is a non-parametric prior that averages the k nearest-neighbor action chunks from a frozen bank indexed by the observation embedding e . The three cited works used k -NN actions directly as policies; we repurpose this approach as a prior for fine-tuning.

Some existing non-Gaussian prior methods, such as action-to-action FM [7] and VITA [42], replace the policy network with a custom architecture, conflating the effect of the prior with architectural changes. These methods are thus incompatible with fine-tuning from pretrained weights. The seven priors above all preserve the pretrained encoder and action-head weights, so *only the prior differs across experiments*.

V. EXPERIMENTAL SETUP

In this section, we detail our experimental setup for evaluating whether and under what conditions non-Gaussian priors benefit policy fine-tuning. We cover: how we control for confounds, test for statistical significance, evaluate in simulation and on robot hardware, and define our plotting conventions.

A. Experimental Controls

Throughout the paper, we hold the task, data, architecture, optimizer, and number of fine-tuning steps identical across all prior variants within each experimental setting (see Appendix C for hyperparameter choices). This protocol applies uniformly to all settings. For prior noise levels, we select σ_A and σ_E via grid search (Fig. 19; values in Appendix A).

B. Statistical Testing

We treat policy comparison as a formal hypothesis test rather than only reporting empirical average success rates, following prior work [3], [18], [43]. For binary success/failure we use the STEP statistical framework [43]; for continuous metrics (per-observation ℓ_2 distance and cosine similarity) we use Welch’s t -test [44]. All tests use a global significance level $\alpha=0.05$ (95% confidence level). When more than two priors are compared we apply Bonferroni correction: the family-wise false-positive probability of m independent pairwise tests at level α is $1-(1-\alpha)^m$. Bonferroni controls this by testing each pair at α/m such that the overall false-positive rate is upper-bounded by α . A compact letter display (CLD) [45] concisely summarizes each method comparison: two methods share at least one alphabetical letter *if and only if* their pairwise difference is not significant at the corrected level. Implementation details (STEP’s anytime-valid guarantee, Lai’s [46] fallback at per-method sample sizes above ~ 600 , and CLD ordering conventions) are in Appendix B.

C. Hardware A/B testing

Following best practices for robot policy evaluation [47], hardware rollouts use blind A/B testing: for each fixed initial condition (IC), all priors under evaluation are run in a randomized order before moving to the next IC, ensuring the human operator does not know which prior is being tested at any given trial. This design eliminates operator bias and, because priors are interleaved rather than grouped by prior, distributes any ambient drift (e.g., lighting, wear, operator attention) approximately equally across all priors.

D. Benchmarks

We use two simulators and one hardware setup. The *LBM Eval* [3] Drake-based [48] simulation benchmark serves as our main testbed. It consists of eight bimanual manipulation tasks of varying difficulty and sample sizes (approximately 196–392 per task at 100%) that were unseen during the pretraining phase for LBM 1.0 (see [3] for pretraining data selection). We also evaluate on the *RoboCasa365* [49] MuJoCo-based [50] simulation suite. This benchmark is split into 18 *atomic* tasks (single-step manipulations, e.g., opening a cabinet) and 16 *composite* tasks (multi-step sequences, e.g., preparing coffee), totaling 34 tasks. Hardware is a tabletop bimanual setup with two Franka FR3 arms and parallel grippers. Each rollout is scored by a human with a binary success label and a task-specific *partial task progress* rubric. The latter decomposes the task into ordered milestone predicates (e.g., *rotor picked up*, *rotor seated*, *locking tightened*) and reports the fraction of milestones achieved. Hardware details and per-task demonstration counts are in Appendix C.

E. Plotting Conventions

In all figures, an ($\uparrow$) arrow indicates higher is better and a ($\downarrow$) arrow lower is better. Violin plots for binary success rates show Bayesian posterior distributions; a black horizontal tick marks the posterior mean and a colored dot marks the empirical mean. Violin plots for continuous metrics (e.g., ℓ_2 distance, cosine similarity, task progress) show empirical distributions with a colored dot at the empirical mean. All plots are annotated with CLD letters: methods sharing a letter are statistically indistinguishable; methods with different letters are statistically separable.

VI. RESEARCH QUESTIONS AND RESULTS

Our experiments are guided by three research questions:

- (Q1) Do non-Gaussian priors that are measurably closer to the target improve policy fine-tuning performance?
- (Q2) Do informed priors have an advantage across data fractions, particularly in the low-data regime where the added inductive bias may be most beneficial?
- (Q3) Is the effect of prior choice consistent across model architectures as well as simulation and hardware experiments?

We design four experimental settings, summarized in Table II, to carefully study these three questions. Setting A isolates Q1 using a contrived prior that is by construction closer in ℓ_2 distance to the target action distribution than the Gaussian. Setting B_{sim} addresses both Q1 and Q2 by varying the amount of fine-tuning data across five fractions on eight tasks. Setting B_{real} tests Q3 on whether the result transfers to hardware. Setting C further studies Q3 on other LBMs by repeating the experiments on $\pi_{0.5}$ and GR00T N1.5. We consistently find that non-Gaussian priors do not improve fine-tuning, with a possible exception in very low-data regimes. Alongside each main setting we run targeted ablations (encoder learning rate, training objective, noise scale), each isolating a single confounding variable.

TABLE II
INDEX OF ALL EXPERIMENTAL SETTINGS, ABLATIONS, AND ANALYSES. EACH ROW LINKS TO ITS CORRESPONDING FIGURE(S). † REUSES THE ROLLOUTS OF ITS CORRESPONDING MAIN SETTING (NO ADDITIONAL EVALUATIONS).

Setting	Description	Model	Figures	# Rollouts
<i>Main experiments</i>				
A	Do closer-to-target priors improve fine-tuning performance?	LBM 1.0 [3]	2, 3	14000
B _{sim}	Do priors matter across varying data fractions?	LBM 1.0 [3]	4	56000
Analysis	How similar are fine-tuned models across priors?		7–10	†
Encoder LR	Does the encoder matter more than the prior?		11	9600
B _{real}	Do priors matter on robot hardware?	LBM 1.0 [3]	5	1250
Encoder LR	Does the encoder-LR effect carry over to hardware?		11	500
C	Do priors matter across LBM architectures?	$\pi_{0.5}$ [4] GR00T N1.5 [5]	6	27200 27200
<i>Appendix controls & ablations</i>				
B _{sim} : easy/hard	Does task difficulty affect prior behavior?	LBM 1.0 [3]	12	†
B _{FP}	Do priors depend on the pretraining objective?	LBM 1.0 [3]	14	7000
Analysis	Do the Section VII diagnostics hold for flow-matching-only training?		15–18	†
Noise σ	How do noise-perturbed priors scale with σ ?		19	6000
B _{real} : pilot	Which learned prior to carry forward to the full hardware evaluation (small real-robot pilot)?	LBM 1.0 [3]	20	225

A. Setting A: Fraction-Pretrained Prior

The key challenge in studying non-Gaussian priors for fine-tuning is controlling prior quality: a pretrained LBM has never seen the fine-tuning task, so its zero-shot actions may be out-of-distribution and form a weak prior. A null result would then be hard to tell apart from a poor-quality prior. Setting A therefore isolates this question by constructing an artificially strong prior that is measurably close to the target in ℓ_2 distance. We use a two-stage pipeline: first, the pretrained LBM 1.0 is fine-tuned using the Gaussian prior on 50% of the demonstrations to produce a task-aware checkpoint whose actions are much closer to the target than the Gaussian (Fig. 2, left); second, this checkpoint serves as the “pretrained model” for another round of fine-tuning, in which we vary only the prior. We evaluate on five kitchen-scenario tasks from LBM Eval (see Table VI for full names) and compare all seven priors from Table I, with 200 rollouts per task (1000 total per prior).

Results. Fig. 3 summarizes the key finding across all seven prior variants. When training from scratch (i.e., training the same LBM 1.0 architecture from random weight initialization), all non-Gaussian priors (except for Cocos; see Appendix A) outperform the Gaussian baseline $\mathcal{Z}$ based on empirical success rates, though they are largely statistically indistinguishable. The simplest of our proposed priors, A_{pre} , significantly outperforms $\mathcal{Z}$ (CLD “a” vs. “bc”) with an average success rate of roughly 28% vs. 18%. This is consistent with prior work on non-Gaussian priors in the from-scratch regime [7]–[9]. Nevertheless, this empirical advantage disappears entirely in the fine-tuning regime: A_{pre} becomes the worst-performing method based on success rate and no prior is statistically separable from the Gaussian. Additionally, the posterior distributions overlap more substantially in the fine-tuning case than in the from-scratch case, reflecting the lesser impact of action priors on policy performance during fine-tuning.

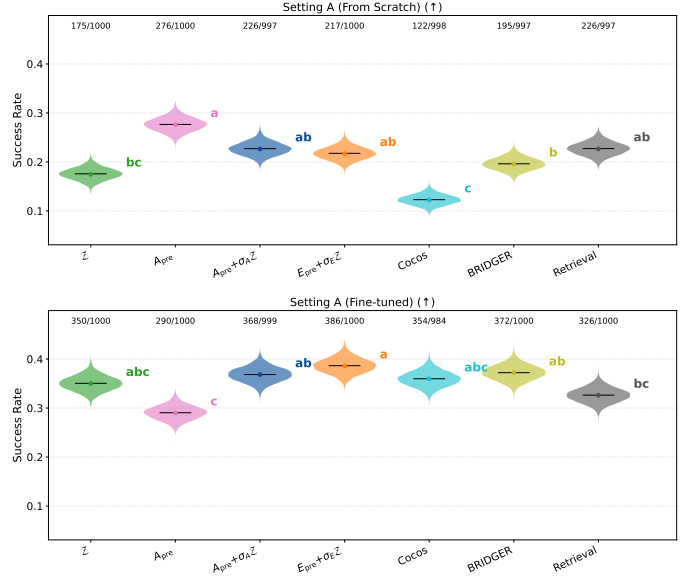


Fig. 3. Setting A: full seven-prior comparison; each violin corresponds to 1000 rollouts. **Top:** from scratch; A_{pre} statistically outperforms $\mathcal{Z}$, which is second lowest in empirical success rate among all priors. **Bottom:** fine-tuned; all priors perform comparably to $\mathcal{Z}$ with six of seven priors sharing CLD group “a” or “b”.

B. Setting B_{sim}: LBM Fine-Tuning in Simulation

Setting A tests Q1 and finds no prior advantage under a controlled good prior at small scale. Two questions remain for the practical fine-tuning regime. First, do priors matter at full LBM scale without the artificial construction in Setting A (Q1)? Second, how do learned priors perform across fine-tuning data fractions, especially in the low-data regime that may benefit from inductive biases of learned priors (Q2)? To address both questions, we fine-tune the LBM 1.0 model on the eight LBM

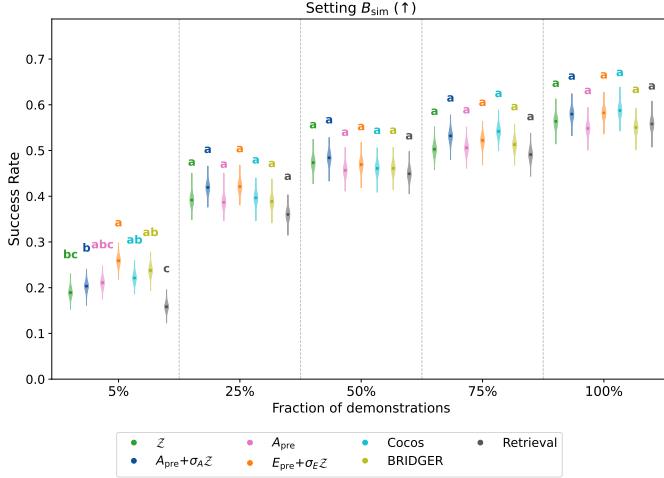


Fig. 4. Fine-tuning performance across data fractions (Setting B_{sim} , eight tasks). CLD letters are computed within each data fraction independently. All priors are statistically indistinguishable or worse than Z across fractions. The one exception is $E_{\text{pre}} + \sigma_E Z$ at 5% data, which significantly outperforms Z .

Eval unseen tasks and sweep over multiple data fractions (5%, 25%, 50%, 75%, 100% of available demonstrations). This mirrors the standard LBM deployment pipeline [3], [18]. Of the eight tasks, three are comparatively easier (higher success rates under the standard Gaussian) and five are harder; we report both the overall average and the per-task breakdown (Appendix D). Each of the seven priors is evaluated with 200 rollouts per prior–task–fraction combination, totaling 56,000 simulation rollouts for this setting.

Results. Fig. 4 shows the success rates averaged across the eight simulation tasks as a function of data fraction. At every data fraction from 25% upward, most prior variants cluster closely and all share the CLD letter “a”. The one exception is at 5% data, where $E_{\text{pre}} + \sigma_E Z$ outperforms Z (CLD “a” vs. “bc,” 25.9% vs. 18.9% success rate). Intuitively, at such a low data fraction the demonstrations cover only a small portion of the target action space, and the embedding-space jitter from $E_{\text{pre}} + \sigma_E Z$ appears to act as an informative inductive bias that the Gaussian cannot provide. This very low-data regime (approximately 10–20 demonstrations per task) is the one exception in our study where a learned prior helps.

C. Setting B_{real} : LBM Fine-Tuning on Robot Hardware

Settings A and B_{sim} both rely on simulation, which could in principle mask simulation-specific artifacts. The hardware setting tests whether the finding holds under real-world conditions that cannot be replicated in simulation. We fine-tune LBM 1.0 on five bimanual FR3 hardware tasks, spanning varying difficulty levels: *FoodBank*, *ClearKitchenCounter* [3], *GatheringIngredientsForCookies*, *BusBin*, and *BikeRotorInstall* [3]. Per-task descriptions and demonstration counts are in Appendix C. We compare five of the seven prior variants: Z (the standard Gaussian baseline), $E_{\text{pre}} + \sigma_E Z$, Cocos, BRIDGER¹, and Retrieval, with 50 A/B-test rollouts per prior

¹Its CVAE training was unstable on both simulation and hardware datasets (NaN losses, requiring restarts with different random seeds until stable); despite this instability, stable runs of BRIDGER were included in the evaluation.

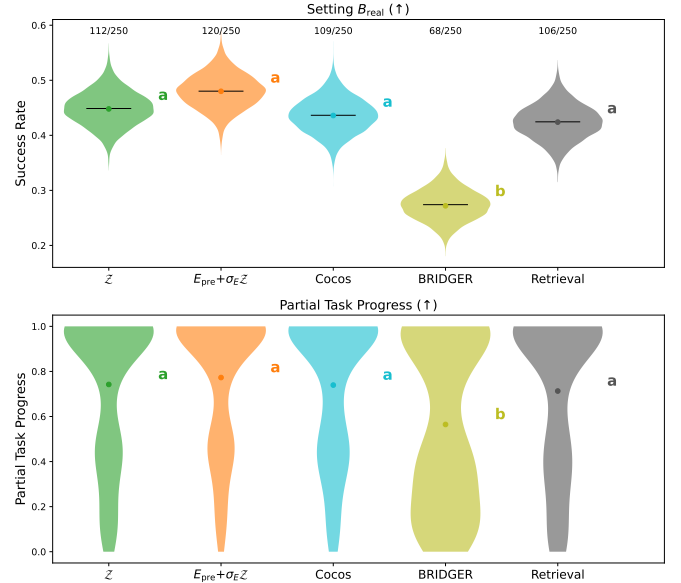


Fig. 5. Setting B_{real} : hardware A/B testing (five bimanual FR3 tasks, 50 rollouts per prior per task, 1250 total). **Top:** Success rate violins; four of five priors share CLD group “a” and BRIDGER (group “b”) underperforms. **Bottom:** Partial-task-progress violins; the same grouping holds.

per task (250 per prior, 1250 total; protocol in Section V). The remaining two priors, A_{pre} and $A_{\text{pre}} + \sigma_A Z$, were dropped after a pilot study (Appendix D-F) in which all three action-space priors were statistically tied and $E_{\text{pre}} + \sigma_E Z$ was selected as the representative learned prior for the full evaluation.

Results. Fig. 5 shows full results across the five bimanual FR3 hardware tasks. Out of 250 A/B-test rollouts per prior, four of the five priors (i.e., Z , $E_{\text{pre}} + \sigma_E Z$, Cocos, and Retrieval) fall in the same CLD group “a,” with partial-progress scores in the 71–77% range. BRIDGER is the sole outlier (group “b,” partial progress $\sim 56\%$), consistent with the training instability noted earlier and documented in Appendix A. Excluding BRIDGER, the pattern mirrors our simulation findings: the choice of prior has no statistically significant effect on fine-tuning performance.

D. Setting C: Validation on Other LBM Architectures

Settings A, B_{sim} , and B_{real} all use LBM 1.0 as the backbone, which uses a CLIP encoder and DiT action head. Setting C tests whether the results generalize across model architectures and simulation environments. We repeat the comparison on the RoboCasa simulation benchmark using two additional LBMs: $\pi_{0.5}$ [4] (fused SigLIP + Gemma 2B with interleaved cross-attention) and GR00T N1.5 [5] (Eagle-2 VLM + diffusion transformer action expert).

Each model is fine-tuned on each of the 34 RoboCasa tasks and evaluated with 200 rollouts per task (see Appendix D-H for per-task breakdowns). The fused architecture constrains which priors can be ported from the LBM 1.0 experiments. Of the seven priors from Setting B_{sim} , four have natural analogues here: Z (Gaussian), A_{pre} (pretrained actions), $A_{\text{pre}} + \sigma_A Z$ (action + noise), and Retrieval (k -NN in the LBM’s vision-encoder feature space, $k=3$). We construct A_{pre} for each model

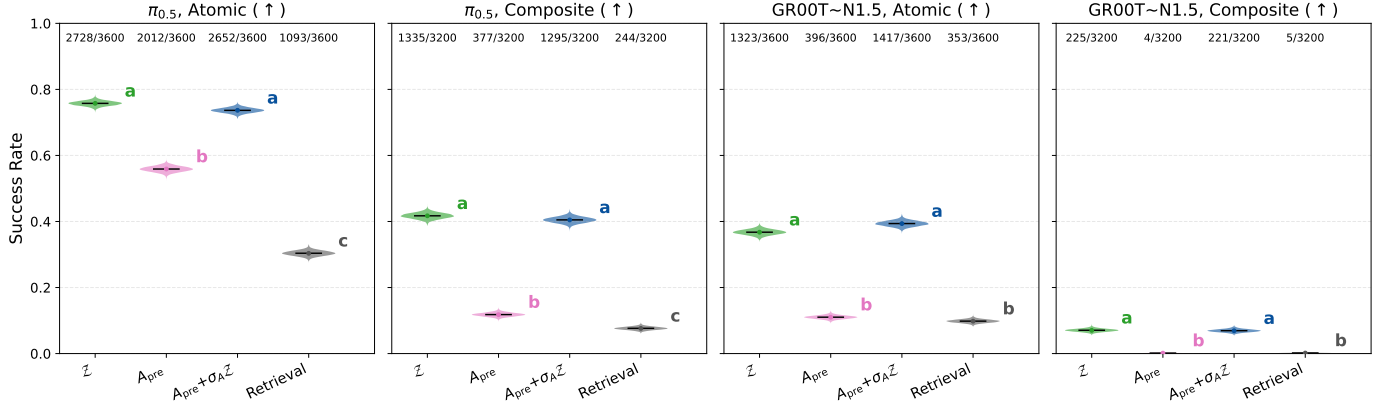


Fig. 6. Setting C: LBM fine-tuning on RoboCasa, with shared y-axis (Success Rate, 0–1) across all four panels. **Left two panels:** $\pi_{0.5}$ on 18 atomic (3600 rollouts per prior) and 16 composite (3200 rollouts per prior) tasks. **Right two panels:** GR00T N1.5 on the same splits. Both architectures share the same qualitative ordering; $\mathcal{Z}$ and $A_{\text{pre}} + \sigma_A \mathcal{Z}$ tie at the top on both task types, while A_{pre} and Retrieval priors underperform.

from a frozen copy of itself (e.g., $\pi_{0.5}$ ’s A_{pre} uses frozen $\pi_{0.5}$). The embedding-space priors $E_{\text{pre}} + \sigma_E \mathcal{Z}$, Cocos, and BRIDGER are not immediately portable because both LBMs lack a single bottleneck embedding: visual and action tokens are fused across many transformer layers with per-layer cross-attention, leaving no natural injection point. They are thus omitted in this setting.

Results. Fig. 6 reports success rates for atomic and composite tasks separately for each of the two LBMs. Both architectures exhibit the same qualitative ordering across task types: $\mathcal{Z}$ and $A_{\text{pre}} + \sigma_A \mathcal{Z}$ tie at the top in CLD group “a,” while the pure learned prior A_{pre} falls into group “b”; Retrieval is either “b” or in a distinct group “c.” Despite the observed performance degradation of A_{pre} and Retrieval, A_{pre} remains $\sim 3\times$ closer to the target than $\mathcal{Z}$ in ℓ_2 on RoboCasa (Fig. 21), the same factor as in Setting A on LBM Eval (Fig. 2, left). Yet A_{pre} degrades only here in Setting C, indicating that geometric proximity to the target is not what drives the gap. A key difference in this setting compared to the earlier ones is that LBM 1.0’s pretraining includes simulation data from the same simulator, whereas $\pi_{0.5}$ and GR00T N1.5 were pretrained predominantly on real-world data. This domain gap specifically hurts A_{pre} and Retrieval because both rely solely on a frozen copy of the pretrained LBM (for action generation and for building the retrieval bank, respectively). We hypothesize that the other two priors, $\mathcal{Z}$ and $A_{\text{pre}} + \sigma_A \mathcal{Z}$, are regularized by random noise and thus are less affected by this domain gap.

VII. ANALYSIS: WHY DOES THE PRIOR NOT MATTER UNDER FINE-TUNING?

Summarizing the negative-result experiments against our research questions, we found that Q1 (closer priors should help) yields a negative answer, Q2 (advantage across fine-tuning data fractions) is only partly supported at 5% data, and Q3 (generalization across policies and hardware) also yields a negative answer. The experimental results thus raise a central question: if a non-Gaussian prior is measurably closer to the target distribution (in Setting A by construction, and in Settings B and C empirically), why does this proximity not translate to better fine-tuning performance?

We approach this question from two complementary perspectives. First, we ask whether fine-tuned models *converge toward the target* despite starting from different priors. We examine convergence at two levels: whether different priors produce encoder features with comparable predictive power for the target actions (Section VII-A), and whether the resulting action predictions are similarly close to ground truth (Section VII-B). Second, we ask whether fine-tuned models *diverge from a shared starting point*: that is, whether different priors lead to structurally different optimization trajectories, as measured by ℓ_2 distance and cosine similarity in the action or embedding space (Section VII-C). The observed embedding-space divergence motivates a natural hypothesis: fine-tuning performance is governed primarily by the observation encoder rather than by the prior itself. We test this hypothesis directly through a learning-rate ablation that deliberately undertrains the encoder (Section VII-D).

All analyses in this section are carried out in Setting B_{sim} (LBM 1.0 simulation), where we have the most comprehensive results: seven prior variants $\times$ five data fractions $\times$ eight tasks, enabling controlled comparisons across the prior–data landscape. We focus on four representative priors: $\mathcal{Z}$ (Gaussian), A_{pre} (pure learned prior, closest to target), $A_{\text{pre}} + \sigma_A \mathcal{Z}$ (action + noise; defined in Section IV), and $E_{\text{pre}} + \sigma_E \mathcal{Z}$ (embedding + noise). The baselines (Cocos, BRIDGER, Retrieval) are omitted from these introspective ablations as their performance was found to be worse than or comparable to these four priors. Action-space metrics are computed on the flattened action chunk. Embedding-space metrics are computed on the observation embedding e defined in Section III. For notation, we reuse A_{pre} from Section IV (actions drawn from the frozen pretrained policy) and introduce A_{post} for the corresponding actions drawn from the fine-tuned policy. The observation embedding e (Section III) has two instances depending on the encoder checkpoint: $E_{\text{pre}} = e_{\pi_{\text{pre}}}(o)$ is the embedding from the pretrained encoder, and $E_{\text{post}} = e_{\pi_{\text{post}}}(o)$ is the embedding from the fine-tuned encoder, evaluated on the same o .

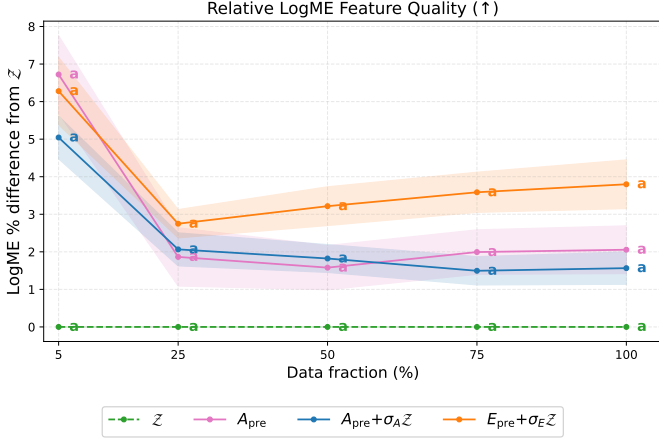


Fig. 7. LogME feature quality as percentage difference from $\mathcal{Z}$ at each fraction (Setting B_{sim} , eight tasks). A value of five means LogME for that prior is 5% higher than $\mathcal{Z}$ (dashed green line at zero) at the same fraction. Ribbons show ± 1 standard error across the eight tasks. We normalize because raw LogME drops sharply from 5% to 25% data due to a small-sample artifact of the LogME computation (see raw values in Fig. 13); normalizing to $\mathcal{Z}$ removes this artifact since we only compare across priors at the same sample size. CLD letters (computed on raw values) confirm all priors share group “a” at every fraction.

A. Feature Quality: LogME Analysis

We begin at the feature layer, asking whether fine-tuned observation encoders produce features of equal predictive power for the target actions, regardless of the prior used during training. We measure this with LogME [24], a training-free proxy for feature quality that estimates the log marginal evidence of a Bayesian linear regression from encoder features E_{post} to ground-truth actions A_{target} . LogME produces a single aggregate score per prior–task–fraction combination by fitting a weight matrix to all (observation-embedding, action) pairs. Higher LogME indicates more predictive encoder features for action prediction. As shown in Fig. 7, LogME scores are inseparable across all priors. The prior thus governs how much the encoder restructures, but not the quality of the final result: all priors arrive at features that are equally predictive of actions (Section VII-A).

B. Action-Space Similarity After Fine-Tuning

Section VII-A established convergence at the feature layer: fine-tuned observation encoders are equally predictive of actions across priors. We now examine convergence at the output layer, asking whether the post-fine-tuning predicted actions A_{post} converge to the ground-truth actions A_{target} in the target demonstration data. Quantitatively, we measure the per-observation ℓ_2 distance (computed over random samples in the demonstration data; Fig. 8) and the per-observation cosine similarity (Fig. 8). The two metrics probe complementary aspects of the action discrepancy: ℓ_2 captures both magnitude and directional errors, while cosine similarity is scale-invariant and isolates directional agreement. We also note that the per-prior sample size in these distributions is very large ($n \approx 16000$ per prior per fraction), so Welch’s t -test with Bonferroni correction can detect extremely small mean differences. CLD

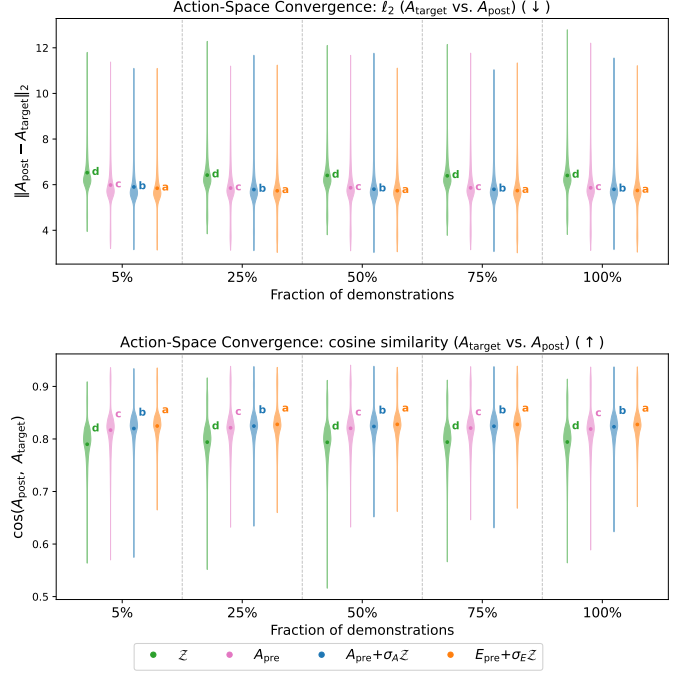


Fig. 8. Action-space similarity after fine-tuning (Setting B_{sim} , eight tasks). Violins show per-observation distances between predicted and ground-truth actions at each fraction, with CLD letters from Welch’s t -test. **Top:** ℓ_2 distance ($\downarrow$). All four priors cluster within a narrow range (5.7–6.4), with the Gaussian consistently ~ 0.5 higher than the non-Gaussian priors. **Bottom:** cosine similarity ($\uparrow$). All four priors largely overlap at every fraction (means within ~ 0.79 – 0.83). Note: per-prior sample sizes are very large ($n \approx 16000$), so CLD letters here flag any detectable mean difference and may not reflect practical separation.

letters here should therefore be interpreted as indicators of *any* detectable mean difference, not as indicators of practically significant separation.

The post-fine-tuning action predictions are broadly similar across priors under both metrics, with violin distributions that overlap substantially at every fraction. Under ℓ_2 (Fig. 8), the Gaussian prior ($\mathcal{Z}$) consistently shows a slightly higher (worse) mean distance (~ 6.4) than the non-Gaussian priors (~ 5.7 – 5.9), with the absolute gap remaining below 0.8 at every data fraction. Under cosine similarity (Fig. 8), the four priors still have largely overlapping violins, with means within ~ 0.79 – 0.83 at every fraction; again, $\mathcal{Z}$ is consistently slightly worse (lower cosine similarity). Note that this small open-loop gap in both metrics does not translate into closed-loop success rate differences (Fig. 4, except against $E_{\text{pre}} + \sigma_E \mathcal{Z}$ at 5%), suggesting it is within the noise tolerance of the downstream rollout evaluation. These action-space results confirm that post-fine-tuning predictions are similar across priors, consistent with the equal feature quality reported in Section VII-A.

C. Representation Divergence Despite Similar Outputs

The preceding two analyses together characterize convergence *toward the target*: fine-tuned observation encoders produce features of indistinguishable quality for action prediction (Section VII-A), and their action predictions reach the ground truth with broadly comparable accuracy (Section VII-B). We

now shift perspective and measure *divergence from the pre-trained starting point*: how much do the model’s representations change during fine-tuning, and does the amount of change depend on the prior? If different priors induce different amounts of change, this would indicate that prior choices affect the optimization landscape yet admit multiple equally good solutions as judged by closed-loop performance. We quantify this also at two levels (action-space and embedding-space) using two complementary metrics: the per-observation ℓ_2 distance between pre- and post-fine-tuning outputs (Fig. 9) and the corresponding per-observation cosine similarity (Fig. 10). Note that absolute ℓ_2 values are not directly comparable across the two panels because the action and embedding spaces have different dimensionalities; cosine similarity is scale-free and enables direct comparison.

Action-space change. Across all priors, the post-fine-tuning actions remain close to their pretrained counterparts: both ℓ_2 metrics (~ 2.5 – 2.9) and cosine similarity metrics (~ 0.92 – 0.95) span a narrow band (top figures of Fig. 9 and Fig. 10). The takeaway is that fine-tuning makes comparable adjustments to the action output across priors, though A_{pre} shifts slightly less (smaller ℓ_2 metrics and higher cosine similarity) from the pretrained starting point than the other priors.

Embedding-space change. In the observation embedding space (E_{post} vs. E_{pre}), however, the priors diverge sharply, and the CLD analysis reveals statistically significant separation at every data fraction under both ℓ_2 (Fig. 9 bottom) and cosine similarity (Fig. 10 bottom). The Gaussian prior shows the smallest change (CLD group “a”), meaning the encoder barely moves from its pretrained state. The learned prior A_{pre} , however, induces the largest restructuring (group “d”), roughly 2 to 2.5 times the Gaussian’s ℓ_2 change and the largest drop in cosine similarity. The mixed priors $A_{\text{pre}} + \sigma_A \mathcal{Z}$ and $E_{\text{pre}} + \sigma_E \mathcal{Z}$ fall between these extremes, each occupying a distinct CLD group under both metrics.

Practical significance. The action-space distributions (Fig. 9 top) also show statistically distinct CLD groups, but their empirical distributions overlap almost entirely—the detected differences are too small to be practically meaningful. In contrast, the embedding-space separability corresponds to large, visually apparent gaps between the violin distributions, particularly at lower data fractions, and the same prior ranking holds under both ℓ_2 and cosine metrics. We speculate one possible explanation for the large embedding-space changes: because the Gaussian was used during pretraining, fine-tuning with $\mathcal{Z}$ only shifts the target action distribution toward the fine-tuning task data while the prior remains unchanged. Fine-tuning with a pretrained action prior A_{pre} , in contrast, shifts both the prior and the target relative to the pretrained policy, forcing the encoder to compensate more. Mixed priors that combine noise with pretrained actions dilute this shift and produce intermediate encoder changes.

The prior thus governs how much the encoder restructures, but not the quality of the final result: all priors arrive at features that are equally predictive of actions (Section VII-A). This view resonates with previous loss landscape analyses [19], [20] on vision transfer learning, where different encoder configurations could lead to equivalent downstream performance.

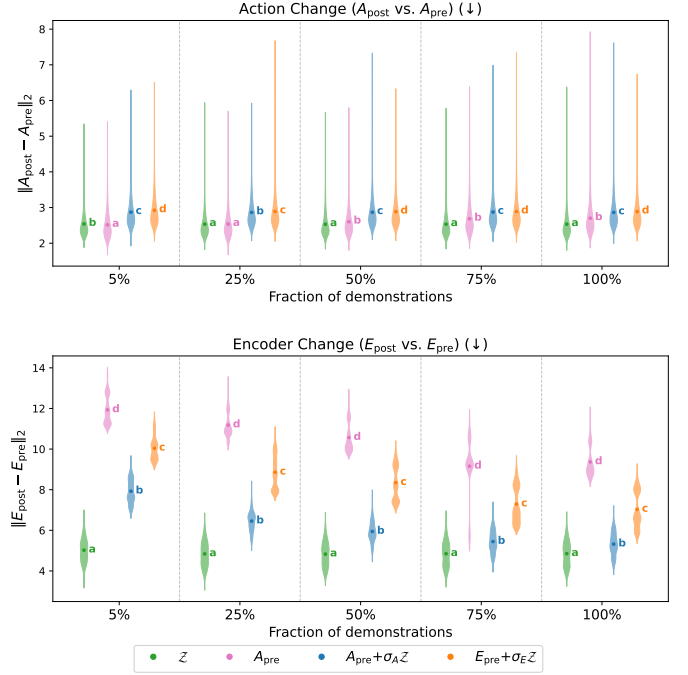


Fig. 9. How much does fine-tuning change the model’s internal representations? (Setting B_{sim} , eight tasks.) Violins show per-observation ℓ_2 distances with CLD letters from Welch’s t -test at each fraction. **Top:** action change (A_{post} vs. A_{pre}); all priors cluster within a narrow band (2.5–2.9). **Bottom:** encoder change (E_{post} vs. E_{pre}); the four priors show large, visually apparent separation between violin distributions at every fraction (unlike the action change above, where the four CLD groups arise from small mean differences within an otherwise largely overlapping band), with A_{pre} restructuring the encoder substantially (mean ℓ_2 roughly 9–12 across fractions) while $\mathcal{Z}$ moves much less ($\ell_2 \approx 5$).

In our case, it is likely that the prior determines which local minimum the optimizer settles into, but these local minima are comparably effective downstream. Whether this divergence in encoder representations has other implications for transfer learning, beyond the downstream success rate we measure here, remains to be explored.

D. Encoder Training Matters More Than Prior Choice

The previous subsection shows that with respect to the pretrained policy, action-space changes after fine-tuning are relatively small (significant overlap in the violins), whereas the observation encoder restructures substantially across priors. This naturally raises the question: how much does encoder training quality matter relative to prior choice? If the encoder dominates and the prior is secondary, then degrading the encoder should hurt performance far more than changing the prior. We test this by reducing the observation encoder’s learning rate to $1/10$ of the action head’s, deliberately undertraining it while keeping all other hyperparameters identical.

In simulation, we run this ablation on the LBM 1.0 model with three representative priors— $\mathcal{Z}$ (Gaussian baseline), $E_{\text{pre}} + \sigma_E \mathcal{Z}$ (best-performing among our proposed priors), and Cocos (most recent external baseline)—across all eight Setting B_{sim} tasks at 100% data (Fig. 11). Reducing the encoder learning rate consistently and significantly degrades performance across all three priors: success rates drop from the 56–59% range to the 41–43% range. Statistical tests confirm

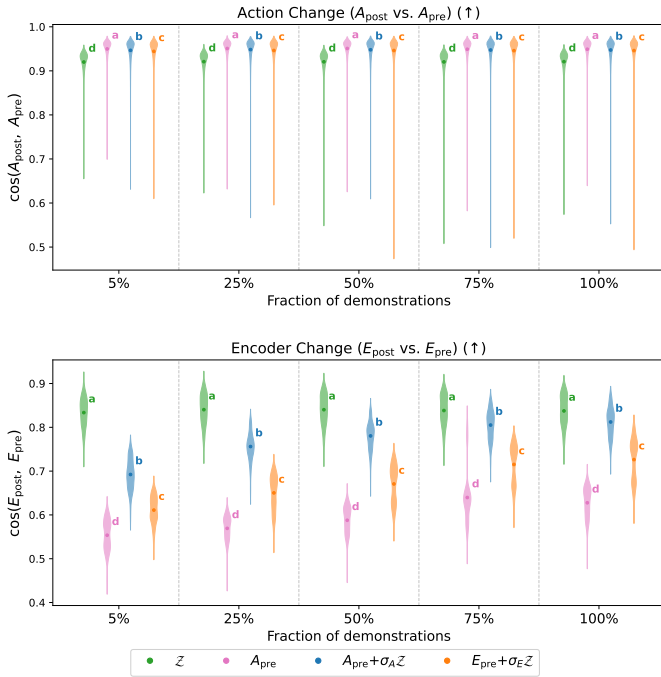


Fig. 10. Representation change, cosine-similarity view (Setting B_{sim} , eight tasks); companion to Fig. 9. Higher values indicate less directional rotation from the pretrained representation. **Top:** action change; all priors largely overlap ($\cos \approx 0.92\text{--}0.95$). **Bottom:** encoder change; Z retains the highest similarity ($\cos \approx 0.84$) while A_{pre} rotates the most ($\cos \approx 0.55\text{--}0.64$), with the mixed priors in between.

significant degradation for all three priors (reduced-LR variants share CLD group “b”; standard-LR variants share group “a”). Moreover, the magnitude of degradation is comparable across priors, which is exactly the expected result if the encoder, not the prior, is the dominant factor.

On hardware (Setting B_{real}), we validate this finding with the Gaussian prior across five tasks (Fig. 11). The standard encoder LR achieves 37% success rate, while the reduced 1/10 LR drops to 28%, a statistically significant decrease (different CLD letters). This hardware result confirms the simulation finding: undertraining the encoder has a far larger impact on policy quality than any prior-related change we study.

Overall, observation-encoder training quality is the dominant factor in fine-tuning performance: within each encoder-LR regime, the choice of action-space prior makes no measurable difference, while changing the encoder’s learning rate shifts performance by an amount that exceeds any prior-related effect. This resonates with recent work on run-time monitoring [51], [52], which found that policy encoder embeddings carry rich information predictive of downstream task success. Our encoder-LR ablation supports the view that the quality of these embeddings is a critical lever for policy performance.

VIII. DISCUSSION

A. Implications for Practitioners

In practice, the standard Gaussian prior is sufficient for fine-tuning LBM with a flow-matching action head. Implementing learned priors adds engineering complexity with no measurable benefit. Our encoder-LR ablation (Section VII-D)

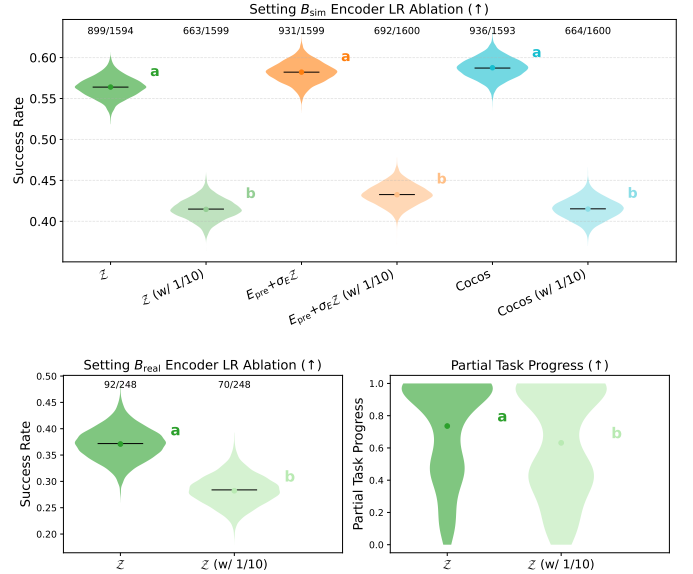


Fig. 11. Encoder learning rate ablation. **Top:** Setting B_{sim} (eight tasks, 100% data, three representative priors). Reducing the encoder LR to 1/10 of the action head’s (lighter violins, “b”) degrades performance by 15–17% across all three priors, while the choice of prior has minimal effect within each LR group (both groups share a single CLD letter). **Bottom:** Setting B_{real} (Gaussian prior, five tasks). Reducing the encoder LR to 1/10 drops both success rate and partial task progress (different CLD groups), a larger effect than any prior-related difference in this study.

suggests that the key factor appears to be not the prior itself but the quality of the pretrained representations: once the encoder is sufficiently well-pretrained, the encoder dominates the transport regardless of where the flow starts. On the other hand, in the from-scratch regime without pretrained models, learned priors may still be valuable, improving success rates even though differences could be statistically indistinguishable, as supported by both our results and prior work [7]–[10].

B. Limitations

Our findings are established on three LBMs (LBM 1.0, $\pi_{0.5}$, and GR00T N1.5). The prior-independence finding holds consistently across all three, suggesting it is not an artifact of a particular model capacity or architecture. Nevertheless, we cannot rule out that much larger or smaller models, or models trained on special pretraining distributions, might behave differently. Similarly, our fine-tuning datasets are relatively small (20–400 demonstrations per task); whether the prior-independence finding persists under substantially larger or more diverse fine-tuning data remains open. Our proposed explanation, that the pretrained encoder dominates the action prior choices, is plausible at other scales but remains unverified. Although the metrics we use (LogME, per-observation ℓ_2 , and cosine similarity) are widely adopted in the transfer learning literature [19], [24], alternative metrics such as LEEP [53] or H-Score [54] could reveal additional aspects of how priors affect LBM fine-tuning that we have not explored.

C. Future Work

We point out three concrete research directions:

- 1) **Where is the transition?** At 5% data in Setting B_{sim} (approximately 20 demonstrations per task), $E_{\text{pre}} + \sigma_E \mathcal{Z}$ shows significant improvements over $\mathcal{Z}$. This hints at a regime where the prior begins to matter, but the boundary is not clearly characterized. Mapping this transition as a function of dataset size, task complexity, and model capacity would let practitioners decide when learned priors are worth the engineering cost.
- 2) **Why do different priors yield divergent encoder representations of equal quality?** Our analysis shows that they do, but not why the loss landscape admits multiple equally good encoder configurations. Loss-landscape geometry, plasticity [55], and linear mode connectivity [56] are all plausible directions worth investigating.
- 3) **Does encoder divergence have downstream consequences?** Different priors lead to structurally different encoders with indistinguishable nominal policy performance. Whether these solutions differ in robustness to distribution shift, continual-learning stability, or catastrophic forgetting is an open question with practical implications.

IX. CONCLUSION

We presented a large-scale empirical study of non-Gaussian priors under LBM fine-tuning for visuomotor imitation learning. Across 100K+ cross-architecture simulation rollouts on 40+ tasks in two simulation platforms, and 1250 hardware rollouts, the evidence consistently points in the same direction: learned priors that help when training from scratch do not improve fine-tuning performance. Diagnostic analyses reveal that regardless of prior choice, fine-tuned encoders yield indistinguishable conditional action likelihoods, and the resulting policy outputs similar actions. However, their encoder representations diverge from the pretrained model and each other. An encoder-LR ablation further confirms that the observation encoder, not the prior, is the dominant factor. These findings corroborate that the standard Gaussian prior used in current LBM fine-tuning pipelines is a well-supported choice. More broadly, given these negative results, we hope to inspire the community to explore the underlying learning dynamics of large-scale visuomotor architectures, focusing on why certain components (such as the observation encoder) matter more than others and how that understanding can improve different aspects of downstream robot policy performance.

ACKNOWLEDGMENT

The authors thank Paarth Shah, Eric Cousineau, Aykut Onol, and Katherine Liu for helpful discussions and feedback, and Patrick Tree Miller and all the robot teachers (Rudy Bravo, Emma Dixon, Fredy Navas, Christopher Rodriguez, Derick Seale) for assistance with the hardware experiments.

REFERENCES

- [1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” in *Proc. Robotics: Science and Systems (RSS)*, 2023.
- [2] M. Reuss, M. Li, X. Jia, and R. Lioutikov, “Goal-conditioned imitation learning using score-based diffusion policies,” in *Proc. Robotics: Science and Systems (RSS)*, 2023.
- [3] J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina *et al.*, “A careful examination of large behavior models for multitask dexterous manipulation,” *Science Robotics*, 2026.
- [4] Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky, “ $\pi_{0.5}$: A vision-language-action model with open-world generalization,” *arXiv:2504.16054*, 2025.
- [5] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, “Gr00t n1: An open foundation model for generalist humanoid robots,” *arXiv:2503.14734*, 2025.
- [6] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” in *Proc. International Conference on Learning Representations (ICLR)*, 2023.
- [7] J. Jia, G. Li, X. Chen, T. An, Y. Hu, J. Li, X. Guo, and J. Yang, “Action-to-action flow matching,” in *Proc. Robotics: Science and Systems (RSS)*, 2026.
- [8] K. Chen, E. Lim, K. Lin, Y. Chen, and H. Soh, “Don’t start from scratch: Behavioral refinement via interpolant-based policy diffusion,” in *Proc. Robotics: Science and Systems (RSS)*, 2024.
- [9] Z. Dong, Y. Liu, Y. Li, H. Zhao, and J. Hao, “Conditioning matters: Training diffusion policies is faster than you think,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2026.
- [10] J. Chang, R. Mei, W. Ke, and X. Xu, “Efficientflow: Efficient equivariant flow policy learning for embodied ai,” in *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2026.
- [11] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, “LIBERO: Benchmarking knowledge transfer for lifelong robot learning,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [12] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning,” in *Proc. Conference on Robot Learning (CoRL)*, 2020.
- [13] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, “Learning complex dexterous manipulation with deep reinforcement learning and demonstrations,” in *Proc. Robotics: Science and Systems (RSS)*, 2018.
- [14] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, “ π_0 : A vision-language-action flow model for general robot control,” in *Proc. Robotics: Science and Systems (RSS)*, 2025.
- [15] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [16] Y. Feng, J. Zheng, Z. Wang, D. Liu, J. Li, J. Pang, T. Wang, and X. Zhan, “Demystifying action space design for robotic manipulation policies,” in *Proc. International Conference on Learning Representations (ICLR)*, 2026.
- [17] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao, “Data scaling laws in imitation learning for robotic manipulation,” in *Proc. International Conference on Learning Representations (ICLR)*, 2025.
- [18] F. Lin, K. Arora, J. Mercat, H. Nishimura, P. Shah, C. Xu, M. Zhang, M. Zolotas, M. Angeles, O. Pfannenstiehl, A. Beaulieu, and J. Barreiros, “A systematic study of data modalities and strategies for co-training large behavior models for robot manipulation,” in *Proc. Robotics: Science and Systems (RSS)*, 2026.
- [19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2014.
- [20] K. He, R. Girshick, and P. Dollár, “Rethinking ImageNet pre-training,” in *Proc. International Conference on Computer Vision (ICCV)*, 2019.

- [21] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell, "Real-world robot learning with masked visual pre-training," in *Proc. Conference on Robot Learning (CoRL)*, 2022.
- [22] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3M: A universal visual representation for robot manipulation," in *Proc. Conference on Robot Learning (CoRL)*, 2022.
- [23] L. Wang, X. Chen, J. Zhao, and K. He, "Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- [24] K. You, Y. Liu, J. Wang, and M. Long, "LogME: Practical assessment of pre-trained models for transfer learning," in *Proc. International Conference on Machine Learning (ICML)*, 2021.
- [25] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv:2001.08361*, 2020.
- [26] R. Schaeffer, B. Miranda, and S. Koyejo, "Are emergent abilities of large language models a mirage?" in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [27] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.
- [28] C. Chiang, K.-H. Tu, J.-W. Liao, C.-F. Chou, and T.-W. Ke, "COT-FM: Cluster-wise optimal transport flow matching," in *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [29] E. Su, T. Westenbroek, A. Nagabandi, and A. Gupta, "RFS: Reinforcement learning with residual flow steering for dexterous manipulation," in *Proc. International Conference on Learning Representations (ICLR)*, 2026.
- [30] Z. Liu, M. Pan, Z. Wang, K. Zhu, H. Lu, H. Zhang, J. Wang, and Y. Shi, "Sample from what you see: Visuomotor policy learning via diffusion bridge with observation-embedded stochastic differential equation," *arXiv:2512.07212*, 2025.
- [31] K. Nguyen and M. N. Vu, "XFlowMP: Task-conditioned motion fields for generative robot planning with schrödinger bridges," *arXiv:2512.00022*, 2025.
- [32] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, "Steering your diffusion policy with latent space reinforcement learning," in *Proc. Conference on Robot Learning (CoRL)*, 2025.
- [33] O. Patil, O. Biza, T. Weng, K. Schmeckpeper, W. Thomason, X. Zhang, R. Walters, N. Gopalan, S. Castro, and E. Rosen, "You've got a golden ticket: Improving generative robot policies with a single noise vector," *arXiv:2603.15757*, 2026.
- [34] H. Zhang, Y. Sun, and H. Xiong, "Goldenstart: Q-guided priors and entropy control for distilling flow policies," in *Proc. International Conference on Learning Representations (ICLR)*, 2026.
- [35] J. Li, Y. Cong, Y. Wang, H. Xia, S. Huang, Y. Zhang, N. Xu, and G. Dai, "STEP: Warm-started visuomotor policies with spatiotemporal consistency prediction," *arXiv:2602.08245*, 2026.
- [36] S. Jiang, X. Fang, N. Roy, T. Lozano-Pérez, L. P. Kaelbling, and S. Ancha, "Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories," in *Proc. Conference on Robot Learning (CoRL)*, 2025.
- [37] Z. Li, R. Yang, R. Chen, Z. Luo, and L. Chen, "ADPro: A test-time adaptive diffusion policy via manifold-constrained denoising and task-aware initialization for robotic manipulation," *arXiv:2508.06266*, 2025.
- [38] S. Odonchimed, T. Matsushima, S. Holk, Y. Iwasawa, and Y. Matsuo, "Retrieve-augmented generation for speeding up diffusion policy without additional training," *arXiv:2507.21452*, 2025.
- [39] J. Pari, N. M. M. Shafiullah, S. P. Arunachalam, and L. Pinto, "The surprising effectiveness of representation learning for visual imitation," in *Proc. Robotics: Science and Systems (RSS)*, 2022.
- [40] C. He, X. Liu, G. M. Szaier Camps, J. Bruno, G. A. Sartoretti, and M. Schwager, "Demystifying robot diffusion policies: Action memorization and a simple lookup table alternative," in *Proc. International Conference on Learning Representations (ICLR)*, 2026.
- [41] K. Chen, C. Li, C. Tu, J. Pan, Y. Ma, W. Chen, Z. Zhou, X. Xu, S. James, C.-W. Fu, R. Xiong, P. Abbeel, Y.-H. Liu, and Q. Dou, "A retrieval-augmented framework enabling vlm spatial awareness for object-centric robot manipulation," *Science Robotics*, 2026.
- [42] D. Gao, B. Zhao, A. Lee, I. Chuang, H. Zhou, H. Wang, Z. Zhao, J. Zhang, and I. Soltani, "VITA: Vision-to-action flow matching policy," in *Proc. International Conference on Learning Representations (ICLR)*, 2026.
- [43] D. Snyder, A. J. Hancock, A. Badithela, E. Dixon, P. Miller, R. A. Ambrus, A. Majumdar, M. Itkina, and H. Nishimura, "Is your imitation learning policy better than mine? policy comparison with near-optimal stopping," in *Proc. Robotics: Science and Systems (RSS)*, 2025.
- [44] B. L. Welch, "The generalization of 'Student's' problem when several different population variances are involved," *Biometrika*, 1947.
- [45] H.-P. Piepho, "An algorithm for a letter-based representation of all-pairwise comparisons," *Journal of Computational and Graphical Statistics*, 2004.
- [46] T. L. Lai, "Nearly optimal sequential tests of composite hypotheses," *The Annals of Statistics*, 1988.
- [47] H. Kress-Gazit, K. Hashimoto, N. Kuppaswamy, P. Shah, P. Horgan, G. Richardson, S. Feng, and B. Burchfiel, "Robot learning as an empirical science: Best practices for policy evaluation," *arXiv:2409.09491*, 2024.
- [48] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," <https://drake.mit.edu>, 2019.
- [49] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu, "RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots," in *Proc. International Conference on Learning Representations (ICLR)*, 2026.
- [50] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- [51] C. Xu, T. K. Nguyen, E. Dixon, C. Rodriguez, P. Miller, R. Lee, P. Shah, R. A. Ambrus, H. Nishimura, and M. Itkina, "Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies," in *Proc. Robotics: Science and Systems (RSS)*, 2025.
- [52] Q. Gu, Y. Ju, S. Sun, I. Gilitschenski, H. Nishimura, M. Itkina, and F. Shkurti, "SAFE: Multitask failure detection for vision-language-action models," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- [53] C. V. Nguyen, T. Hassner, M. Seeger, and C. Archambeau, "LEEP: A new measure to evaluate transferability of learned representations," in *Proc. International Conference on Machine Learning (ICML)*, 2020.
- [54] Y. Bao, Y. Li, S.-L. Huang, L. Zhang, L. Zheng, A. Zamir, and L. Guibas, "An information-theoretic approach to transferability in task transfer learning," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2019.
- [55] C. Lyle, Z. Zheng, E. Nikishin, B. Avila Pires, R. Pascanu, and W. Dabney, "Understanding plasticity in neural networks," in *Proc. International Conference on Machine Learning (ICML)*, 2023.
- [56] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Linear mode connectivity and the lottery ticket hypothesis," in *Proc. International Conference on Machine Learning (ICML)*, 2020.

APPENDIX A PRIOR FORMULATIONS

This appendix specifies, for each of the seven prior variants in Table I, the exact sampling procedure used to draw the prior sample a_0 that the flow-matching velocity field transports to the target action during training and inference. A common convention applies to denoising budgets: if invoking π_{pre} to denoise from a Gaussian to a clean action would take N steps under its pretraining objective (DDPM for LBM 1.0, flow matching for $\pi_{0.5}$ and GR00T N1.5), then priors that invoke π_{pre} to construct a_0 (A_{pre} , $A_{\text{pre}} + \sigma_A \mathcal{Z}$, $E_{\text{pre}} + \sigma_E \mathcal{Z}$) spend $N/2$ steps on prior construction and $N/2$ on the subsequent flow-matching transport. Priors that do not invoke π_{pre} during sampling ($\mathcal{Z}$, Cocos, BRIDGER, Retrieval) use the full N steps for transport. Concretely, $N=8$ for LBM 1.0, $N=10$ for $\pi_{0.5}$, and $N=4$ for GR00T N1.5. We select $\sigma_A=1$ and $\sigma_E=0.25$ for the main experiments; see Appendix D-E for the supporting grid search.

Following Section III, o denotes the conditioning observation and $e \in \mathbb{R}^{d_e}$ the observation embedding passed to the pretrained action head. For LBM 1.0, e is the concatenated output of the CLIP ViT-B/16 vision encoder applied to each camera over the two-frame history ($d_e=6732$). For $\pi_{0.5}$, e is the concatenation of the SigLIP visual tokens across cameras. For GR00T N1.5, e is the 1536-dimensional EAGLE-VLM embedding (Qwen3-1.7B LLM + SigLIP-2 vision encoder with a learned projection) conditioning the DiT action head. H denotes the action-horizon length and d_a the per-timestep action dimensionality, so each action sample lies in $\mathbb{R}^{H \times d_a}$: $H=16$, $d_a=20$ for LBM 1.0; $H=50$, $d_a=12$ for $\pi_{0.5}$ on RoboCasa; and $H=16$, $d_a=12$ (effective; padded to a 32-dimensional action slot) for GR00T N1.5 on RoboCasa.

a) *Gaussian*, $\mathcal{Z}$:

$$a_0 \sim \mathcal{N}(0, I), \quad a_0 \in \mathbb{R}^{H \times d_a}. \quad (2)$$

No auxiliary training.

b) *Pretrained-model actions*, A_{pre} : a_0 is produced by denoising a Gaussian sample through a frozen copy of the pretrained policy for $N/2$ steps,

$$a_0 = \pi_{\text{pre}}(z \mid e; N/2), \quad z \sim \mathcal{N}(0, I), \quad (3)$$

where $\pi_{\text{pre}}(\cdot \mid e; K)$ denotes running the pretrained denoiser from z for K steps conditioned on e , under its pretraining objective (DDPM for LBM 1.0, flow matching for $\pi_{0.5}$ and GR00T N1.5). No auxiliary training.

c) *Perturbed pretrained-model actions*, $A_{\text{pre}} + \sigma_A \mathcal{Z}$:

$$a_0 = \pi_{\text{pre}}(z \mid e; N/2) + \sigma_A \tilde{z}, \quad z, \tilde{z} \sim \mathcal{N}(0, I), \quad (4)$$

with $\sigma_A=1$ (selected from the noise-scale ablation of Fig. 19). No auxiliary training.

d) *Perturbed-embedding prior*, $E_{\text{pre}} + \sigma_E \mathcal{Z}$: Noise is injected into the observation embedding before π_{pre} denoises,

$$\tilde{e} = e + \sigma_E \xi, \quad \xi \sim \mathcal{N}(0, I); \quad a_0 = \pi_{\text{pre}}(z \mid \tilde{e}; N/2), \quad (5)$$

with $\sigma_E=0.25$ (selected from Fig. 19). No auxiliary training.

e) *Cocos* [9]: An auxiliary encoder-decoder F_ϕ is trained to reconstruct e through an action-space bottleneck of dimension $H \cdot d_a$; its bottleneck activation is then used as the mean of a Gaussian prior in the action space,

$$a_0 \sim \mathcal{N}(\alpha F_\phi(e), \beta^2 I), \quad \alpha=1, \beta=1. \quad (6)$$

F_ϕ is a single-layer Transformer autoencoder (hidden dimension 384, 6 attention heads) whose decoder output matches e . Before policy fine-tuning, we collect 1000 steps of (e, a) pairs under the frozen pretrained policy (2000 for Setting B_{real}), then train F_ϕ for 100 epochs with a negative cosine-similarity reconstruction loss (Adam, learning rate 3×10^{-4} , batch size 256). F_ϕ is frozen thereafter. The original Cocos paper uses $\beta=0.2$; that value collapsed the prior too tightly around its mean to remain useful under policy fine-tuning, so we report $\beta=1$, which was reliable in preliminary sweeps. Additionally, we noted that the original paper kept the policy encoder frozen during training, but the frozen encoder led to $\sim 0\%$ success rate on the LBM Eval benchmark. We hence keep the policy encoder trainable when training the policy action head, and note that this is a usage difference from the original paper.

f) *BRIDGER* [8]: A CVAE maps a Gaussian latent to an action prior, conditioned on the observation embedding:

$$z \sim \mathcal{N}(0, I), \quad a_0 = D_\psi(z, e) + \sigma \eta, \quad \eta \sim \mathcal{N}(0, I), \quad (7)$$

with $\sigma=0.5$ and latent dimension 64. Encoder and decoder are 3-layer MLPs (hidden dimension 512, ELU activations). Auxiliary training mirrors Cocos: 1000 frozen-policy collection steps (2000 for Setting B_{real}), followed by 200 epochs of CVAE training (Adam, learning rate 10^{-3} , batch size 256, ELBO with reconstruction MSE and KL weight annealed linearly from 0 to $\beta_{\text{KL}}=0.5$). The CVAE is frozen thereafter. The original BRIDGER paper reports $\sigma=0$ and $\beta_{\text{KL}}=0.01$; in our fine-tuning regime those values collapsed the decoder onto a narrow slice of the demonstration manifold, so we report the values above, identified in preliminary sweeps. BRIDGER's CVAE training was also unstable on the Setting B_{real} datasets: NaN losses required restarting with a different random seed to obtain a usable prior.

g) *Retrieval*, k -NN: Before policy fine-tuning, we collect 1000 steps of (e_i, a_i) pairs under the frozen pretrained policy (2000 for Setting B_{real}) and store them as a retrieval bank. At every training and inference step, given a query embedding e , we compute

$$\mathcal{I}_k(e) = \arg \text{top-}k \{ -\|e - e_i\|_2 \}, \quad a_0 = \frac{1}{k} \sum_{i \in \mathcal{I}_k(e)} a_i + \sigma \eta, \quad (8)$$

with $k=3$, $\sigma=0.5$, and $\eta \sim \mathcal{N}(0, I)$. The bank is built once at the end of warm-up and then frozen; distances are in the same embedding space e that conditions the pretrained action head.

APPENDIX B STATISTICAL TESTING DETAILS

This appendix expands on the condensed stats description in Section V.

STEP for binary outcomes. Sequential Testing for Efficient Policy Comparison (STEP) [43] provides strict error control with adaptive stopping for success/failure comparisons. Because STEP’s false-positive rate is guaranteed to stay below a specific value (e.g., 0.05) at *any* stopping time (“anytime-valid” error control), we can monitor the test as rollouts accumulate and stop as soon as the result is decisive, without inflating the false-positive rate.

Lai fallback at large n . At per-method sample sizes above ~ 600 , STEP’s per-comparison computation becomes heavy; we fall back to the Lai sequential test [46], which is near-optimal asymptotically for the same hypothesis class. The fallback is used for the per-task CLD plots in Appendix D-H where n accumulates across tasks.

Welch’s t -test for continuous outcomes. For continuous metrics (per-observation ℓ_2 distance and cosine similarity in Section VII) we use Welch’s t -test [44]. The per-method sample sizes in our experiments (typically ≥ 10000) allow tight central-limit-theorem approximation for the sample mean.

Compact letter display (CLD) [45]. CLD assigns one or more letters to each method such that two methods *share at least one letter if and only if* their pairwise difference is not significant at the corrected level. Letters are typically ordered by mean performance (earlier letters = better) unless false negatives occur between some pairs. For example, a four-method CLD of “a,” “ab,” “b,” “c” means the first and third differ significantly, the second is indistinguishable from the first and third, and the fourth is strictly worse than all others.

APPENDIX C IMPLEMENTATION DETAILS

All fine-tuning runs use a single AWS `p5.48xlarge` node with up to $8 \times$ NVIDIA H100 80 GB GPUs.

A. Fine-Tuning Hyperparameters: LBM 1.0 Backbone

Settings A, B_{sim} , B_{real} , and their ablations (B_{FP} , A' , noise σ , encoder LR) all fine-tune the same pretrained LBM 1.0 policy [3] ($\sim 560\text{M}$ parameters: CLIP ViT-B/16 vision encoder, DiT action head with depth 10, embed dim 768, 12 heads). Table III lists the hyperparameters held identical across all prior variants; per the controlled comparison protocol of Section V, only the prior differs.

The Gaussian baseline $\mathcal{Z}$ keeps LBM 1.0’s DDPM pretraining objective; all non-Gaussian priors use flow matching. Setting B_{FP} (Fig. 14) re-runs the Gaussian under flow matching to confirm that this objective choice does not confound the comparison.

B. Fine-Tuning Hyperparameters: $\pi_{0.5}$ Backbone

Setting C fine-tunes the public $\pi_{0.5}$ checkpoint [4] ($\sim 3\text{B}$ parameters: SigLIP vision encoder, Gemma 2B VLM, Gemma 300M action expert). Hyperparameters follow the official `openpi` release and are reproduced in Table IV for completeness.

C. Fine-Tuning Hyperparameters: GR00T N1.5 Backbone

Setting C’s GR00T N1.5 runs fine-tune the public GR00T N1.5 3B checkpoint [5] (EAGLE vision-language backbone with Qwen3-1.7B LLM and SigLIP-2 vision encoder, flow-matching DiT action head; $\sim 3.7\text{B}$ total parameters). Following the official GR00T N1.5 release, we freeze the EAGLE backbone and update only the action-head projector and the DiT; Table V lists the hyperparameters.

D. Tasks and Demonstration Counts

Table VI lists the eight LBM Eval tasks used in Settings A and B_{sim} , with full names as in the LBM 1.0 paper [3]. Setting A uses only the five-task Kitchen-K subset. All eight tasks were unseen during LBM 1.0 pretraining: the five Kitchen-K tasks come from a new scenario, while the three Breakfast/Shelf/DryingRack tasks are unseen-variant tasks whose scenarios (but not the tasks themselves) appeared during pretraining. The three unseen-variant tasks form the “easier” group of Fig. 12, and the five Kitchen-K tasks the “harder” group.

Table VII lists the five hardware tasks used in Setting B_{real} . *BikeRotorInstall* and *ClearKitchenCounter* are adapted from LBM 1.0 [3]; the remaining three (*FoodBank*, *GatheringIngredientsForCookies*, *BusBin*) are new long-horizon bimanual tasks (kitchen preparation and sorting) introduced in this work.

For Setting C, we use all 34 RoboCasa [49] tasks (18 atomic, 16 composite) as released; demonstration counts and per-task descriptions follow the RoboCasa paper and we do not reproduce them here.

APPENDIX D ADDITIONAL EXPERIMENTAL RESULTS

A. Setting B_{sim} : Easier vs. Harder Task Breakdown

The eight tasks in Setting B_{sim} were all unseen during LBM 1.0 pretraining but vary in difficulty. Fig. 12 splits them into the three easier tasks (higher baseline success rates) and the five harder Kitchen-K tasks; the prior-independence finding holds within each group.

B. Raw LogME Scores

Fig. 13 shows the raw (unnormalized) LogME scores across data fractions. The sharp drop from 5% to 25% data is a small-sample artifact of the LogME computation (not necessarily a real change in feature quality). CLD letters are identical to those in Fig. 7 since both are computed from the same raw values.

C. Setting B_{FP} : Flow-Matching-Pretrained Baseline

In the main experiments, the Gaussian baseline keeps LBM 1.0’s DDPM pretraining objective while the non-Gaussian priors use flow matching (Section V). To rule out that this objective mismatch confounds our results, Setting B_{FP} re-trains all seven methods—including the Gaussian—under flow matching end-to-end on the five Kitchen-K tasks at 100% data. Fig. 14 shows that the conclusion is unchanged: no prior significantly outperforms the others. The three diagnostic analyses of Section VII carry over to this regime and are reported in Appendix D-D (Figs. 15 to 18).

TABLE III
FINE-TUNING HYPERPARAMETERS FOR THE LBM 1.0 BACKBONE (SETTINGS A, B_{SIM} , B_{REAL}). VALUES IN PARENTHESES APPLY TO THE CORRESPONDING ABLATIONS. THE FINE-TUNING STEP BUDGET IN SETTING B_{SIM} FOLLOWS A DISCRETE LOOKUP BY DATA FRACTION.

Hyperparameter	Value
Optimizer	AdamW ($\beta_1=0.95$, $\beta_2=0.999$, $\epsilon=10^{-8}$)
Weight decay	10^{-6}
Peak learning rate	2×10^{-5}
LR schedule	Cosine decay with 50-step linear warmup
Trainable parameters	Full model (CLIP ViT-B/16 encoder + DiT action head), with encoder LR multiplier 1.0 (0.1 in the encoder-LR ablation)
Global batch size	304 (38 per GPU $\times$ 8 GPUs)
Gradient accumulation	1
EMA	enabled (inverse-gamma 1.0, power 0.75, max decay 0.9999)
Action horizon H	16 (first 8 executed before replanning)
Observation history	2 frames
Fine-tuning steps (Setting B_{SIM} , 100%)	3000
Fine-tuning steps (Setting B_{SIM} , 5/25/50/75%)	500 / 1000 / 1500 / 2500
Fine-tuning steps (Setting A, stage 1 + 2)	1500 (on 50% data) + 3000 (on 100% data)
Fine-tuning steps (Setting B_{REAL})	30000
Denoising steps at inference	8 total (4 via frozen copy for A_{pre} construction, 4 via the fine-tuned action expert)
Aux-model data collection (Cocos, BRIDGER, Retrieval)	1000 steps of frozen-policy (e, a) collection (2000 for Setting B_{REAL})

TABLE IV
FINE-TUNING HYPERPARAMETERS FOR THE $\pi_{0.5}$ BACKBONE (SETTING C).

Hyperparameter	Value
Optimizer	AdamW ($\beta_1=0.9$, $\beta_2=0.95$, $\epsilon=10^{-8}$)
Weight decay	10^{-10}
Peak learning rate	2.5×10^{-5}
LR schedule	Cosine decay to 2.5×10^{-6} with 100-step linear warmup
Gradient clipping	$\ g\ _2 \leq 1.0$
Trainable parameters	Full model (SigLIP vision encoder + Gemma 2B VLM + Gemma 300M action expert)
Global batch size	64 (8 per GPU $\times$ 8 GPUs)
Gradient accumulation	1
Precision	bfloat16
Action horizon H	50
Observation history	1 frame
Fine-tuning steps	4000
Denoising steps at inference	10 total (5 via frozen copy for A_{pre} construction, 5 via the fine-tuned action expert)

TABLE V
FINE-TUNING HYPERPARAMETERS FOR THE GR00T N1.5 BACKBONE (SETTING C).

Hyperparameter	Value
Optimizer	AdamW ($\beta_1=0.95$, $\beta_2=0.999$, $\epsilon=10^{-8}$)
Weight decay	10^{-5}
Peak learning rate	3×10^{-5}
LR schedule	Cosine decay with 200-step linear warmup (5% of 4000 steps)
Gradient clipping	$\ g\ _2 \leq 1.0$
Trainable parameters	Action-head projector + DiT (EAGLE backbone frozen)
Global batch size	128 (64 per GPU $\times$ 2 GPUs)
Gradient accumulation	1
Precision	bfloat16 (with tf32 matmul accumulators)
Action horizon H	16
Observation history	1 frame
Fine-tuning steps	4000
Denoising steps at inference	4 total (2 via frozen copy for A_{pre} construction, 2 via the fine-tuned action expert)

TABLE VI
SIMULATION TASKS USED IN SETTINGS A AND B_{SIM} (LBM EVAL BENCHMARK). DEMONSTRATION COUNTS FOLLOW LBM 1.0 [3]. FRACTIONAL DATA EXPERIMENTS IN SETTING B_{SIM} RANDOMLY SUBSAMPLE THIS FULL POOL PER TASK WITHOUT REPLACEMENT (E.G., 50% OF *DumpVeg* USES 196 RANDOMLY SELECTED DEMONSTRATIONS OUT OF 392).

Short name	Full name	Scenario	# demos
<i>Easier group (three tasks, used in Setting B_{sim} only)</i>			
PutMug	PutMugInCenterOfTable	Breakfast (B)	294
PlaceAvocado	PlaceAvocadoFromBowlIntoBin	Shelf (S)	196
PutSpatula	PutSpatulaOnPlateFromUtensilCrock	DryingRack (D)	196
<i>Harder group (five tasks, used in Settings A and B)</i>			
DumpVeg	DumpVegetablesFromSmallToLargeContainer	Kitchen (K)	392
PutContainer	PutContainersOnPlate	Kitchen (K)	392
PutFruit	PutFruitInLargeContainerAndCoverWithPlate	Kitchen (K)	392
SeparateFruit	SeparateFruitsVegetablesIntoContainers	Kitchen (K)	392
TurnContainer	TurnLargeContainerUpsideDown	Kitchen (K)	392

TABLE VII
HARDWARE TASKS USED IN SETTING B_{REAL} (BIMANUAL FRANKA FR3).

Task name	One-line description	# demos
FoodBank	Pack the food box with at least one of each type of food item on the table	354
GatheringIngredientsForCookies	Gather the ingredients for making cookies	199
BusBin	Place plates and silverware into the separate bus bin, and place compost and recycling into the correct bins	202
ClearKitchenCounter [3]	Place tools into a tray, clean a cutting board with a sponge, and sweep the remaining waste into a trash bin	305
BikeRotorInstall [3]	Seat a rotor on a bicycle wheel, place a lockring, and tighten it at least one revolution with a tool	533

D. Setting B_{FP}: Diagnostic Analyses

We re-run the three analyses of Section VII (LogME feature quality, action-space similarity, and representation divergence) on Setting B_{FP} rollouts, where every method uses flow matching end-to-end, to confirm the diagnostic picture is not driven by the objective mismatch in Setting B. Only the 100% data fraction is available, so each prior contributes a single per-observation distribution per metric; violins are therefore laid out horizontally along the prior axis and the two representation-change panels shown side by side. Distance conventions follow Section VII: un-normalized action coordinates for action-space metrics, the observation embedding e for embedding-space metrics.

Feature quality (LogME). Fig. 15 reports LogME averaged across the five tasks. All four priors share CLD group “a” with means within a ~ 0.02 band, matching Section VII-A: the fine-tuned encoder is equally predictive of target actions regardless of prior.

Action-space similarity after fine-tuning. Fig. 16 shows $\|A_{\text{post}} - A_{\text{target}}\|_2$ and the corresponding cosine similarity across priors. As in Setting B (Fig. 8), the four priors largely overlap: post-fine-tuning action predictions are comparably accurate.

Representation divergence from the pretrained starting point. Fig. 17 shows the ℓ_2 action change $\|A_{\text{post}} - A_{\text{pre}}\|_2$ alongside the encoder change $\|E_{\text{post}} - E_{\text{pre}}\|_2$, with the cosine view in Fig. 18. The pattern of Section VII-C is preserved: action-space changes largely overlap while embedding-space

changes are separated more, with $\mathcal{Z}$ moving the encoder least and A_{pre} the most. Since every method now trains under flow matching, the encoder-representation divergence is driven by the prior rather than by the pretraining objective.

E. Setting B_{FP}: Noise Ablation

The noise scales σ_A and σ_E in the mixed priors $A_{\text{pre}} + \sigma_A \mathcal{Z}$ and $E_{\text{pre}} + \sigma_E \mathcal{Z}$ trade off informativeness (low scale, prior dominates) against coverage (high scale, closer to Gaussian). We run this ablation under Setting B_{FP} so that every variant shares the flow-matching objective. Fig. 19 sweeps both scales over $\{0.25, 0.5, 1.0\}$. The E_{pre} variants are robust across scales; the A_{pre} variants degrade at higher noise, suggesting that action-space jitter is more disruptive than embedding-space jitter. Based on this grid search we select $\sigma_A=1$ for $A_{\text{pre}} + \sigma_A \mathcal{Z}$ and $\sigma_E=0.25$ for $E_{\text{pre}} + \sigma_E \mathcal{Z}$ in the main experiments.

We also tried a learned observation-dependent noise scale $\sigma(o)$ parameterized as a small network, but this approach overfit severely: $\sigma(o)$ collapsed toward zero, driving training loss down while degrading rollout performance. We therefore use a fixed σ throughout.

F. Setting B_{real}: Hardware Prior-Selection Pilot

Before the full hardware evaluation (Section VI-C), we ran a pilot to choose a single representative learned prior among our three proposed action- and embedding-space variants. We compared A_{pre} , $A_{\text{pre}} + \sigma_A \mathcal{Z}$, and $E_{\text{pre}} + \sigma_E \mathcal{Z}$ on 3 of the 5

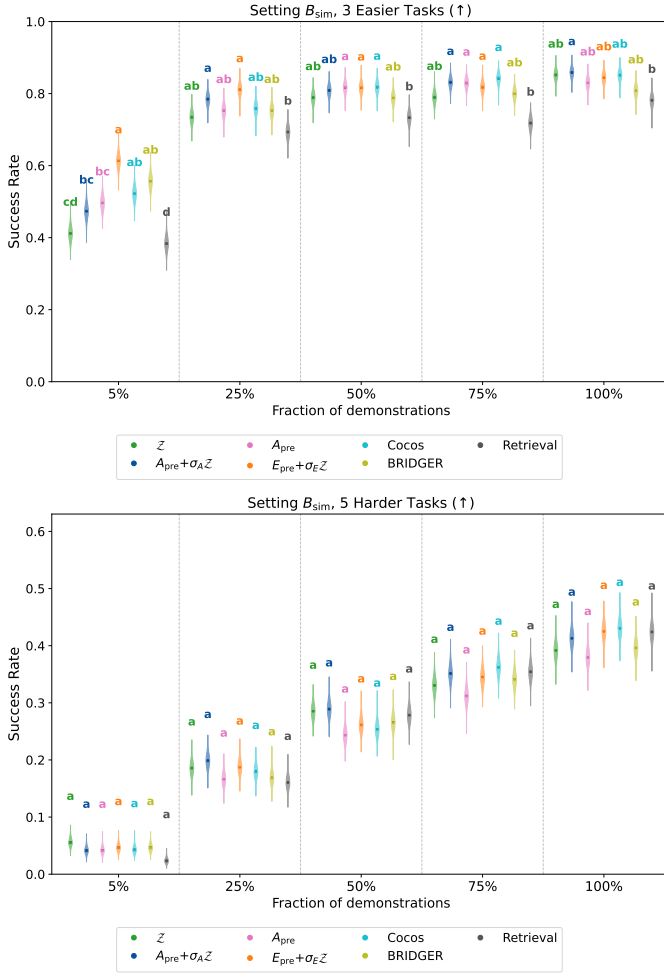


Fig. 12. Setting B_{sim} : easier (top, three tasks) vs. harder (bottom, five tasks) breakdown.

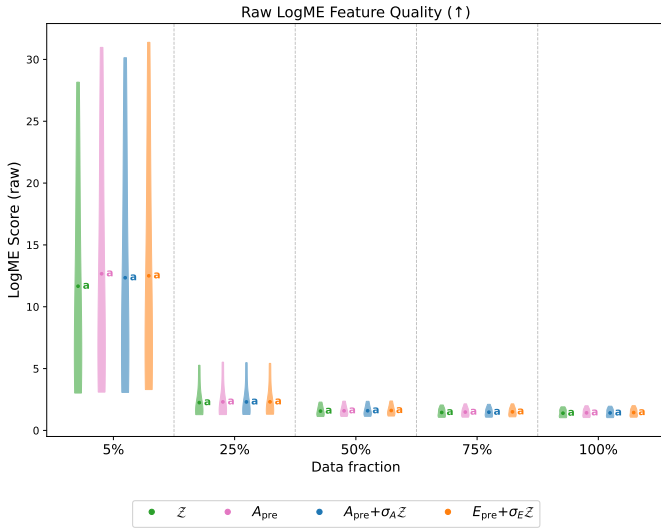


Fig. 13. Raw LogME scores across data fractions (Setting B_{sim} , eight tasks). Each violin shows the distribution of scores across tasks. The sharp drop from 5% to 25% is a small-sample artifact of the LogME computation, not necessarily a real change in feature quality. All priors share CLD group “a” at every fraction.

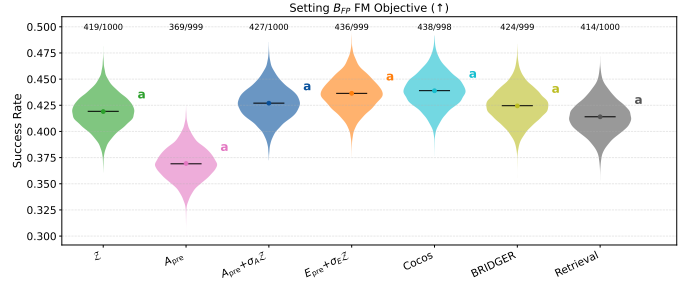


Fig. 14. Setting B_{FP} : all methods, including Gaussian, use the flow-matching objective in both pretraining and fine-tuning (five tasks).

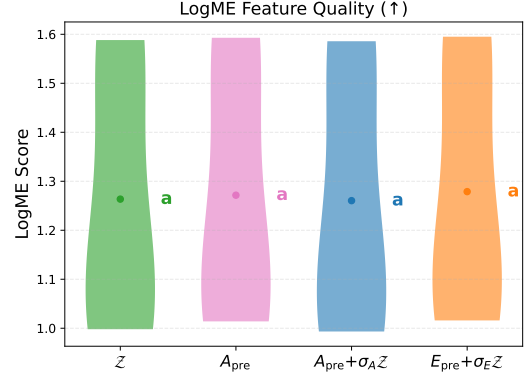


Fig. 15. LogME feature quality in Setting B_{FP} (five tasks, 100% data). Each violin shows the distribution of scores across tasks. All four priors are indistinguishable.

hardware tasks with 25 rollouts per method per task (75 per method, 225 total). Fig. 20 shows all three fall in CLD group “a” on partial-progress (69%, 74%, and 76% respectively) and on binary success (31/75, 36/75, 40/75). We carry $E_{\text{pre}} + \sigma_E Z$ forward to the full 5-method evaluation on the strength of its numerical lead, alongside Z and the three external baselines (Cocos, BRIDGER, Retrieval).

G. Setting C: Zero-Shot Prior Quality

In Setting C (Section VI-D), A_{pre} and Retrieval both underperform the Gaussian, which could suggest they are poor priors. We rule this out in Fig. 21: across both task splits, zero-shot $\pi_{0.5}$ actions remain measurably closer to the target than the Gaussian, by a factor of $\sim 3\times$.

H. Per-Task CLD Breakdowns

The main text reports success rates averaged across tasks within each setting. Here we break those averages down per task for the two LBM 1.0 settings, with CLD letters computed independently within each task. Fig. 22 shows Setting B_{sim} at 100% data across all eight tasks, ordered easier (three seen-scenario tasks) to harder (five Kitchen-K tasks); Fig. 23 shows Setting B_{real} across the five hardware tasks, with partial task progress printed under each task name. The Gaussian Z sits in CLD group “a” or “ab” on every task in both settings, so it remains statistically indistinguishable from every other prior at the task level.

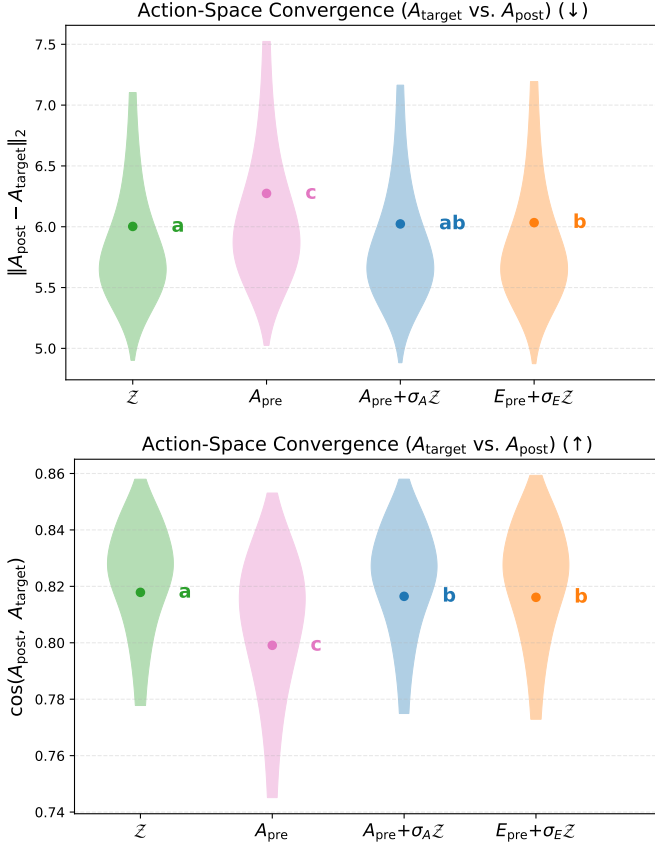


Fig. 16. Action-space similarity in Setting B_{Fp} (five tasks, 100% data). **Top:** $\|A_{\text{post}} - A_{\text{target}}\|_2$. **Bottom:** $\cos(A_{\text{post}}, A_{\text{target}})$. Violins show per-observation distributions with CLD letters from Welch’s t -test.

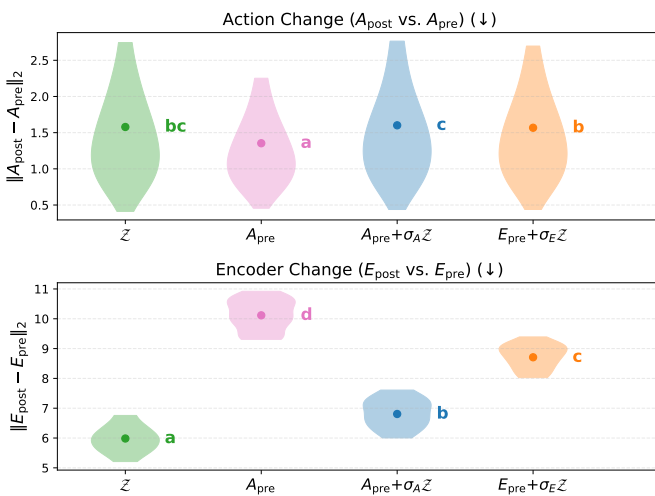


Fig. 17. Representation change in Setting B_{Fp} (ℓ_2 view, five tasks, 100% data). **Top:** action change $\|A_{\text{post}} - A_{\text{pre}}\|_2$; all priors cluster within a narrow band. **Bottom:** encoder change $\|E_{\text{post}} - E_{\text{pre}}\|_2$; A_{pre} restructures the encoder most and Z the least, with the mixed priors in between.

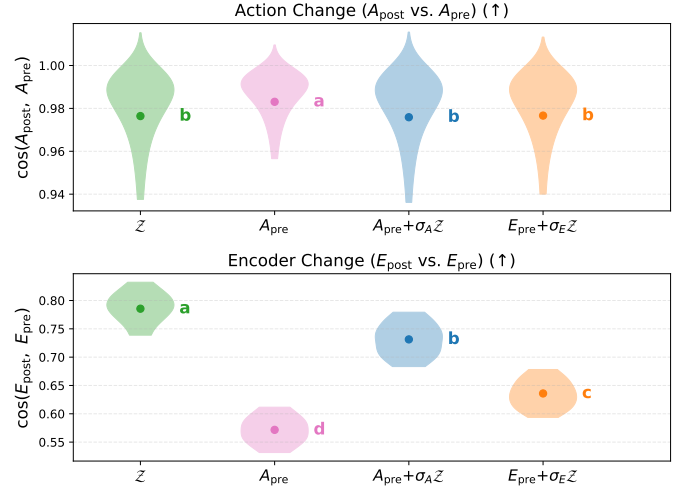


Fig. 18. Representation change in Setting B_{Fp} , cosine-similarity view (companion to Fig. 17). Higher values indicate less directional rotation from the pretrained representation.

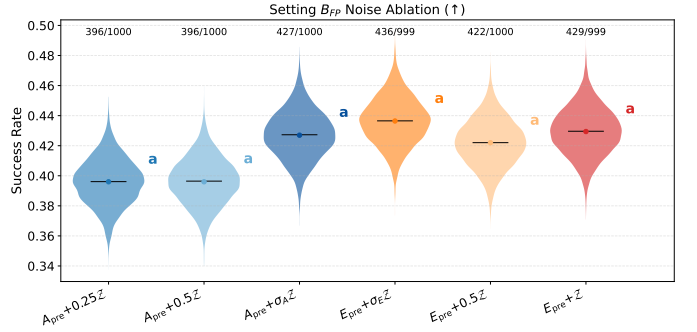


Fig. 19. Noise ablation (Setting B_{Fp} , five tasks). Performance is robust to noise scale for E_{pre} variants; A_{pre} variants degrade with higher noise.

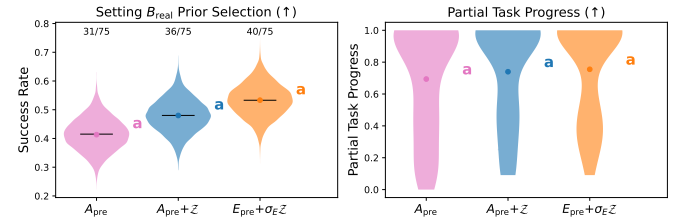


Fig. 20. Setting B_{real} prior-selection pilot (three tasks, 25 rollouts per method). **Left:** success rate. **Right:** partial task progress. All three priors are statistically tied on both metrics.

I. Zero-Shot Prior Rollouts

To complement Fig. 1, Fig. 24 shows rollouts of all seven prior variants side by side on the DumpVeg task: columns are priors in the order of Table I, rows are fixed time steps of the 20 s rollout ($\sim 1, 5, 9, 13, 15, 19$ s). To probe how closely each prior already aligns with the target task before any prior-based fine-tuning, we replace the policy’s predicted action at every step with a sample drawn directly from the prior.

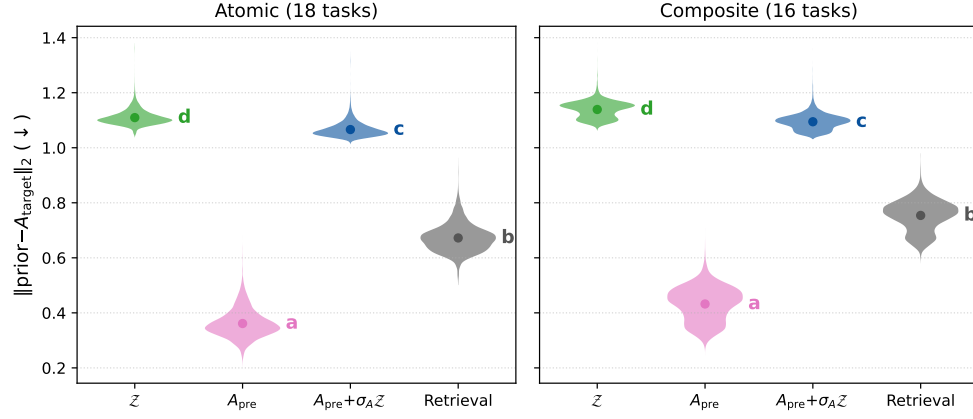


Fig. 21. Zero-shot prior-to-target distance for $\pi_{0.5}$ on RoboCasa. Violins show the distribution of per-step prior-to-target ℓ_2 distances pooled across tasks ($n=18$ atomic, $n=16$ composite); dots mark the mean and CLD letters are from Welch’s t -test. A_{pre} remains substantially closer to the target than the Gaussian on both splits.

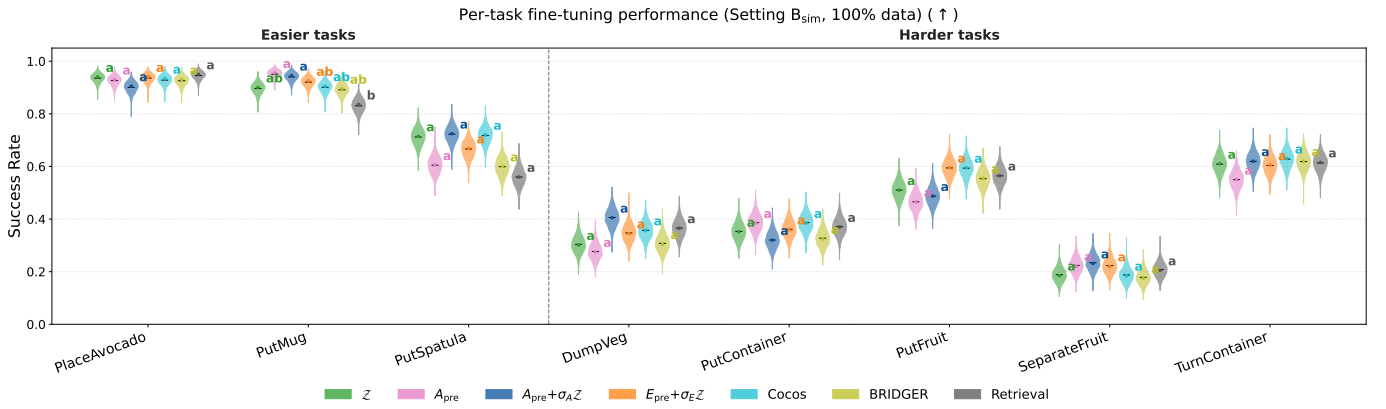


Fig. 22. Setting B_{sim} per-task success rates at 100% data (LBM 1.0, eight tasks, 200 rollouts per method-task). CLD letters are assigned independently within each task. The vertical dotted line separates the 3 easier tasks (left) from the 5 harder Kitchen-K tasks (right).

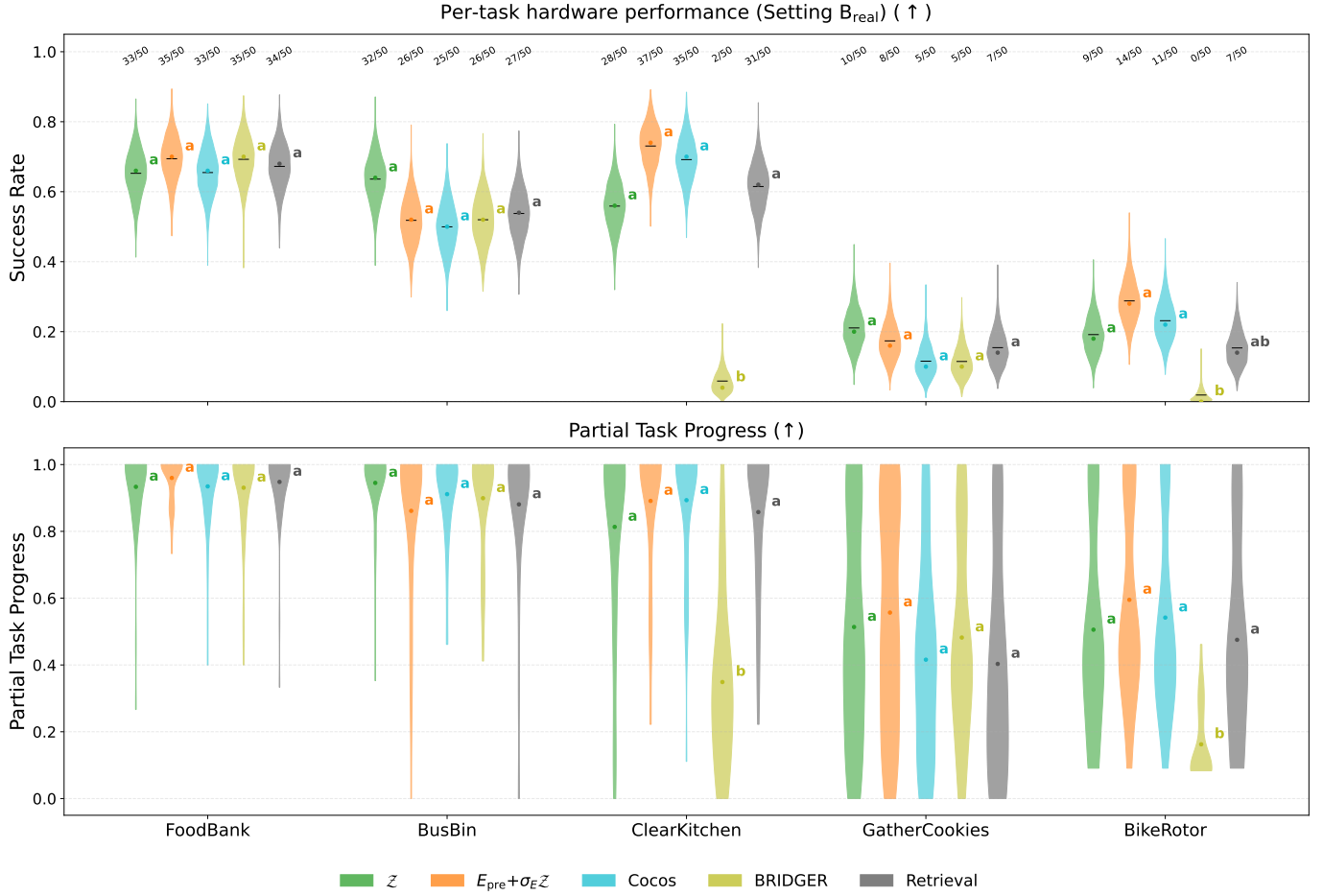


Fig. 23. Setting B_{real} per-task performance (LBM 1.0 on hardware, five tasks, 50 A/B-test rollouts per method-task). **Top:** success rate violins with CLD letters computed within each task. **Bottom:** partial-task-progress violins with Welch CLD.

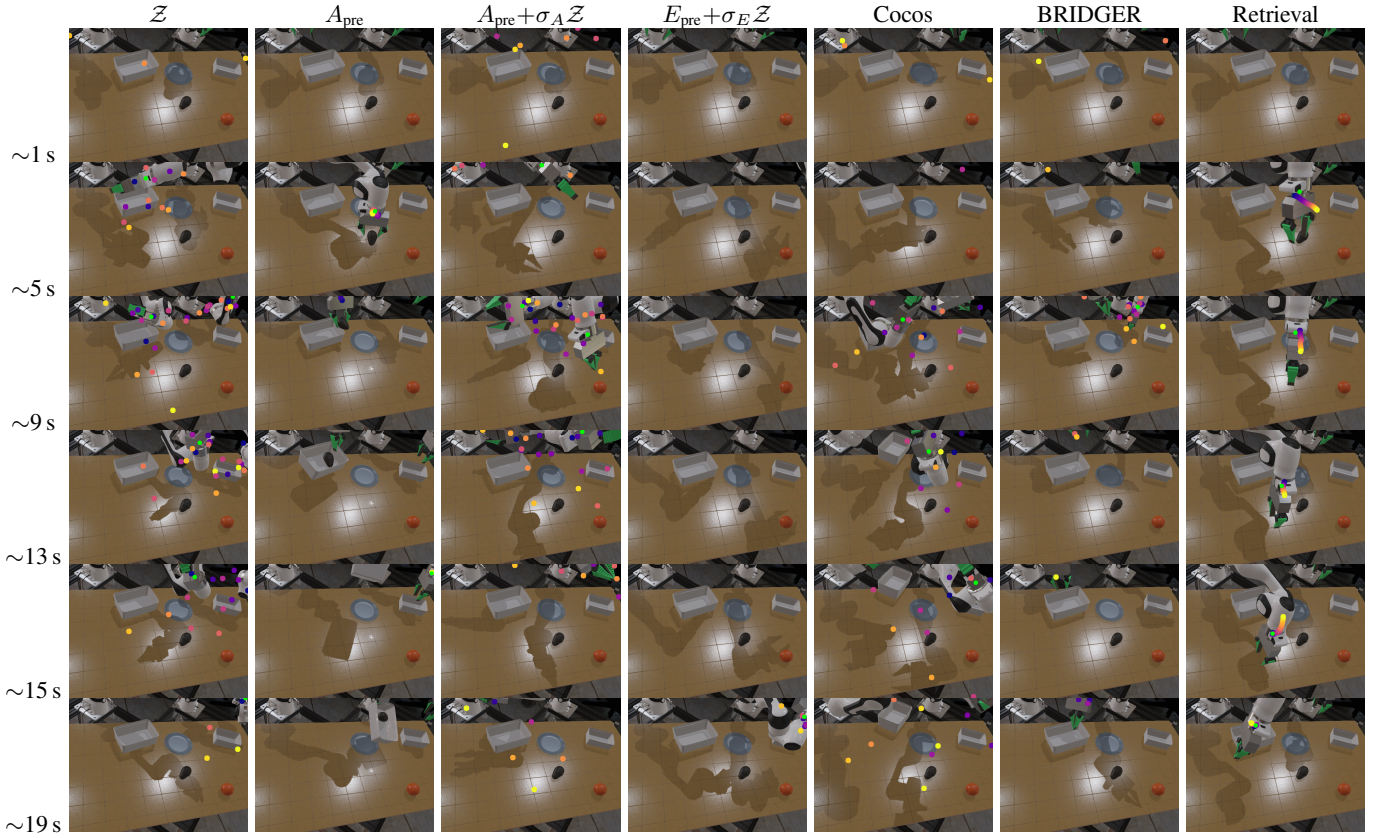


Fig. 24. Zero-shot rollouts of all seven priors on DumpVeg (columns: priors as in Table I; rows: progression over a 20 s rollout). Colored dots show both arms’ predicted end-effector positions over the action chunk horizon (color gradient indicates temporal ordering). Frames without visible dots occur when predicted positions project outside the camera’s field of view (e.g., A_{pre} at $\sim 9\text{--}19$ s, where arms moved off-screen after task progress) or cluster into a single indistinguishable point due to near-stationary predictions (e.g., Cocos at ~ 5 s). The Gaussian baseline (Z) is erratic throughout, as expected from pure noise. Several learned priors (e.g., A_{pre} , Cocos, Retrieval) occasionally produce rollouts that make legitimate task progress; the rest sit between these extremes.