

FlowEval: How Much Adaptation Does an Embodied Foundation Model Need at Deployment?

Zhang Xi-Jia¹, Roman Yakunin¹, Shivang Chopra¹, Chengyue Huang¹, Che Chen²
Siddharth Karamcheti¹, Zsolt Kira¹

Georgia Tech¹, University of Michigan²

Abstract: Large-scale pretraining is reducing the amount of adaptation required to deploy embodied foundation models, but it is not guaranteed that a pretrained model already provide the level of performance required in every deployment. Practitioners therefore need to understand how well a pretrained model matches a target deployment and how much adaptation data is left to be collected. We study whether a small set of successful target-domain demonstrations can reveal whether and how much target-domain data will be required before any adaptation. We introduce FLOWEVAL, a set of training-free metrics that use the flow-matching objective to probe an embodied foundation model with a set of demonstrations. FLOWEVAL quantifies predictive disagreement on successful target demonstrations to assess how well a pretrained model fits a target deployment. Across four embodied foundation model families: Pi-0.5, MolmoAct2, GR00T, and DreamZero, FLOWEVAL distinguishes seen from unseen deployments across real-world and simulated manipulation. Further, by finetuning Pi-0.5 and GR00T on 17 RoboCasa365 manipulation tasks, we showed that FLOWEVAL correlates with how success rate improves as additional adaptation data is provided. Together, these results suggest that FLOWEVAL can serve as a lightweight, training-free diagnostic of how much model–deployment adaptation may be required.

1 Introduction

Large-scale pretraining is enabling embodied foundation models to generalize across increasingly broad variation in tasks, environments, and robot embodiments [1, 2, 3, 4]. Yet broad generalization does not imply that a pretrained model will already provide the level of performance required in every downstream deployment [3, 5, 6]. In practice, models may still require target-domain adaptation to acquire novel physical behaviors, transfer to another action space, or improve behaviors that are only partially captured by the pretrained prior. While broad pretraining increasingly reduces the amount of task-specific learning required, adaptation remains an important component of deploying today’s embodied foundation models [2, 7, 8].

As embodied models become more general-purpose, this changes the deployment problem itself. Rather than asking only how to adapt a pretrained model to a downstream setting, practitioners increasingly need to understand *how much of the required behavior is already provided by pretraining and how much additional adaptation remains*. Importantly, poor zero-shot performance does not reveal whether the desired behavior can be recovered with a small amount of target-domain data or requires substantially more adaptation, and a model with stronger broad capabilities need not require less adaptation for a particular deployment. This motivates two questions: 1) whether the target deployment is already seen by the model, and, ultimately, 2) how much additional data will be required to achieve the desired level of performance?

We study whether the answer to these questions can be probed directly with a small set of successful target-domain demonstrations. Such trajectories provide examples of the behavior the deployed system is expected to reproduce, and can often be collected with substantially less effort than the

full datasets required for finetuning. This suggests a simple hypothesis: downstream adaptation difficulty should depend on how well the pretrained model already fits the intended target behavior. Rather than first finetuning several candidate models and observing which one adapts successfully, we ask whether this model–deployment relationship can be measured *before adaptation*.

A natural quantity to examine is the pretrained model’s training loss on successful target-domain trajectories. In language modeling, held-out cross-entropy and perplexity have long been used to characterize how well a model fits a target domain and to select data that better matches that domain [9, 10, 11]. Flow matching is particularly well suited to assessing model–data fit because its training objective probes not only the predicted action, but the model’s local direction of improvement toward the demonstrated action [12]. We therefore introduce FLOWEVAL, a set of training-free metrics derived from the flow-matching objective that measure how well a pretrained model fits successful target-domain demonstrations by computing the predictive disagreement. FLOWEVAL requires no finetuning and applies whether or not the pretrained model and target deployment share the same action representation. This is particularly important for deployment, as action representations and deployment interfaces can differ widely across model families and robot embodiments. [4, 13]

We evaluate FLOWEVAL on Pi-0.5, MolmoAct2, GR00T, and DreamZero, across real-world and simulated manipulation and multiple robot embodiments. Across these settings, FLOWEVAL predicted directional disagreement correlates distinguishes seen deployment distributions from unseen deployment distributions. More importantly, by finetuning Pi-0.5 and GR00T on 17 RoboCasa365 manipulation tasks, we find that the same signal correlates with how performance scales as additional adaptation data is provided. Together, these results suggest that FLOWEVAL can serve as a training-free diagnostic of how much target-domain adaptation may be required.

2 Related Work

2.1 Adapting embodied foundation models

Downstream finetuning is the widely adopted approach to adapt a general pretrained policy. State-of-the-art open source embodied foundation models, including but not limited to DreamZero [8], MolmoAct2 [7], Pi-05 [2], and GR00T [13], provide finetuning pipelines and include finetuning guidelines at release. Prior to them, OpenVLA [1], OpenVLA-OFT [5], and Octo [3] expected downstream finetuning even more. On the proprietary end, Generalist Gen1.5 [14] reported using few gradient step adaptation to better learn novel tasks; and Gemini Robotics [15, 6] reported finetuning the model to target embodiments. Although Figure AI [16], Skild [17], and Genesis AI [18] do not report downstream fine-tuning, notably, their pretraining datasets are directly built for their deployment settings. This is fundamentally different from the situation faced by practitioners, who lack the ability to construct a large-scale pretraining dataset specifically for their deployment.

Learning In-Context is a promising yet constrained approach for a pretrained embodied foundation model to adapt to a new task. DreamZero [8] and Skild S1 [17] provided video as context and shows the model is able to infer tasks from the world to state information in the videos. Generalist Gen1.5 [14] provided one-shot state to action demonstration as context and shows emerging ability of the model generalizing to novel objects. However, all require the embodiment and action representation to align between the pretrained and deployment settings. Team [14] also reported that skills learned in-context are more brittle than finetuned models.

Zero-shot execution is possible in scenarios the embodiment and action representation between pretrained and deployment settings are aligned. Fang et al. [7], Octo Model Team et al. [3], Team [19] showed zero-shot execution of tasks on supported embodiments. Similar to in-context learning, their prerequisites are more restrictive than most practical deployment settings.

2.2 Training objectives as Evaluations

The optimization objective, such as loss or negative log-likelihood (NLL), is commonly used as an evaluation metric for model performance in machine learning [20, 21] and especially on large language models [22, 23]. In robotics, validation loss is often used a proxy for performance to select policies [24, 25]. As to embodied foundation models including RT-2, Octo [3], and OpenVLA [1], however, multiple works [26, 27] showed that validation loss poorly reflects actual robot performance. Interestingly, none of these work [26, 27] tested flow-matching based models.

Flow objectives as evaluations

Likelihood-based flows naturally allows likelihood evaluations. A common use case is to evaluate the model on an entire held-out dataset, using the likelihood to measure how well the model fits unseen data [28, 29]. In contrast, using the likelihood of a single data point as a measure of whether that point is in- or out-of-distribution is a stronger assumption and is not generally guaranteed [30, 31]. However, in robotics, where the generated distribution is conditional action likelihoods $p(a | s)$, likelihood have been successfully applied as measures of action plausibility or anomaly: Wenger et al. [32] use it to detect several types of antagonistic robot behavior, while Zhang et al. [33], Chae et al. [34] effectively applied it inside offline-RL algorithms as an OOD score.

The flow matching loss shares some common properties with likelihood-trained normalizing flows, such as evaluating how well a policy fits a held-out demonstration set. Zha et al. [35] evaluates the validation loss of its flow-matching action expert on a held-out split and uses it to compare model scaling. Recent robot policy evaluations also compute the flow matching objective on held-out episodes as an offline action-prediction loss [36]. Park [37] evaluated the relationship between flow-matching loss and success rate, and discovers that lower training loss is associated with higher validation loss and higher performance.

FLOWEVAL differs from prior works in two meaningful ways. (1) We discovered that starting from pure noise provided us with cleaner evaluation signals than noisy actions (Appendix. A.1.1) (2) Instead of using loss to select checkpoints from the same training run, we use pre-adaptation loss to probe how finetuning outcomes vary across target deployments and demonstration budgets.

3 Methodology

We begin by clarifying the deployment settings considered in this work. We distinguish the conditions that fall within and outside the scope of our metric evaluation,.

Demonstration: We define a demonstration as an observation-state-action trajectory demonstrating successful completion of a task. In this work, we primarily consider demonstrations collected via human teleoperation. Although approaches such as MimicGen [38], DemoGen [39], and other forms of augmenting existing demonstrations may be interesting directions to explore, they are beyond the scope of this study (Appendix. A.2).

Deployment settings: We define a deployment setting as one or more manipulation tasks performed under a fixed environment, embodiment, and action representation. We further assume access to a small set of demonstrations for the target task(s). We do not assume that the action representations and dimensions of the deployment setting matches with a model’s pretrained distribution, since in actual deployments they often do not.

Practical Scope: Most practitioners do not have access to a foundational model trained specifically for their deployment settings – only those who pretrain the model themselves do. Instead, they often seek to adapt an existing foundation model to a target deployment without full, or even partial, access to its pretraining data. Their deployment may involve a custom embodiment, camera configuration, task setup, or action representation. Even when these components are otherwise standard, they may still differ from the exact configurations supported by the pretrained model. Our definitions therefore describe a deployment scenario that practitioners are most likely to encounter in practice.

3.1 FlowEval Metrics

Using flow matching objectives to evaluate how well a trained model fits a given dataset is a promising yet under-explored direction in robotics (Sec. 2.2). We propose FlowEval metrics based on the intuition that the flow matching objective provides a measurable signal of how well the embodied foundation model fits a deployment dataset, which further affects how much adaptation is needed. Since FlowEval relies on properties of flow-based objectives that may not generalize to other training objectives, its metrics are only applicable to embodied foundation models that use flow matching for action prediction.

For vision-language action (VLA) models, the forward pass can be written as

$$\hat{u}_A = \Phi_\theta(A^{\tau_A}, h_{vision}, h_{language}, h_{state}, h_{time_A}), \quad (1)$$

A world-action model (WAM) additionally predicts future quantities, which can extend beyond videos. We denote those future quantities by m .

$$(\hat{u}_A, \{\hat{u}_m\}_m) = \Psi_\theta(A^{\tau_A}, \{Z_m^{\tau_m}\}_m, h_v, h_\ell, h_s, h_{t_A}, \{h_{t_m}\}_m), \quad (2)$$

For ground-truth actions A^* and future targets Z_m^* , conditional flow matching uses

$$A^{\tau_A} = (1 - \tau_A)A^* + \tau_A\epsilon_A, \quad Z_m^{\tau_m} = (1 - \tau_m)Z_m^* + \tau_m\epsilon_m, \quad (3)$$

with Gaussian noise $\epsilon_A, \epsilon_m \sim \mathcal{N}(0, I)$ and reference velocities $u_A = \epsilon_A - A^*$ and $u_m = \epsilon_m - Z_m^*$.

FlowEval initializes the action and future-target trajectories from Gaussian noise, setting $A^1 = \epsilon_A$ and $Z_m^1 = \epsilon_m$. Subsequent states follow the model’s inference dynamics. At each inference step k , we evaluate $\hat{u}_{A,k}$ and $\hat{u}_{m,k}$ against the fixed reference velocities defined above. Although validating with held-out trajectories conventionally starts from ground-truth-interpolated noisy actions, we find that starting from pure noise instead yields more stable results, discussed in A.1.1.

Velocity error. The action and future-prediction velocity errors are

$$L_{A,k} = \|\hat{u}_{A,k} - u_A\|_2^2, \quad L_{m,k} = \|\hat{u}_{m,k} - u_m\|_2^2, \quad (4)$$

These are basically the loss objectives. We refer to these as velocity error to differentiate from training or validation loss, which conventionally starts from ground-truth-interpolated noisy actions.

Cosine similarity. We measure the directional agreements with the reference velocities:

$$\begin{aligned} \text{cosine}_{A,k} &= \frac{\langle M_A \odot \hat{u}_{A,k}, M_A \odot u_A \rangle}{\|M_A \odot \hat{u}_{A,k}\|_2 \|M_A \odot u_A\|_2 + \varepsilon}, \\ \text{cosine}_{m,k} &= \frac{\langle \hat{u}_{m,k}, u_m \rangle}{\|\hat{u}_{m,k}\|_2 \|u_m\|_2 + \varepsilon}. \end{aligned} \quad (5)$$

where M_A is the action mask, and $\varepsilon > 0$ stabilizes the denominator.

3.2 Design Choices

FLOWEVAL metrics are defined over the full trajectory of diffusion steps under the model’s configuration. However, computing all steps is computationally expensive and may be unnecessary. We conduct an empirical study along two design axes: (i) distinguishability, the extent to which a metric distinguishes different target settings, and (ii) computational efficiency, the cost of computing the metric from target demonstrations.

We define distinguishability as the ratio of between-task to within-task variance; higher scores indicate that the metric varies more across tasks than across demonstrations within a task. Computational efficiency is naively evaluated by the step index because exact n forward passes are required to compute step n . As shown in Table 1, FLOWEVAL metrics on Pi-05-Base and DreamZero-DROID

achieved distinguishability at step 1 exceeding 90% of the maximum across all steps, suggesting that a single step is empirically sufficient when accounting for computational cost. We thus only take the first step for future analysis. Full results of distinguishability scores are included in Table 6 and 5.

Model	Metric	Step 1	Best	Relative
Pi05-Base	Cosine	0.868	0.885	98.0%
	Velocity Error	1.097	1.104	99.4%
DreamZero-DROID	Action Cosine	0.455	0.498	91.3%
	Action Velocity Error	0.244	0.267	91.3%
	Video Cosine	0.799	1.170	68.3%
	Video Velocity Error	0.735	1.141	64.4%

Table 1: Distinguishability score at step 1 compared with the best value observed across all steps. Relative performance is computed as $100 \times \text{step 1} / \text{Best}$. Evaluated on Robocasa365 target set.

4 Experiments

4.1 Predicting Unseen Deployments

We first test whether FLOWEVAL reflects how well a deployment’s required behavior is supported by the pretrained model. Our hypothesis is that successful demonstrations from a seen deployment should induce less predictive disagreement than demonstrations from an unseen deployment. Although scores may be noisy for individual trajectories, we expect this difference to emerge at the distribution level.

We test this hypothesis across three vision-language-action models: Pi-05, GR00T, MolmoAct2, and one world-action model: DreamZero, on deployment settings whose relationship to pretraining is known. For each model, we compare demonstrations from a deployment setting represented during pretraining (*seen*) with demonstrations from a deployment setting absent from pretraining (*unseen*).

Datasets. We compute FlowEval on 7 datasets across real-world and simulation, multiple action representation spaces, and three embodiments: Franka, Bimanual OpenArm, and Bimanual YAM. We collected the Franka and OpenArm datasets ourselves, and additionally use 5 internet datasets: DROID [43], Robocasa365 [44], MolmoAct2-Bimanual-YAM [7], LIBERO [45], and RoboSet [46]. The exact action representation and checkpoint configurations are discussed in Appendix A.3. Note that FLOWEVAL can be computed between different action representations. However, doing so confounds unseen deployment with a change in action representation alone and does not test the other factors of interest. We therefore restrict the main analysis to model–deployment pairs with matching action spaces between pretraining and evaluation. We additionally report cross-representation results in Appendix A.3. As expected, these pairs exhibit significantly higher velocity error and lower cosine similarity than pairs from seen deployments.

In-Pretraining	DROID	Robocasa	YAM	LIBERO	Roboset	OpenArm	Franka
Pi-05	No	No	No	No	No	No	No
GR00T-N1.6	Yes	Yes	No	No	No	No	No
MolmoAct2	Yes	No	Yes	Yes	No	No	No
DreamZero	Yes	No	No	No	No	No	No

Table 2: The full set of model–dataset pairs on which FlowEval metrics are evaluated, along with whether each dataset was included in model pretraining. Information about those datasets is based on model releases [2, 13, 7, 8] and, where applicable, company blogs [40, 41, 42]. ‘No’ indicates that we found no evidence that the dataset was included in pretraining, although the authors did not explicitly state that it was excluded. ‘No’ indicates that the authors explicitly stated that the dataset was excluded, or the pretraining dataset is publicly released and does not contain that dataset.

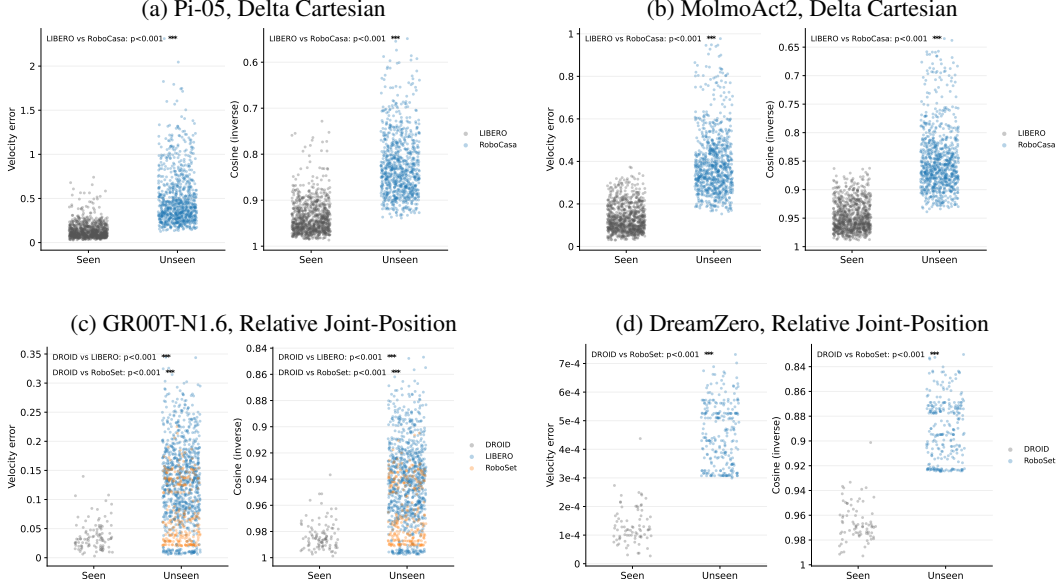


Figure 1: FlowEval results across various models and action representations.

We test whether the seen and unseen deployments are separated using a one-sided two-sample Kolmogorov–Smirnov test. For velocity error, the alternative tests whether target distribution lean toward larger errors; for cosine similarity, it tests whether target distribution lean toward smaller similarities. Across Pi-05, GR00T, MolmoAct2, and DreamZero, the corresponding distributions are significantly separated in this direction. The separation is also visible at the distribution level: trajectories from the pretraining setting extend closer to the limiting values of zero velocity error and unit cosine similarity, whereas target trajectories do not reach the same favorable tail.

As a control, we evaluate new demonstrations excluded from pretraining but collected under the same seen settings. FLOWEVAL scores of these demonstrations cluster with the seen deployment and remain distinct from the unseen deployments (Fig. 8). Because RoboCasa is the only public in-pretraining dataset that provides such held-out trajectories, this control is limited to Robocasa (Appendix A.3). Together, these results support FlowEval’s ability to identify unseen deployments and provide preliminary evidence that it recognizes seen deployments from new demonstrations.

4.2 Predicting downstream adaptation

We show that FLOWEVAL metrics are capable of predicting how performance scales with data under deployment shift. To determine whether the scaling behavior simply comes from more data or is specifically conditioned on deployment shift, we experiment with two groups of finetuning where (1) the deployment settings is genuinely absent from the model’s pretraining (2) the deployment settings is included in the model’s pretraining, but new demonstrations for the settings are available.

We use the Robocasa365 dataset [47], which is built for studying downstream finetuning behavior and thus provide us with cleanly separated pretraining and held-out datasets. Among widely used robot manipulation benchmarks [48, 49, 26, 50, 51, 52], RoboCasa365 uniquely provides the scale and a pretraining/target split over the same task set that this comparison requires, without confounding deployment shift with differences in semantic-level planning. We use all shared atomic-seen manipulation tasks between the pretrain set and target set, so that we are able to finetune on held-out demonstrations. For deployment settings absent from pretraining, we finetune Pi-05 base and GR00T-N1.5, neither of which was pretrained on RoboCasa or RoboCasa365. For settings represented in pretraining, we finetune Pi-05-RoboCasa [47] and GR00T-N1.6, both of which were pretrained on RoboCasa365.

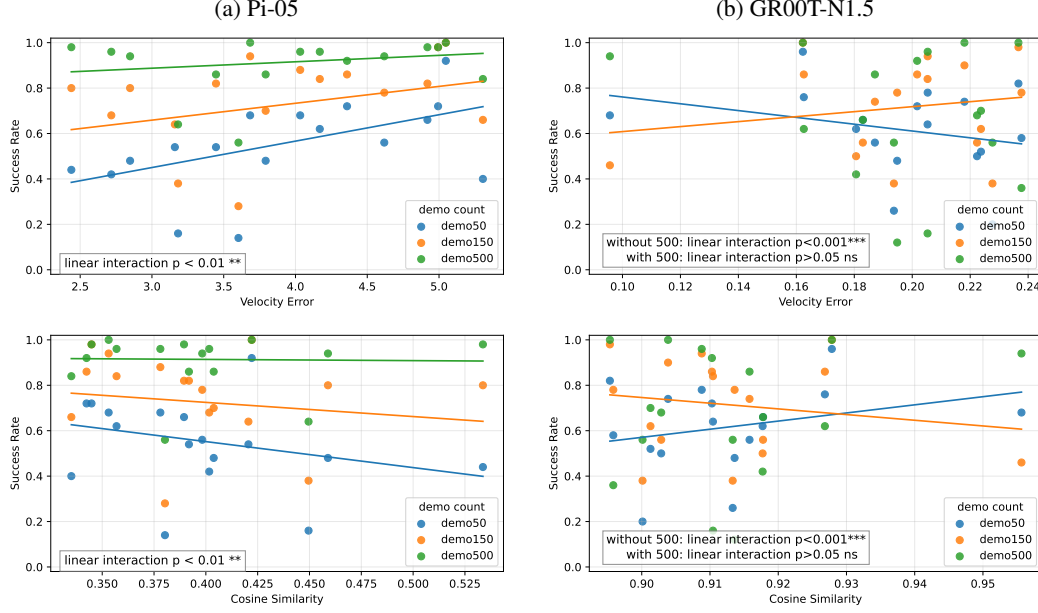


Figure 2: Regression between success rate and metric values, with significances displayed in the plots. All interaction effects are significant. Under the GR00T-N1.5 500-demonstration condition, training instability causes 29.4% of runs to collapse; these runs are excluded from the analysis (Appendix A.5.5). Training collapse is negligible in other conditions.

For each task, we compute FLOWEVAL on the pretrained model before target-domain adaptation, and then finetune with increasing amounts of target data. Following RoboCasa365 [47], we use 50, 150, and 500 demonstrations per task. Each regression observation corresponds to one task–demonstration-count pair, with success rate given by the highest observed checkpoint performance during that finetuning run and the task-level FLOWEVAL score fixed across data budgets. This design tests whether the initial model–domain relationship measured by FLOWEVAL predicts differences in the response to additional adaptation data. We fit the following model:

$$R = \beta_0 + \beta_1 L + \beta_2 m + \beta_3 Lm + \epsilon,$$

where $L = \log(N) - \overline{\log(N)}$ is the centered log number of demonstrations and $m = M - \overline{M}$ is the centered pre-adaptation FLOWEVAL metric. The coefficient of interest is (β_3) : it tests whether the change in success rate associated with additional demonstrations depends on the initial FLOWEVAL value. We fit the model with ordinary least squares using task-clustered robust covariance and test $(H_0: \beta_3 = 0)$ with a two-sided Wald test.

Under deployment shift, we observe significant interactions for both Pi-05 and GR00T-N1.5 (Figure 2). For Pi-05, both velocity error and cosine similarity are associated with how success rate changes across the 50, 150, and 500 demonstration conditions. GR00T-N1.5 shows the same relationship over the 50 and 150 demonstration conditions. Across both model families, FLOWEVAL therefore provides information before adaptation about how performance changes as additional target-domain data is provided.

In contrast, we do not observe significant interactions in the control experiments with Pi-05-RoboCasa and GR00T-N1.6, where the deployment setting is already represented during pretraining (Appendix A.5.1). This contrast supports the interpretation that the interaction is conditioned on deployment shift. Additional controls further ruled out alternative explanations: Within a model, the interaction persists after accounting for success rate improvement rooms (Appendix A.5.3); Across models, we did not find a shared task-difficulty pattern (Appendix A.5.2) explaining the correlation.

Notably, the relationship between FLOWEVAL scores and finetuned performance depends on the adaptation setting. When adaptation crosses action spaces, action-representation mismatch introduces an additional source of predictive disagreement beyond prediction error. Lower error therefore need not imply higher finetuned success. This phenomenon is observed for Pi-05 in Fig.2. We also observe that additional demonstrations sometimes reduce performance on GR00T-N1.5, consistent with prior observations [37]. Training configurations may affect the direction and shape of these curves (Appendix A.5.4). The significant interactions observed across both model families nevertheless suggest that FLOWEVAL’s predictive relationship with performance–data interaction extends beyond a single model and training configuration.

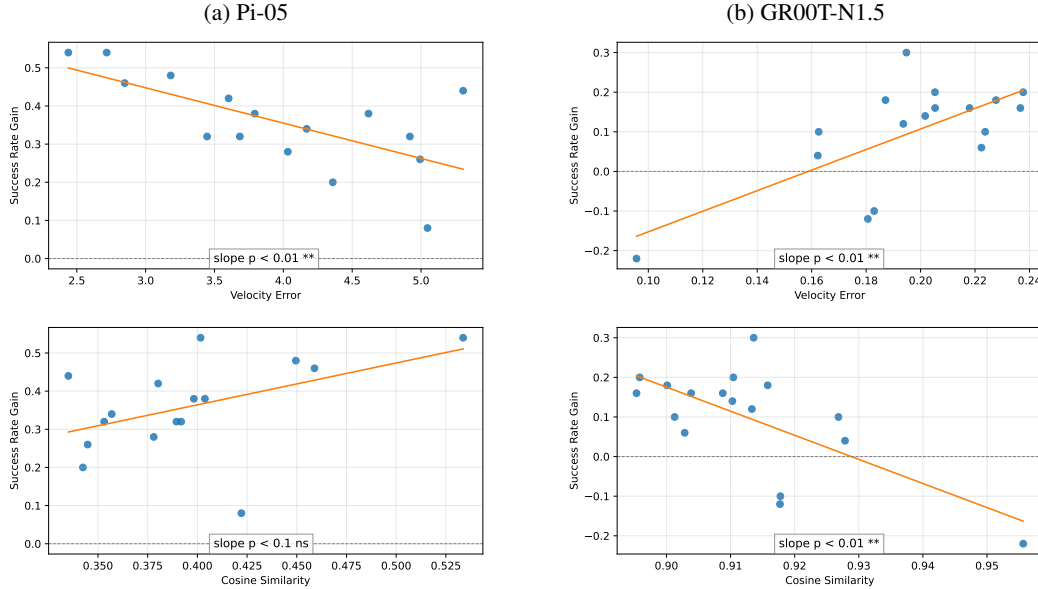


Figure 3: Relation between performance gain and metric values. 500 - 50 demos success rate for Pi-05, 150 - 50 demos success rate for GR00T-N1.5.

5 Conclusion and Future Work

We introduced FLOWEVAL, a training-free diagnostic for probing how well a pretrained embodied foundation model fits a target deployment. Across multiple model families and deployment settings, FLOWEVAL exhibits greater predictive agreement on deployment settings represented during pretraining and, under deployment shift, correlates with how performance scales as additional adaptation data is provided.

Future work should test whether these signals generalize across broader model and task families, and whether they can support calibrated predictions of the amount of adaptation data required to reach a target level of performance. More broadly, the same idea may extend from diagnosing *how much adaptation remains* to diagnosing *what kind of prior is missing*. Modern embodied models increasingly combine heterogeneous sources of prior knowledge—robot trajectories, human video, simulation, world-model pretraining, and large-scale vision-language representations—whose contributions may differ substantially across downstream behaviors. A natural question is whether model-based diagnostics can estimate how much a particular source of additional pretraining would improve a model’s fit to a target deployment, before actually carrying out that pretraining.

Such an analysis would require controlled pretraining interventions across data sources, model families, and scales, and is therefore beyond the scope of an individual academic study. At sufficient scale, however, it could turn the composition of embodied pretraining into a measurable allocation problem: given a model and a family of target deployments, which additional source of experience most efficiently closes the remaining gap?

Acknowledgments

This research used resources provided through NSF ACCESS allocation CIS250679. We are grateful to Dr. Joseph Campbell for helping us obtain this allocation.

References

- [1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model, 2024. URL <https://arxiv.org/abs/2406.09246>.
- [2] P. 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, 2025. URL <https://arxiv.org/abs/2504.16054>.
- [3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.
- [4] E. Collaboration, A. O’Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frueger, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Thompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Pan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, M. Z. Irshad, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, V. Guizilini, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang,

- Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2025. URL <https://arxiv.org/abs/2310.08864>.
- [5] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL <https://arxiv.org/abs/2502.19645>.
- [6] G. R. Team, A. Abdolmaleki, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, A. Balakrishna, N. Batchelor, A. Bewley, J. Bingham, M. Bloesch, K. Bousmalis, P. Brakel, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, C. Chan, O. Chang, L. Chappellet-Volpini, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, A. Collister, D. B. D’Ambrosio, S. Dasari, T. Davchev, M. K. Dave, C. Devin, N. D. Palo, T. Ding, C. Dörsch, A. Dostmohamed, Y. Du, D. Dwibedi, S. T. Egambaram, M. Elabd, T. Erez, X. Fang, C. Fantacci, C. Fong, E. Frey, C. Fu, R. Gao, M. Giustina, K. Gopalakrishnan, L. Graesser, O. Groth, A. Gupta, R. Hafner, S. Hansen, L. Hasenclever, S. Haves, N. Heess, B. Hernaez, A. Hofer, J. Hsu, L. Huang, S. H. Huang, A. Iscen, M. G. Jacob, D. Jain, S. Jesmonth, A. Jindal, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, M. Kecman, J. C. Kew, D. Kim, F. Kim, J. Kim, T. Kipf, S. Kirmani, K. Konyushkova, L. Y. Ku, Y. Kuang, T. Lampe, A. Laurens, T. A. Le, I. Leal, A. X. Lee, T.-W. E. Lee, G. Lever, J. Liang, L.-H. Lin, F. Liu, S. Long, C. Lu, S. Maddineni, A. Majumdar, K.-K. Maninis, A. Marmon, S. Martinez, A. H. Michaely, N. Milonopoulos, J. Moore, R. Moreno, M. Neunert, F. Nori, J. Ortiz, K. Oslund, C. Parada, E. Parisotto, A. Paryag, A. Pooley, T. Power, A. Quaglino, H. Qureshi, R. V. Raju, H. Ran, D. Rao, K. Rao, I. Reid, D. Rendleman, K. Reymann, M. Rivas, F. Romano, Y. Rubanova, P. P. Sampedro, P. R. Sanketi, D. Shah, M. Sharma, K. Shea, M. Shridhar, C. Shu, V. Sindhvani, S. Singh, R. Soricut, R. Sterneck, I. Storz, R. Surdulescu, J. Tan, J. Tompson, S. Tunyasuvunakool, J. Varley, G. Vesom, G. Vezzani, M. B. Villalonga, O. Vinyals, R. Wagner, A. Wahid, S. Welker, P. Wohlhart, C. Wu, M. Wulfmeier, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, J. Yan, S. Yang, S. Yang, Y. Yang, H. H. Yu, W. Yu, W. Yuan, Y. Yuan, J. Zhang, T. Zhang, Z. Zhang, A. Zhou, G. Zhou, and Y. Zhou. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer, 2025. URL <https://arxiv.org/abs/2510.03342>.
- [7] H. Fang, J. Duan, D. Clay, S. Wang, S. Liu, W. Huang, X. Fan, W.-C. Tsai, S. Chen, Y. R. Wang, S. Xing, J. Cho, J. S. Park, A. Eftekhari, P. Sushko, K. Farley, A. Wadhwa, C. Harrison, W. Han, Y.-C. Lee, E. VanderBilt, R. Hendrix, S. Ellawela, L. Ngoo, J. Chai, Z. Ren, A. Farhadi, D. Fox, and R. Krishna. Molmoact2: Action reasoning models for real-world deployment, 2026. URL <https://arxiv.org/abs/2605.02881>.
- [8] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, G. Kurian, S. Indupuru, Y. L. Tan, C. Zhu, J. Xiang, A. Malik, K. Lee, W. Liang, N. Ranawaka, J. Gu, Y. Xu, G. Wang, F. Hu, A. Narayan, J. Bjorck, J. Wang, G. Kim, D. Niu, R. Zheng, Y. Xie, J. Wu, Q. Wang, R. Julian, D. Xu, Y. Du, Y. Chebotar, S. Reed, J. Kautz, Y. Zhu, L. J. Fan, and J. Jang. World action models are zero-shot policies, 2026. URL <https://arxiv.org/abs/2602.15922>.
- [9] R. C. Moore and W. Lewis. Intelligent selection of language model training data. In J. Hajič, S. Carberry, S. Clark, and J. Nivre, editors, *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL <https://aclanthology.org/P10-2041/>.
- [10] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don’t stop pretraining: Adapt language models to domains and tasks, 2020. URL <https://arxiv.org/abs/2004.10964>.
- [11] S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. S. Liang, Q. V. Le, T. Ma, and A. W. Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Inform*

- tion Processing Systems, volume 36, pages 69798–69818. Curran Associates, Inc., 2023. doi:10.52202/075280-3059. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/dcba6be91359358c2355cd920da3fcbd-Paper-Conference.pdf.
- [12] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling, 2023. URL <https://arxiv.org/abs/2210.02747>.
 - [13] NVIDIA, J. Bjorck, N. C. Fernando Castañeda, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.
 - [14] G. Team. Gen-1.5: Embodied foundation models are one-shot learners. *Generalist AI Blog*, 2026. <https://generalistai.com/blog/gen-1.5>.
 - [15] Google DeepMind. Gemini robotics 2. <https://deepmind.google/models/gemini-robotics/vla/>, 2026. Model webpage.
 - [16] Figure. Introducing helix 02: Full-body autonomy, 2026. URL <https://www.figure.ai/news/helix-02>.
 - [17] S. AI. Introducing s1: In-context learning for robotics. *Skild AI Blog*, August 2026. URL <https://skild.ai/blogs/s1>.
 - [18] G. A. Team. Gene-26.5: Advancing robotic manipulation to human level. *Genesis AI Blog*, May 2026. URL <https://genesis.ai/blog/gene-26-5-advancing-robotic-manipulation-to-human-level>.
 - [19] R. A. Team. Om-1: Frontier robot intelligence, learned firsthand from humans. *Reward AI Blog*, 2026. <https://rewardai.com/blog/OM-1/>.
 - [20] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization, 2017. URL <https://arxiv.org/abs/1611.03530>.
 - [21] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks, 2017. URL <https://arxiv.org/abs/1706.04599>.
 - [22] 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, 2020. URL <https://arxiv.org/abs/2001.08361>.
 - [23] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models, 2022. URL <https://arxiv.org/abs/2203.15556>.
 - [24] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL <https://arxiv.org/abs/1011.0686>.
 - [25] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL <https://arxiv.org/abs/2304.13705>.
 - [26] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation, 2024. URL <https://arxiv.org/abs/2405.05941>.

- [27] Z. Zhou, P. Atreya, Y. L. Tan, K. Pertsch, and S. Levine. Autoeval: Autonomous evaluation of generalist robot manipulation policies in the real world, 2025. URL <https://arxiv.org/abs/2503.24278>.
- [28] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios. Neural spline flows, 2019. URL <https://arxiv.org/abs/1906.04032>.
- [29] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021. URL <http://jmlr.org/papers/v22/19-1028.html>.
- [30] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don’t know?, 2019. URL <https://arxiv.org/abs/1810.09136>.
- [31] Y. Han, H. Sun, and T. Lin. When can conditional flow matching replace pointwise negative log-likelihood?, 2026. URL <https://arxiv.org/abs/2608.28010>.
- [32] I. Wenger, H. Ebel, and P. Eberhard. Anomalous acting agents: the deployment problem. *Multibody System Dynamics*, 64(3):465–483, 2025. ISSN 1573-272X. doi:10.1007/s11044-024-09993-1. URL <https://doi.org/10.1007/s11044-024-09993-1>.
- [33] J. Zhang, C. Zhang, W. Wang, and B. Jing. Constrained policy optimization with explicit behavior density for offline reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 5616–5630. Curran Associates, Inc., 2023. doi:10.52202/075280-0245. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/11e1900e680f5fe1893a8e27362dbe2c-Paper-Conference.pdf.
- [34] J. Chae, J. Park, Y. Shin, G. Kim, S. Han, and Y. Sung. Flow actor-critic for offline reinforcement learning, 2026. URL <https://arxiv.org/abs/2602.18015>.
- [35] L. Zha, A. J. Hancock, M. Zhang, T. Yin, Y. Huang, D. Shah, A. Z. Ren, and A. Majumdar. Lap: Language-action pre-training enables zero-shot cross-embodiment transfer, 2026. URL <https://arxiv.org/abs/2602.10556>.
- [36] Robocurve. GR00T N1.7: SO-101 fine-tune on MolmoAct2 community data. Hugging Face, 2026. URL <https://huggingface.co/robocurve/gr00t-n1.7-so101-molmoact2>. Accessed: 2026-09-15.
- [37] S. Park. Behavioral cloning mystery. <https://seohong.me/blog/behavioral-cloning-mystery/>, Aug. 2026. Blog post.
- [38] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations, 2023. URL <https://arxiv.org/abs/2310.17596>.
- [39] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning, 2025. URL <https://arxiv.org/abs/2502.16932>.
- [40] K. Black et al. $\pi_{0.5}$: A VLA with open-world generalization. *Physical Intelligence*, Apr. 2025. URL <https://www.pi.website/blog/pi05>. Published April 22, 2025. Accessed: 2026-10-04.
- [41] GEAR Team et al. GR00T N1.6: An improved open foundation model for generalist humanoid robots. NVIDIA Research, Dec. 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_6/. Published December 15, 2025. Accessed: 2026-10-04.

- [42] S. Ye et al. DreamZero: World action models are zero-shot policies, 2026. URL <https://dreamzero0.github.io/>. Accessed: 2026-10-04.
- [43] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, V. Guizilini, D. A. Herrera, M. Heo, K. Hsu, J. Hu, M. Z. Irshad, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O’Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL <https://arxiv.org/abs/2403.12945>.
- [44] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots, 2026. URL <https://arxiv.org/abs/2603.04356>.
- [45] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL <https://arxiv.org/abs/2306.03310>.
- [46] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking, 2023. URL <https://arxiv.org/abs/2309.01918>.
- [47] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots, 2026. URL <https://arxiv.org/abs/2603.04356>.
- [48] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL <https://arxiv.org/abs/2306.03310>.
- [49] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022. URL <https://arxiv.org/abs/2112.03227>.
- [50] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, W. Ai, B. Martinez, H. Yin, M. Lingelbach, M. Hwang, A. Hiranaka, S. Garlanka, A. Aydin, S. Lee, J. Sun, M. Anvari, M. Sharma, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, Y. Li, S. Savarese, H. Gweon, C. K. Liu, J. Wu, and L. Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024.
- [51] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment, 2019. URL <https://arxiv.org/abs/1909.12271>.
- [52] S. Zhang, Z. Xu, P. Liu, X. Yu, Y. Li, Q. Gao, Z. Fei, Z. Yin, Z. Wu, Y.-G. Jiang, and X. Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks, 2024. URL <https://arxiv.org/abs/2412.18194>.

A Appendix

A.1 Design Choice

Here we further discuss the design choices of FLOWEVAL, including why pure noise is chosen over noisy actions; the first diffusion step is chosen over the rest; and cosine similarity and velocity error are chosen over action MSE.

A.1.1 Velocity Error vs. Validation loss

Validation loss is used to probe performance [37, 26, 27], yet flow matching introduces another complicity in that the input noise can either be infused by ground truth action or not. We tested both conditions on four real-world Franka tasks, results in Fig. 4 and Table. 3. Results show that pure noise yields notably more stable results than noisy actions, suggesting that it provides a more stable and reliable evaluation signal.

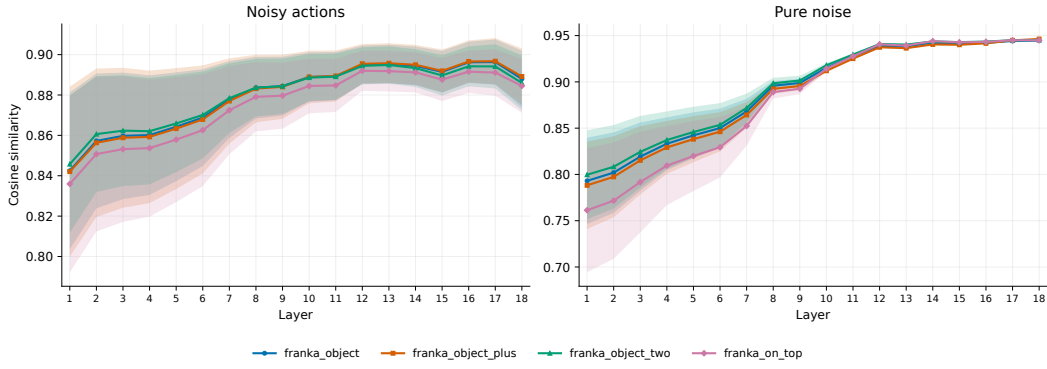


Figure 4: Cosine similarity per layer, evaluated on Pi-05. Left: noisy actions, right: pure noise.

Table 3: Statistics under noisy-action and pure-noise conditions on 4 collected Franka tasks.

Task	Condition	Mean	Variance
Franka Object	Noisy actions	3.121	3.390
	Pure noise	1.770	0.504
Franka Object+	Noisy actions	3.089	3.286
	Pure noise	1.704	0.492
Franka Object Two	Noisy actions	3.198	3.274
	Pure noise	1.775	0.513
Franka On Top	Noisy actions	3.123	3.017
	Pure noise	1.762	0.540

A.1.2 Distinguishability Scores

We include the full results used to generate Table 1 in Section 3.2. We ran 5 tasks on DreamZero-DROID, 16 diffusion steps, and 17 tasks on Pi-05-base, 10 diffusion steps. The tasks come from

Table 4: Distinguishability score of Action MSE compared with FLOWEVAL METRICS

Model	Action Cosine	Action VE	Action MSE	MSE / Cosine	MSE / VE
$\pi_{0.5}$	0.816	1.07	1.10	$1.35\times$	$1.03\times$
DreamZero	0.498	0.267	0.179	$0.359\times$	$0.670\times$

Table 5: DreamZero 16 diffusion steps

Step	0	1	2	3	4	5	6	7
Cosine (Action)	0.498	0.455	0.398	0.398	0.398	0.398	0.323	0.323
VE (Action)	0.267	0.244	0.205	0.205	0.205	0.205	0.144	0.144
Cosine (Video)	1.17	0.799	0.799	0.799	0.799	0.799	1.07	1.07
VE (Video)	0.956	0.735	0.723	0.723	0.723	0.723	1.14	1.14
Step	8	9	10	11	12	13	14	15
Cosine (Action)	0.323	0.323	0.231	0.231	0.231	0.224	0.234	0.231
VE (Action)	0.144	0.144	0.130	0.130	0.130	0.158	0.170	0.180
Cosine (Video)	1.07	1.07	0.897	0.897	0.897	0.845	0.846	0.850
VE (Video)	1.14	1.14	0.906	0.906	0.906	0.830	0.830	0.854

Table 6: Pi-05 10 diffusion steps

Step	0	1	2	3	4	5	6	7	8	9
Cosine	0.816	0.868	0.862	0.875	0.886	0.869	0.858	0.847	0.836	0.833
VE	1.07	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.09

the 17 manipulation tasks within the Robocasa365 target dataset [47]. DreamZero-DROID requires significantly more computing resources than Pi-05, so we were unable to evaluate it on the full set.

Scores for every diffusion step are included in Table. 6 and 5, while advantage of choosing Action MSE over velocity error and cosine similarity is expressed by the division values in Table. 4. Choosing action MSE provides little gain for Pi-05, while causing a significant drop for DreamZero.

A.1.3 Comparison with Action MSE

Action MSE is a competitive baseline used to determine performance [37]. It requires computing the whole diffusion process and thus can be up to 16 times more computationally inefficient, depending on the model configurations. However, faster computation does not necessarily translate to a better practical signal, raising the question of whether there is sufficient justification for designing FLOWEVAL around the first diffusion step instead of action MSE.

We test from two perspectives (1) whether action MSE provide better distinguishability over a set of tasks compared with cosine similarity and velocity error, and (2) whether using action MSE instead of cosine similarity and velocity error changes our core results. We discovered that choosing action MSE provides little gain for Pi-05, while causing a significant drop for DreamZero.

However, distinguishability alone only lets us select a metric that’s fairly separable (expressive), on top of that we still need practical signals showing the separation is actually reflective of the properties we care about. We therefore further evaluate whether the proposed FLOWEVAL metrics reflect the downstream behavior we ultimately aim to predict. In particular, we show that FLOWEVAL performs comparably to action MSE in correlating with how task success improves as additional adaptation data is provided (Figure. 5). Action MSE had a p value of ..., cosine similarity..., velocity error..

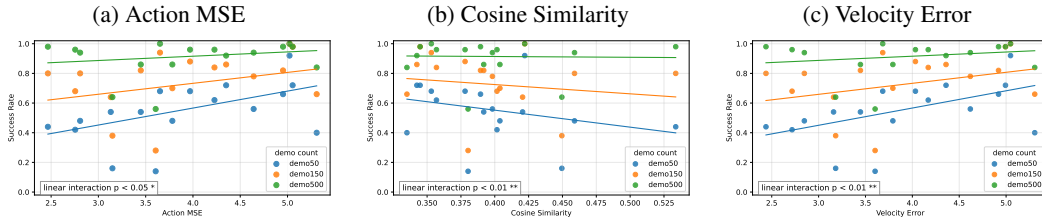


Figure 5: Regression results in Sec.4.2, recomputed with action MSE (left).

A.2 Limitation: Post transformed demos

A reasonable limitation of FLOWEVAL is that it is designed for real human-teleoperated demonstrations, where 1) the evaluated action targets are the actions actually issued during data collection and 2) the demonstration was obtained with an actual rollout. Post-hoc transformed demonstrations do not necessarily satisfy this assumption and therefore may not be valid inputs to the metric. We do not attempt to enumerate every possible transformation. Instead, we present one concrete counterexample in which FLOWEVAL does not apply reliably: demonstrations whose action targets are reconstructed after collection.

We collect the Franka demonstrations using a Cartesian-action controller (Section ??). Given that Pi-05, MolmoAct2, GR00T-N1.6 and DreamZero pretrained real world Franka data in action spaces of delta joint-position, absolute joint-position, relative joint-position and relative joint-position respectively (Table. 7), to evaluate these checkpoints within the same action representation they’re pretrained to predict, we reconstruct joint-space action targets from the Cartesian demonstrations after collection. This reconstruction creates a qualitatively different target from the original demonstration.

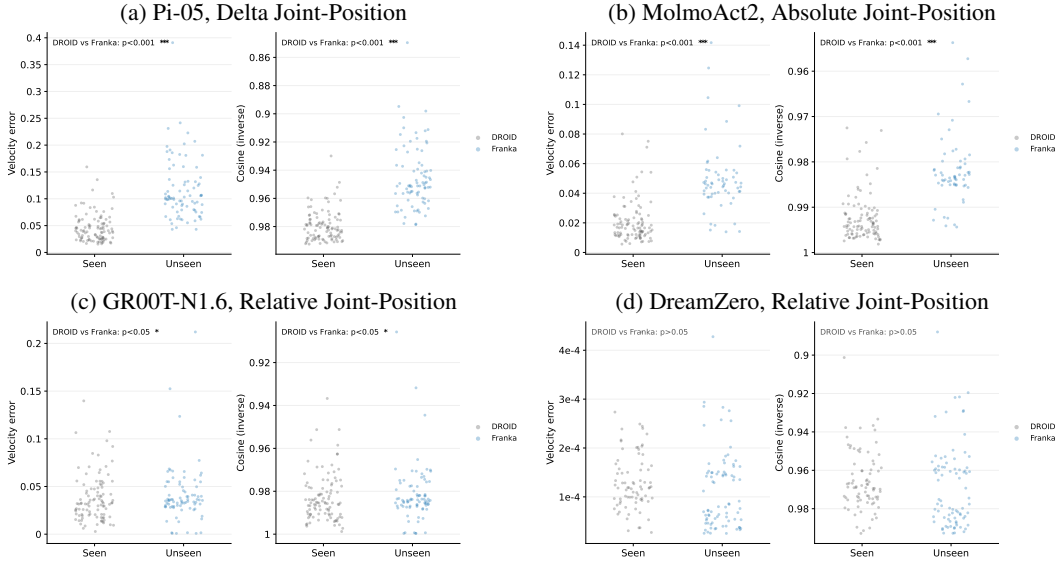


Figure 6: FlowEval results across various models and action representations on the reconstructed franka dataset.

We observe exactly this failure mode. On the reconstructed Franka trajectories, the Franka action targets are concentrated close to the hold-position action, causing the DROID reference and Franka deployment to overlap much more strongly than for the other unseen deployments. DreamZero shows no significant separation ($p > 0.05$), while GR00T-N1.6 shows only weaker separation ($p < 0.05$), compared with $p < 0.001$ on the other unseen target deployments. Upon further investigation (Fig. 7), we find that DreamZero’s mean full-action error is 1.2313×10^{-4} , whereas simply predicting the hold-position action gives a lower error of 5.6037×10^{-5} . The paired difference is 6.7093×10^{-5} , with a task-bootstrap 95% interval of $[4.1594, 9.4574] \times 10^{-5}$.

This failure mode is specific to post-hoc reconstruction of the action labels. In ordinary robot-learning pipelines, converting demonstrations to another control interface would typically require recollecting actions under the desired controller, rather than substituting the next measured robot state as the action target. We therefore do not view this result as limiting the use of FLOWEVAL on ordinary deployment adaptations. Instead, it shows that FLOWEVAL should not be applied to post-reconstructed action targets without verifying that the reconstruction preserves the demonstrated control signal. More broadly, this example also illustrates why post-hoc reconstructed targets of this

form can be unsuitable supervision for robot learning: they can collapse toward nearly stationary achieved states even when the original demonstration contains meaningful control commands.

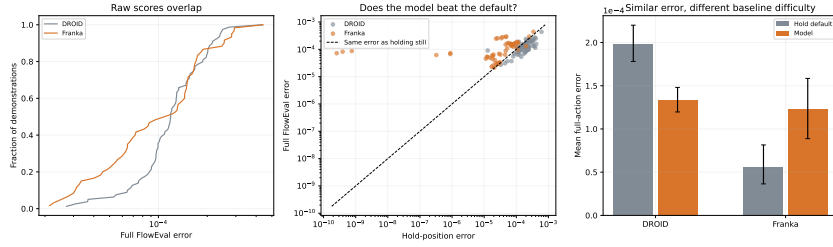


Figure 7: DreamZero comparison with hold still baseline on Franka and DROID.

A.3 Predicting unseen deployments

Seen deployments always have the same embodiment and action representation as one of the pre-trained checkpoints. If an unseen deployment has the same embodiment and action representation as one of the pretrained checkpoints already support, we select the corresponding checkpoint and action representation space (Table. 7). If an unseen deployment does not match any of the configurations, we use the action representation mapping rule specified in Appendix. A.4.

Table 7: Action channels the checkpoints are pretrained on [2, 7, 13, 8].

Checkpoint	Action Channel
DreamZero-DROID	Relative joint position
GR00T-N1.6-DROID	Relative joint position
GR00T-N1.5-DROID	Absolute joint position; joint velocity
GR00T-N1.5	Delta Cartesian
Pi-0.5-Base	Delta joint position
Pi-0.5-DROID	Joint velocity
Pi-0.5-RoboCasa	Delta Cartesian
MolmoAct2	Mixed
MolmoAct2-DROID	Absolute joint position
MolmoAct2-LIBERO	Delta Cartesian

A.3.1 Statistical tests

Results for both matched and mismatched action representation spaces are shown in 8. The matched condition is a subset reported in the main paper (Sec. 4.1). In all cases, unseen deployments are statistically distinguishable ($p < 0.001$). We report the p values of the one-sided two-sample Kolmogorov–Smirnov (KS) test in Table. 8.

A.4 Action Representation Mapping

When comparing a pretrained model with a deployment setting that uses a different action representation, the model and dataset dimensions may not align directly. In such cases, we follow the mapping rules below.

Some mappings may appear unintuitive because the paired dimensions do not share the same physical units. However, the goal of FLOWEVAL is to probe the adaptation gap between the model prediction and the downstream ground truth. For example, if a model is pretrained to predict joint positions while the downstream objective uses joint velocities, adaptation must still learn the corresponding distribution shift. We therefore pair joint positions with joint velocities in such cases.

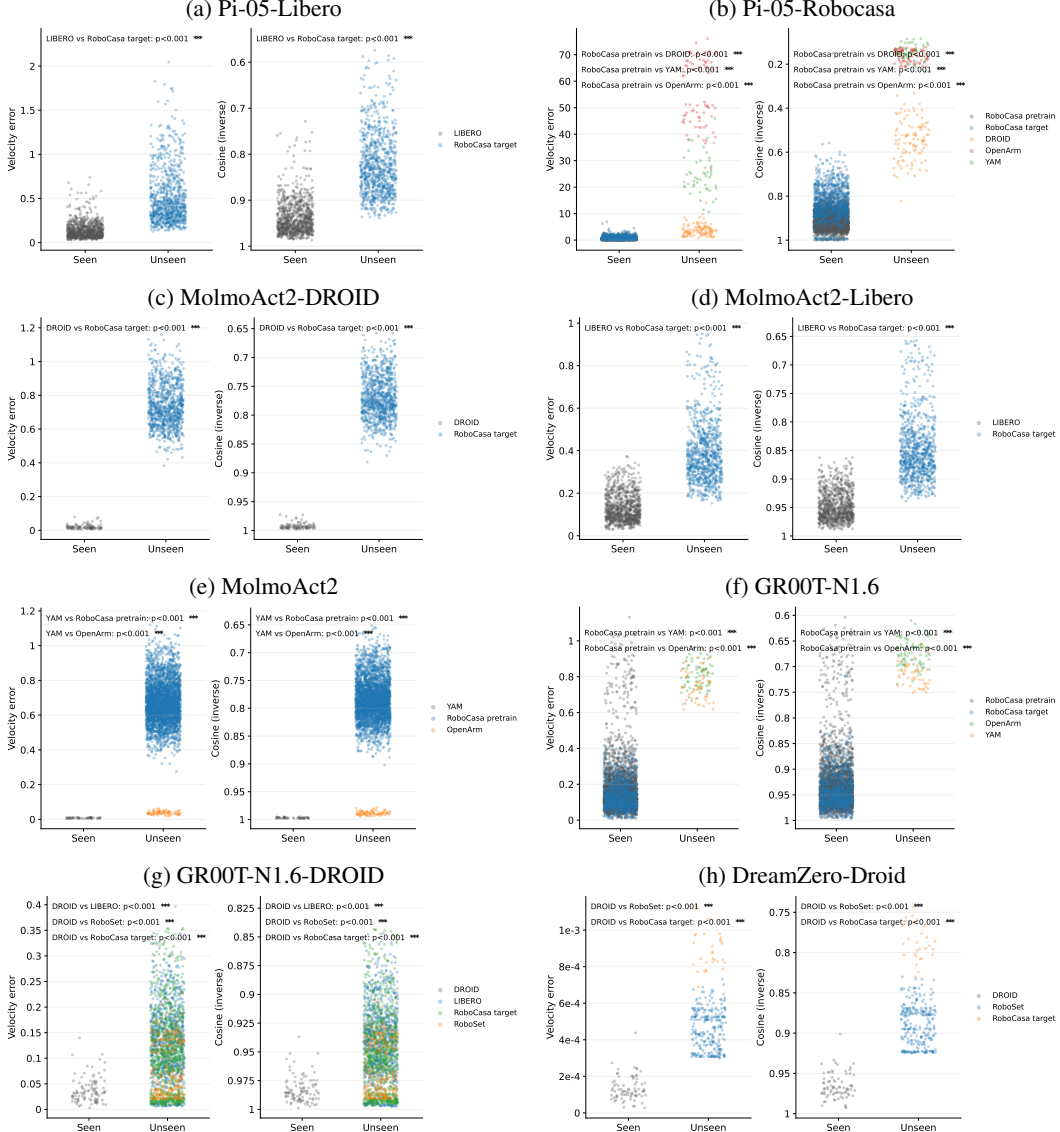


Figure 8: Additional results of FLOWEVAL rating unseen deployment settings.

Semantic mapping is used to identify the lowest-friction correspondence for learning this distribution shift. It does not eliminate or override the shift itself; rather, it provides the simplest correspondence through which that shift can be learned. Accordingly, gripper dimensions are mapped to gripper dimensions, arm dimensions to arm dimensions, and base dimensions to base dimensions.

1) Valid dataset dimensions determine the dimensions involved in the loss computation.

- This dimension in the dataset should be valid and non-trivial. Examples include joint positions or gripper states. Counter-examples include dimensions that remain a constant, such as the torso lift dimension in robocasa.
- Each valid dataset dimension require a corresponding model output dimension. When $\text{len}(\text{dataset}) > \text{len}(\text{valid_model_output})$, where $\text{valid_model_output}$ means the dimensions in the model that's activated (actually trained with much data), always take the 'unused' next model output dimensions to fill in the slot. When $\text{len}(\text{dataset}) < \text{len}(\text{valid_model_output})$, discard some model outputs so that they map one by one.

Table 8: Two-sample Kolmogorov–Smirnov tests comparing FLOWEVAL score distributions on seen-reference and unseen datasets. All tested unseen deployments are statistically distinguishable from their corresponding seen-reference distributions.

Checkpoint	Seen reference	Unseen deployment	KS statistic	p -value	Sig.
Velocity error					
Pi-0.5-RoboCasa	RoboCasa-pretrain	DROID	0.940	3.88×10^{-114}	***
Pi-0.5-RoboCasa	RoboCasa-pretrain	MolmoAct2-YAM	1.000	2.57×10^{-112}	***
Pi-0.5-RoboCasa	RoboCasa-pretrain	OpenArm	1.000	4.05×10^{-162}	***
Pi-0.5-LIBERO	LIBERO	RoboCasa-target	0.775	6.08×10^{-229}	***
MolmoAct2	MolmoAct2-YAM	RoboCasa-pretrain	1.000	5.24×10^{-112}	***
MolmoAct2	MolmoAct2-YAM	OpenArm	1.000	2.42×10^{-36}	***
MolmoAct2-DROID	DROID	RoboCasa-target	1.000	1.57×10^{-140}	***
MolmoAct2-LIBERO	LIBERO	RoboCasa-target	0.825	1.29×10^{-259}	***
GR00T-N1.6	RoboCasa-pretrain	MolmoAct2-YAM	0.971	8.80×10^{-73}	***
GR00T-N1.6	RoboCasa-pretrain	OpenArm	0.973	2.40×10^{-110}	***
GR00T-N1.6-DROID	DROID	LIBERO	0.743	7.24×10^{-51}	***
GR00T-N1.6-DROID	DROID	RoboSet	0.438	3.93×10^{-13}	***
GR00T-N1.6-DROID	DROID	RoboCasa-target	0.600	2.76×10^{-31}	***
DreamZero-DROID	DROID	RoboSet	0.987	3.00×10^{-73}	***
DreamZero-DROID	DROID	RoboCasa-target	1.000	5.97×10^{-27}	***
Cosine similarity					
Pi-0.5-RoboCasa	RoboCasa-pretrain	RoboCasa-target	0.566	9.89×10^{-198}	***
Pi-0.5-RoboCasa	RoboCasa-pretrain	DROID	0.988	1.23×10^{-161}	***
Pi-0.5-RoboCasa	RoboCasa-pretrain	MolmoAct2-YAM	1.000	1.31×10^{-110}	***
Pi-0.5-RoboCasa	RoboCasa-pretrain	OpenArm	1.000	3.24×10^{-160}	***
Pi-0.5-LIBERO	LIBERO	RoboCasa-target	0.779	2.95×10^{-231}	***
MolmoAct2	MolmoAct2-YAM	RoboCasa-pretrain	1.000	5.24×10^{-112}	***
MolmoAct2	MolmoAct2-YAM	OpenArm	1.000	2.42×10^{-36}	***
MolmoAct2-DROID	DROID	RoboCasa-target	1.000	1.57×10^{-140}	***
MolmoAct2-LIBERO	LIBERO	RoboCasa-target	0.828	1.26×10^{-261}	***
GR00T-N1.6	RoboCasa-pretrain	MolmoAct2-YAM	0.974	2.60×10^{-74}	***
GR00T-N1.6	RoboCasa-pretrain	OpenArm	0.976	3.27×10^{-113}	***
GR00T-N1.6-DROID	DROID	LIBERO	0.737	8.57×10^{-50}	***
GR00T-N1.6-DROID	DROID	RoboSet	0.416	7.02×10^{-12}	***
GR00T-N1.6-DROID	DROID	RoboCasa-target	0.593	1.68×10^{-30}	***
DreamZero-DROID	DROID	RoboSet	0.987	3.00×10^{-73}	***
DreamZero-DROID	DROID	RoboCasa-target	1.000	5.97×10^{-27}	***

2) Same semantic dimensions should map to each other.

- Grippers should be mapped with grippers, arms should be mapped with arms, and bases should be mapped with bases.
- For bimanual to single arm mapping, map the single arm to one of the bimanual arms.

3) Arms should be mapped to arms.

- Dataset dimensions decide the dimensions involved: If dataset has 6D arm and the model has 7D arm, discard the last arm dim in model and only include 6 dims in the loss computation; if dataset has 7D arm and the model has 6D arm, use the next 'unused' dimension (note that this may not be the immediate following dimension, since the model may have other outputs e.g. base/gripper after the arm. It should be the next 'unused' not associated with a semantic under the config) to map with the last arm dim in the dataset and include 7 dims in the loss computation.

A.5 Predicting downstream adaptation

A.5.1 Control 1: Deployment shift

In Sec. 4.2 we observe significant interactions between success rate gain and how much additional data is provided, indicating that FLOWEVAL provides useful signal about how performance in an unseen deployment scales with additional data. This control examines whether this signal reflects a generic scaling effect or is specific to deployment shift. To this end, we additionally finetune Pi05-RoboCasa and GR00T-N1.6 (without shift) and compare their results with Pi05-base and GR00T-N1.5 (with shift) on the RoboCasa tasks. All models use 50, 150, and 500 demonstrations except GR00T-N1.5, for which we use 50 and 150 (Appendix. 9).

Shift	Model	Demos	Cosine similarity	Velocity error / loss
With	Pi05-base	50/150/500	+0.474 (.0047)	−0.0382 (.0093)
With	GR00T-N1.5	50/150	−5.537 (1.81×10^{-8})	+2.363 (3.12×10^{-8})
Without	Pi05-RoboCasa	50/150/500	+0.275 (.3112)	−0.0382 (.1056)
Without	GR00T-N1.6	50/150/500	+0.266 (.4521)	−0.1102 (.4464)

Table 9: Initial-metric $\times$ demonstration-budget interaction coefficients, with p -values in parentheses. Bold denotes $p < .05$.

Both metrics show significant interactions with demonstration budget under deployment shift, whereas neither is significant without shift. These results support the interpretation that the interaction arises from deployment shift rather than a generic scaling effect.

Deployment shift	Model	Cosine similarity	Velocity error / loss
With	Pi05-base	+0.474 (.0047)	−0.0382 (.0093)
With	GR00T-N1.5	−0.0073 (.9964)	−0.0258 (.9699)
Without	Pi05-RoboCasa	+0.275 (.3112)	−0.0382 (.1056)
Without	GR00T-N1.6	+0.266 (.4521)	−0.1102 (.4464)

Table 10: Initial-metric $\times$ demonstration-budget interaction coefficients, with p -values in parentheses; bold denotes $p < .05$. Positive cosine coefficients associate higher initial cosine similarity with larger gains; negative loss coefficients associate lower initial loss with larger gains.

A.5.2 Control 2: Shared Task Effects

One alternative explanation for the observed interaction is shared task effects instead of model-deployment fit. If some tasks are consistently easier to learn, and these differences in task learnability align with FLOWEVAL scores, they could induce a correlation with scaling. In short, we want to distinguish whether FLOWEVAL measures a merely task level effect or a model-deployment interaction effect.

We test this possibility through cross-model task consistency. We compare Pi05-Base, Pi05-RoboCasa, GR00T-N1.5, and GR00T-N1.6. We compute pairwise Pearson correlations across shared tasks, with Holm correction across the six model pairs for each outcome. We additionally test task effects on success after accounting for model using a partial F-test, and test cross-model metric consistency using Bartlett’s test of sphericity.

FLOWEVAL scores show limited evidence of cross-model task consistency. No pairwise cosine-similarity correlations remain significant after Holm correction, and only one velocity-error correlation does, between Pi05-Base and Pi05-RoboCasa ($r = 0.634$, adjusted $p = 0.0374$). Bartlett’s tests do not reject identity for either metric’s cross-model correlation matrix ($p = 0.0854$ for cosine similarity and $p = 0.0945$ for velocity error). These results provide little evidence that FLOWEVAL consistently ranks the same tasks similarly across models, a pattern compatible with model-task-specific variation. Therefore, FLOWEVAL reasonably measures the model-deployment interaction effect we hypothesized.

Table 11: Cross-model task consistency of success and FlowEval metrics. Strong success consistency confirms shared task effects in RoboCasa; corresponding FlowEval consistency would indicate that these effects could induce the FlowEval–performance relationship.

Cross-model pairwise results		Global test
Cosine similarity	0/6 pairs after Holm correction	Bartlett $p = 0.0854$
Velocity error	1/6 pairs: Pi05-Base/Pi05-RoboCasa, $r = 0.634$, adjusted $p = 0.0374$	Bartlett $p = 0.0945$

Table 12: Control for baseline success in the FlowEval scaling analysis. Reported coefficients are the FlowEval metric $\times \log(\text{demos})$ interaction β_3 . The controlled specification additionally includes 50-demo success S_{50} and its interaction with $\log(\text{demos})$. Coefficient magnitudes should be compared before and after adjustment within each metric rather than across metrics.

Model	Demos	Metric	β_3	95% CI	p
Pi05-Base	50/150/500	Cosine similarity (Original)	0.4738	[0.1455, 0.8022]	0.00468
		Cosine similarity (Controlled)	0.2595	[−0.04683, 0.5659]	0.09684
Pi05-Base	50/150/500	Velocity error (Original)	−0.03820	[−0.06699, −0.00942]	0.009283
		Velocity error (Controlled)	−0.02106	[−0.04391, 0.001801]	0.07099
GR00T-N1.5	50/150	Cosine similarity (Original)	−5.537	[−7.465, −3.609]	1.81×10^{-8}
		Cosine similarity (Controlled)	−5.468	[−7.563, −3.373]	3.12×10^{-7}
GR00T-N1.5	50/150	Velocity error (Original)	2.363	[1.526, 3.200]	3.12×10^{-8}
		Velocity error (Controlled)	2.337	[1.433, 3.241]	4.07×10^{-7}

A.5.3 Control 3: Headroom bias

Another alternative explanation for the observed interaction is baseline success. Because success is bounded at 100%, a task starting at 20% success has more room to improve than a task starting at 90%. If FLOWEVAL is correlated with starting performance, part of its interaction with demonstration count could arise from this difference in available headroom rather than from an association with scaling beyond starting performance.

We therefore control for each task’s measured success at 50 demonstrations and, critically, allow the scaling slope itself to depend on this baseline success. This asks whether FLOWEVAL remains associated with scaling after accounting for the possibility that tasks with different starting success rates have different rates of improvement.

Control specification. For each task, let R denote success rate at a given demonstration count, $L = \log(N)$ the natural logarithm of the number of demonstrations, m the task-level step-zero FlowEval metric, and S_{50} the task’s success rate with 50 demonstrations. The original regression is

$$R = \beta_0 + \beta_1 L + \beta_2 m + \beta_3 Lm + \epsilon, \quad (6)$$

where β_3 is the FlowEval metric $\times \log$ -demonstration-count interaction examined in Section 4.2.

The baseline-success-controlled regression is

$$R = \beta_0 + \beta_1 L + \beta_2 m + \beta_3 Lm + \beta_4 S_{50} + \beta_5 LS_{50} + \epsilon. \quad (7)$$

The S_{50} term accounts for differences associated with starting success, while LS_{50} allows the relationship between demonstration count and success to vary with starting success. Including only S_{50} would therefore not directly address the alternative explanation for the scaling interaction. The comparison of interest is how the FlowEval interaction β_3 changes after adding both terms.

L , m , and S_{50} are centered at their cell-level means. The original and controlled regressions use identical observations. No headroom-normalized gain or separate gain regression is introduced, so this analysis directly tests the original interaction under an additional baseline-success control.

Results. For Pi05-Base, controlling for baseline success reduces the cosine and velocity-error interactions by 45.2% and 44.9%, respectively, while preserving their directions and approximately

55% of their original magnitudes. The negative LS_{50} coefficients ($p = 0.00184\text{--}0.00435$) indicate that tasks with higher baseline success tend to gain less from additional demonstrations. This suggests that a ceiling effect contributes to the observed scaling trend but does not fully explain it.

For GR00T-N1.5 over 50–150 demonstrations, controlling for baseline success leaves the cosine and velocity-error interactions almost unchanged. The cosine coefficient changes from -5.537 to -5.468 (1.24%), and the velocity-error coefficient changes from 2.363 to 2.337 (1.11%).

A.5.4 Training Configurations

We use these training configurations for Pi05-Base, Pi05-Robocasa, GR00T-N1.5, and GR00T-N1.6.

OpenPi Hyperparameters

```
global_batch_size: 64
train_steps: 30000

optimizer: AdamW
beta1: 0.9
beta2: 0.95
epsilon: 1e-8
weight_decay: 1e-6
grad_clip_global_norm: 1.0

lr_schedule: warmup-cosine
warmup_steps: 1000
peak_lr: 2.5e-5
decay_steps: 30000
end_lr: 2.5e-6

ema_decay: 0.99
```

GR00T Hyperparameters

```
learning_rate: 1e-4
weight_decay: 1e-5
warmup_ratio: 0.05

global_batch_size: 64
gradient_accumulation_steps: 1

shard_size: 1024
num_shards_per_epoch: 100000
episode_sampling_rate: 0.1

optim: adamw_torch

tune_llm: true
tune_visual: true
tune_projector: true
tune_diffusion_model: true

state_dropout_prob: 0.0
random_rotation_angle: null
extra_augmentation_config: null
```

```

load_bf16: false
reproject_vision: false
eagle_collator: true
backbone_trainable_params_fp32: true
use_relative_action: true

```

A.5.5 Training Collapse in GR00T-N1.5 with 500 Demonstrations

Loss Spikes. We train each GR00T-N1.5 finetuning run for 30k steps to allow sufficient optimization. Around 5k steps, when the learning-rate schedule plateaus, we observe loss spikes and occasional training instability. Performance collapse affects 5 of 17 runs (29.4%) at 500 demonstrations (Fig. 9), compared with one GR00T-N1.5 run at 150 demonstrations and one or two runs across all demonstration counts for the Pi-05 variants (Table 13). We therefore exclude the 500-demonstration condition from the primary GR00T-N1.5 regression analysis and report the interaction over 50–150 demonstrations.

Table 13: Runs exhibiting persistent or sudden performance collapse.

Model	Demonstrations	Collapsed runs	Fraction of runs
GR00T-N1.5	500	5	29.4% (5/17)
GR00T-N1.5	150	1	5.88% (1/17)
GR00T-N1.6 (Action Expert)	All	0	0.00% (0/51)
Pi-05-Base	All	1	1.96% (1/51)
Pi-05-RoboCasa	All	2	3.92% (2/51)



Figure 9: Performance curves for GR00T-N1.5 finetuning. Training collapse occurs on task 18 with 150 demonstrations and on tasks 2, 10, 12, 16, and 17 with 500 demonstrations.