RESIDUAL MODELING CLOSES THE REGRESSION AND GENERATIVE POLICY GAP IN ROBOT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning from demonstration has enabled impressive robot behaviors. A common choice for policy learning is to use diffusion or flow matching (Flow-Policies), which often outperforms direct action regression trained with mean squared error (MSE-Policies). This gap is commonly attributed to multimodal demonstrations. We revisit this gap from the perspective of statistical modeling: how action-prediction residuals shape policy optimization. Our analysis of real-world robot demonstration data reveals substantial state-dependent variation in residual scales and heavier-than-Gaussian tails. While both MSE-Policies and Flow-Policies exhibit heavy-tailed action residuals, their training gradients behave differently: MSE allocates more gradient magnitude to observations with large action residuals, which hurts optimization. Motivated by these findings, we introduce heteroscedastic Student-t action regression (HT-Policies), which learns input-dependent residual scales and reduces the influence of heavy tails. HT-Policies predict action chunks with a single feed-forward pass and can reuse pretrained flow-matching-based policy networks as the backbone. Across four simulation benchmarks and real-robot evaluations, HT-Policies achieves success rates competitive with generative policy baselines, both when trained from scratch and from pretrained vision-language-action and world-action models, despite being faster in training and inference. Together, these findings shed light on the practical advantages of generative objectives in robot learning from demonstrations and offer an efficient direct-regression alternative for a range of architectures and tasks.

1 INTRODUCTION

Learning robot policies from demonstrations is typically formulated as a supervised learning problem, in which an action chunk — a short sequence of consecutive actions — is predicted from observations and goal specifications (Mandlekar et al., 2021). A common approach for policy learning is to use generative models, such as diffusion and flow-matching models (Chi et al., 2023; Pan et al., 2026). In practice, these models often outperform direct action regression trained with mean squared error (MSE-Policies). This performance gap is commonly attributed to multimodality: generative policies can model multiple action modes, whereas MSE-Policies tend to average across modes, potentially producing actions that do not correspond to any demonstrated behavior.

Recent results suggest that explicitly modeling multimodal action distributions is not necessary for competitive performance. Minimal Iterative Policy (MIP), introduced by Pan et al. (2026), uses a *deterministic* network in two successive passes for action prediction. The first pass predicts an action chunk conditioned only on the policy inputs; the second pass additionally conditions on the first prediction to refine it. During training, the network is supervised with a denoising objective. MIP closes the performance gap with Flow-Policies and attributes its success to iterative refinement, which helps keep predicted actions on the *manifold* of demonstrated actions. However, this explanation remains largely empirical, with limited formal characterization and real-data verification.

In this paper, we revisit this problem from a statistical modeling perspective. Specifically, we examine the residual models implicit in different training objectives and how they affect policy optimization. This perspective offers a way to understand the benefits of denoising supervision and motivates a direct action regression objective that can resemble those benefits without iterative inference.

We begin by examining the action residuals of trained policies. From a likelihood perspective, MSE regression corresponds to a fixed-scale Gaussian residual model. Empirically, however, both

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

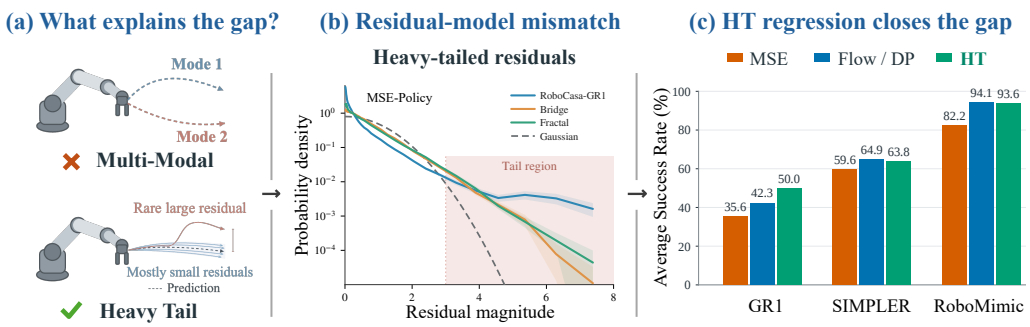

Figure 1: **(a)** We attribute the performance gap between MSE- and Flow-Policies to different statistical modeling of prediction residuals. **(b)** We found observation-dependent and heavy-tailed residuals for MSE-Policies on multiple benchmarks. **Right:** Benchmark success rates (%) for different policy classes. We use flow-matching for RoboCasa-GR1 and SIMPLER, and diffusion for RoboMimic.

MSE-Policies and Flow-Policies exhibit heavy-tailed action residuals. One source of these tails is variation in residual scale across observations: for example, reaching toward a hole and making fine adjustments during insertion may involve different prediction-error scales. Yet scale variation alone does not fully account for the observed tails, which remain heavier than Gaussian after per-observation scale normalization, as illustrated in Figure 1b. This structure matters for optimization because the gradient of MSE with respect to the predicted action is proportional to the residual. We show that high-residual samples account for a dominating share of gradient magnitude under MSE. Flow matching exhibits a similar concentration at high noise levels, but distributes gradient magnitude more evenly, especially at low noise levels due to its denoising training objective.

Motivated by these findings, we introduce *heteroscedastic Student-t* action regression (HT-Policies) (Lange et al., 1989). HT-Policies incorporate the observed residual structure directly into the regression objective, predict action chunks with a single feed-forward pass, and can reuse pre-trained flow-matching policy networks as backbones. In four simulation benchmarks and real-robot evaluations, HT-Policies achieve success rates competitive with generative policy baselines. They perform well both when trained from scratch and when fine-tuned from pretrained vision-language-action and world-action models, while improving training and inference efficiency. Our gradient analysis shows reduced concentration of gradient magnitude on high-residual observations, supporting our hypothesis about the role of residual modeling in policy optimization. Together, these findings shed light on the practical advantages of generative objectives in robot learning and offer practitioners an efficient direct-regression alternative across a range of architectures and tasks. To summarize, our contributions are as follows:

- We characterize robot-policy residuals, revealing input-dependent scales and heavy tails that expose a mismatch with the fixed-scale Gaussian assumption of MSE.
- We further show that this mismatch causes MSE to overemphasize large-action-residual samples, whereas flow matching distributes gradients more evenly, especially in low-noise regimes. We explain the mechanism of this by studying the denoising objectives.
- We introduce HT-Policies, which explicitly model input-dependent scales and heavy tails. Across simulation and real-robot tasks, they match generative baseline performances while enabling more efficient training and inference.

2 Related Work

Variations in robot demonstration data. Human demonstrations exhibit variation arising from multiple sources, including differences in task strategies (Shafiullah et al., 2022; Gandhi et al., 2023; Jia et al., 2024), action rates (Shi et al., 2026), and execution errors or inaccuracies (Chen et al., 2025). Prior work has addressed such variation from two main perspectives. One line of work treats it as distributional multimodality and models it with Gaussian mixtures (Mandlekar et al., 2021), diffusion models (Chi et al., 2023), or conditional VAEs (Zhao et al., 2023). Others treat variation as undesirable noise and seek to suppress it (Shi et al., 2026; Chen et al., 2025). While these approaches can accommodate diverse demonstrations, they largely study how variation should be modeled or filtered empirically, rather than understanding its statistical structure and how it affects learning.

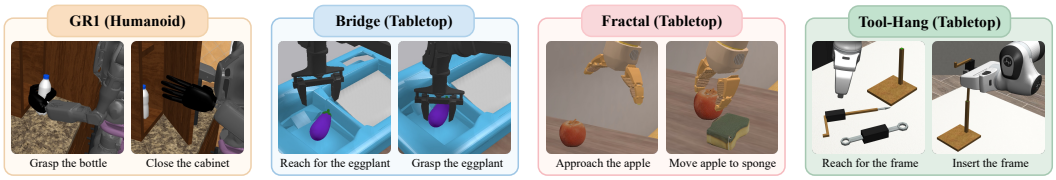

Figure 2: **Example tasks.** We show 4 tasks from RoboCasa-GR1, Bridge, Fractal, and Tool-Hang.

MIP (Pan et al., 2026) shows that averaging flow-policy outputs leaves performance largely unchanged, a finding we extend to VLA and WAM policies (Appendix A.2). Instead of avoiding mode averaging, it argues that flow's iterative refinement moves predictions toward the data manifold, but provides limited formal characterization. Moreover, its study of off-manifold error on out-of-distribution states does not distinguish manifold adherence from generalization. We revisit the gap by studying the statistical residual modeling in MSE-Policies and its effects on policy learning.

Robust and heteroscedastic regression. Huber losses (Huber, 1964) and Student-t likelihoods (Lange et al., 1989) improve regression robustness to large residuals; adaptive robust losses can learn the appropriate degree of robustness from data (Barron, 2019). Complementarily, heteroscedastic regression models input-dependent residual scales (Nix & Weigend, 1994), although variance-based reweighting under Gaussian likelihoods can adversely affect mean estimation (Seitzer et al., 2022a). These objectives have been widely explored in computer vision, including bounding-box regression (Girshick, 2015) and depth prediction (Kendall & Gal, 2017). In imitation learning, OpenVLA-OFT shows strong LIBERO performance with L_1 regression (Kim et al., 2025), although L_1 still underperforms MSE on three of four LIBERO suites when fine-tuning π_0 (Pan et al., 2026). Robust classification objectives have also been studied (Tangkaratt et al., 2021). We combine robust regression with input-dependent scale learning for action learning.

3 Residual Structure and Optimization

We investigate why regression-based policies perform worse than Flow-Policies from the perspective of statistical modeling: the modeling of prediction residuals and its impact on policy optimization. To conduct this study, we take a pretrained GR00T policy and fine-tune it with the MSE regression loss and the flow-matching loss on three benchmarks: RoboCasa-GR1 (NVIDIA et al., 2025), Bridge (Walke et al., 2023), and Fractal (Brohan et al., 2022). We also train a U-Net policy on RoboMimic Tool-Hang (Mandlekar et al., 2021). We collect per-sample predictions and parameter-gradient norms on the training set. Example tasks are visualized in Figure 2. Appendix A.1.1 provides details about these benchmarks and our data processing protocols.

Action residuals have heavy tails. Let o denote the policy inputs and $a \in \mathbb{R}^d$ a demonstrated action chunk. The action residual is $r = \hat{a}(o) - a$, where $\hat{a}(o)$ is predicted by an MSE-Policy or sampled from a Flow-Policy using random initial noise. We use $\mathrm{RMS}(r) = \|r\|_2/\sqrt{d}$ for observation-level analyses. To examine tail behavior, Figure 3 pools scalar residuals normalized by their coordinate RMS. Both policy classes exhibit heavier-than-Gaussian tails across the evaluated settings.

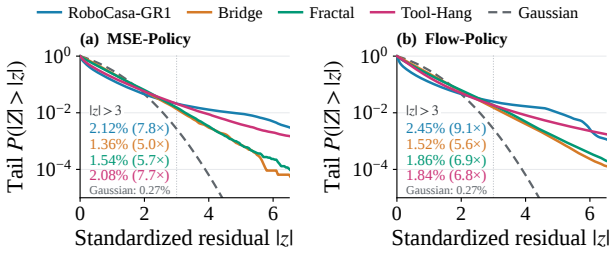

Figure 3: **Heavy-tailed residuals.** Both MSE-Policies **(a)** and Flow-Policies **(b)** exhibit heavy-tailed action-prediction residuals. Percentages report scalar residuals exceeding three times their coordinate RMS in absolute value.

From a statistical learning perspective, MSE corresponds to a Gaussian residual model with a fixed scale σ. Therefore, the results in Figure 3 suggest a real-data mismatch with this model. Next, we illustrate how this will contribute to the performance gap between MSE-Policies and Flow-Policies.

Misspecified residual models hurt optimization. For MSE-Policies, action samples with large residuals receive disproportionate weight during optimization. Such concentration may leave other action samples underfit. To quantify this effect, we measure per-data-point parameter-gradient

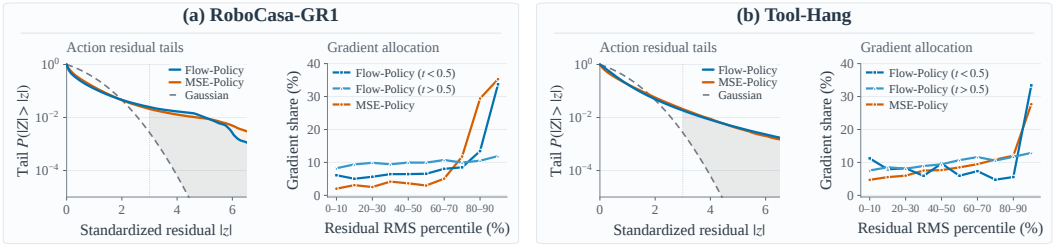

Figure 4: **Gradient distribution for different objectives and noise levels.** We show the relative gradient contribution (%) of each action-residual decile. For Flow, gradient RMS is computed separately for high-noise ($t < 0.5$) and low-noise ($t > 0.5$) regimes. See also the Appendix Figure 12.

norms. For each policy, we rank observations by action-residual RMS and divide them into ten equal-sized groups. For Flow-Policies, we compute the average gradient norms over different input noises and quantify gradients in both high-noise ($t < 0.5$) and low-noise ($t > 0.5$) regimes. Appendix A.1.2 provides additional details. Figure 4 shows the result: MSE and high-noise flow matching concentrate gradients on observations with large action residuals, whereas the low-noise flow-matching objective distributes them more evenly among groups.

Where do the heavy tails come from? We next examine where the heavy-tailed action residuals come from. One potential source is variation in residual scale across observations. Illustrated in Figure 5, residual RMS varies with both demonstrated action magnitude and episode progress. Even if actions follow a Gaussian distribution per observation, differences in their scales across observations can yield a mixture with heavier-than-Gaussian tails (Andrews & Mallows, 1974).

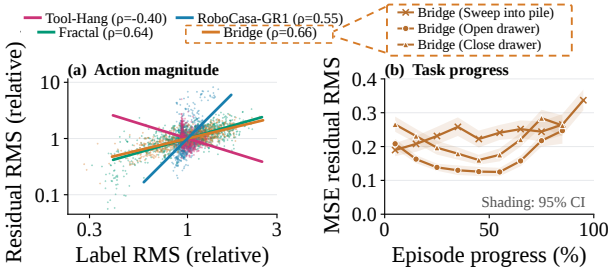

Figure 5: **Input-dependent residual scale.** We evaluate a GR00T N1.7 model trained with the MSE regression loss. **(a)** Residual vs. action RMS. **(b)** Residual RMS vs. episode progress on three Bridge tasks. Shaded bands indicate 95% confidence intervals..

Unfortunately, differences in scale do not fully explain the observed heavy tails. To see this, we normalize each action residual by a per-observation scale estimate and re-examine the tails. In particular, we train a model to jointly estimate the mean and variance of actions conditioned on the state using a heteroscedastic Gaussian (HG) objective (Nix & Weigend, 1994). We then normalize the residuals by the predicted standard deviations $\sigma(o)$. For Flow-Policies, we estimate $\sigma(o)$ from multiple actions sampled from the model. Appendix A.1.1 provides details. In both cases, normalization makes the tails lighter, but they remain heavier than Gaussian (Figure 6).

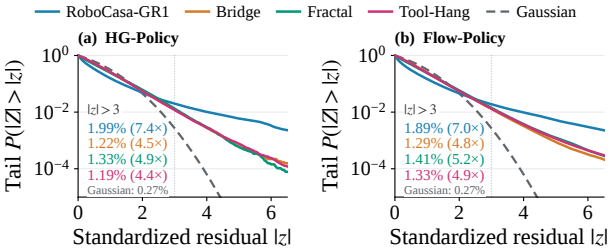

Figure 6: **Heavy tails after normalization.** **(a)** HG-regression residuals normalized by predicted $\sigma(o)$; **(b)** flow-matching residuals normalized by per-observation action sampling.

How flow-matching handles heavy-tailed residuals. To understand the mechanism of more balanced gradients in flow-matching, we examine its denoising-based training objective. The intuition is that a flow-matching network conditions on a noisy demonstrated action. This additional input allows the training residual to have light tails even when action-prediction residuals are heavy-tailed.

Recall that a flow-matching network takes the policy inputs o, a timestep t, and a noisy action $a_t = ta + (1-t)\epsilon$, where $\epsilon \sim \mathcal{N}(0, I_d)$, and minimizes the squared velocity residual $v_\theta(a_t, t, o) - (a - \epsilon)$. For any fixed $t > 0$, consider a naïve baseline predictor $\tilde{v}(a_t, t, o) = a_t/t$. Its velocity residual is

$$\tilde{v}(a_t, t, o) - (a - \epsilon) = \frac{a_t}{t} - a + \epsilon = \frac{\epsilon}{t}.$$

For this baseline model, its network prediction residual is distributed as $\mathcal{N}(0, I_d/t^2)$, regardless of the overall action distribution. Therefore, the flow-matching objective admits a predictor whose residual scale depends only on the injected noise and the timestep.

Empirically, we found that a well-trained predictor that is conditioned on o also inherits this behavior in the sense that, at low-noise timesteps, the learned predictor has tails no heavier than this reference:

$$\Pr[\|v_\theta(a_t, t, o) - (a - \epsilon)\|_2 > \delta \mid t] \leq \Pr[\|\epsilon\|_2 > t\delta] \quad \text{for sufficiently large } \delta.$$

Appendix A.1.3 provides empirical evidence on different benchmarks. The denoising objective can tolerate heavy-tailed action residuals because the conditioning of noisy actions allows the velocity residual to be light-tailed. The reference scale $1/t$ decreases as the noise level decreases, which explains the observation of more balanced gradient distributions in low-noise regimes.

4 Heteroscedastic Student-t Action Regression

Section 3 shows that action residuals have input-dependent scales and heavy tails. MSE-Policies do not account for this modeling and concentrate gradient magnitude on samples with large residuals. We therefore introduce a heteroscedastic Student-t objective for policy learning (HT-Policies). HT-Policies model action residuals with a Student-t distribution whose scale is predicted from the input and whose degrees of freedom ν control tail heaviness. During optimization, the new HT-Policy likelihood objective downweights gradients from samples with large residuals.

HT-Policies are simple to implement: we augment a standard regression-based action prediction model with an input-dependent residual scale predictor and jointly train both using the following heteroscedastic Student-t (HT) loss:

Heteroscedastic Student-t loss. Given a data sample (o, a), let $f_\theta(o) \in \mathbb{R}^d$ be the predicted action chunk and $r = f_\theta(o) - a$ the residual. We assume that r follows a multivariate Student-t distribution centered at zero (Lange et al., 1989), with fixed degrees of freedom ν and scale matrix $\sigma_\theta^2(o)$. The model predicts the action $f_\theta(o)$ and the residual scale $\sigma_\theta(o)$, trained by maximizing the likelihood:

$$\ell_{\mathrm{HT}}(o, a) = \frac{d}{2} \log \sigma_\theta^2(o) + \frac{\nu + d}{2} \log\left(1 + \frac{\|r\|_2^2}{\nu \sigma_\theta^2(o)}\right), \tag{1}$$

Note that $\sigma_\theta(o)$ is the Student-t scale, rather than the conditional standard deviation in MSE.

To understand how the predicted scale affects each sample's contribution to optimization (Seitzer et al., 2022b), we can examine the gradient of the HT loss with respect to the predicted action:

$$\nabla_{f_\theta(o)} \ell_{\mathrm{HT}} = \frac{r}{\sigma_\theta^2(o)}\, w\left(\frac{\|r\|_2^2}{\sigma_\theta^2(o)}\right), \qquad \text{where } w(s) = \frac{\nu + d}{\nu + s}. \tag{2}$$

This gradient depends on both the predicted scale and the magnitude of the prediction error, relative to that scale. A larger predicted scale reduces the gradient through the factor $1/\sigma_\theta^2(o)$. For a fixed scale, the weight $w(s)$ decreases as the residual grows, limiting the influence of large prediction errors. These gradients then propagate through the action prediction layer to the shared backbone.

Selecting the degrees of freedom. The parameter ν controls how quickly the weight decreases as the residual grows. We set $\nu = cd$, where d is the action dimension, and rewrite the weight as

$$w(s) = \frac{\nu + d}{\nu + s} = \frac{c + 1}{c + s/d}, \qquad s = \frac{\|r\|_2^2}{\sigma_\theta^2(o)}. \tag{3}$$

Here, s/d measures the average squared error relative to the predicted scale across all action dimensions. This reparameterization $\nu = cd$ lets us choose a hyperparameter c independent of the number of action dimensions. A smaller c therefore makes the weight drop more quickly as the error grows. As $c \to \infty$, $w(s) \to 1$, and the HT objective reduces to a heteroscedastic Gaussian objective, retaining scale-based weighting but no longer downweighting samples with large residuals. We study how to choose c empirically in Section 5.5.

Fine-tuning pretrained Flow-Policies. We can adapt a pretrained flow network into a direct action predictor by setting its action input to zero and its timestep to the pure-noise endpoint. At this input, the optimal flow predictor outputs the conditional mean action, motivating the readout $f_\theta(o) =$

(a) RoboMimic

Backbone	DP	MSE	HT
U-Net	93.8/84.7	81.7/72.3	**95.3/85.6**
Transformer	**94.3/84.1**	82.6/74.6	91.9/82.9

(b) RoboCasa-GR1

Flow	MSE	HT
14.0	5.8	**33.5**

Table 1: Success rates (%) for policies trained from scratch. (a) **RoboMimic.** Best / mean over the last ten evaluated checkpoints (Chi et al., 2023), averaged over nine tasks and three seeds per task, with state and image results weighted equally. (b) **RoboCasa-GR1.** A randomly initialized GR00T N1.7 action head trained with the official fine-tuning recipe.

$v_\theta(0, 0, o)$. We can then add a scale prediction head and fine-tune the model with the HT loss in Eq. (1), allowing it to predict an action chunk with a single network evaluation.

To see why this readout is natural, consider the flow path $x_\tau = (1 - \tau)\epsilon + \tau a$, where $\epsilon \sim \mathcal{N}(0, I_d)$ is sampled independently of the action and policy inputs. The network learns to predict the velocity $a - \epsilon$ (Lipman et al., 2023). At the pure-noise endpoint $\tau = 0$, we have $x_0 = \epsilon$, so the optimal predictor under squared error satisfies $v^*(x, 0, o) = \mathbb{E}[a \mid o] - x$.

For models using the reversed path $x_t = (1 - t)a + t\epsilon$ with velocity target $\epsilon - a$, the corresponding readout is $f_\theta(o) = -v_\theta(0, 1, o)$. Except for changing the prediction target and adding a scale prediction head, we retain each model's original fine-tuning settings in our experiments.

5 Experiments and Analysis

In Section 3, we have illustrated how a standard MSE-based–regression policy (*MSE-Policy*) fails to model state-dependent residuals and long tails in action learning. This failure leads to imbalanced gradients across state-action pairs with different estimated residuals, which in turn hurts model performance. In the following empirical evaluation, we focus on three questions. First, can the proposed heteroscedastic Student-t policy (*HT-Policy*) address state-dependent residuals and long tails, as evaluated by its performance on robot manipulation tasks in both simulation and the real world (Section 5.1)? Second, can we adapt pre-trained vision-language-action (VLA) and world-action (WAM) models to HT-Policy to fully leverage existing community efforts in generalized policy learning? Third, do the data likelihood (residual modeling) and gradient distributions of HT-Policies trained on real-world data support the observations and principles identified in Section 3?

Beyond their performance and general applicability, HT-Policies are a useful policy class for practitioners because they offer greater training and inference efficiency than generative policies, such as flow-matching–based policies (*Flow-Policies*), as shown in Section 5.4. We conclude this section with ablation studies of different robust-regression objectives and an analysis of the selection of ν.

Furthermore, HT-Policies offer closed-form likelihood evaluation and efficient action sampling, making them well-suited for reinforcement-learning–based fine-tuning. We evaluate their exploration capability on an example task, Tool-Hang, in Appendix A.3. Appendix A.4 provides additional training and evaluation details, and Appendix A.7 provides additional ablation results.

5.1 Closing the Performance Gap When Training from Scratch

We first investigate whether HT-Policy's modeling of state-dependent and long-tail residuals improves robot manipulation performance when training policies from scratch. We compare HT-Policies with MSE-Policies, diffusion–based policies, and Flow-Policies.

Simulation experiment setup. On RoboMimic (Mandlekar et al., 2021), we train U-Net and Transformer policies in both state-based and image-based settings following Diffusion Policy (DP) (Chi et al., 2023). On RoboCasa-GR1 (NVIDIA et al., 2025), we train action decoders from scratch but using image features encoded by a frozen VLM backbone from GR00T N1.7 for the vision input. Appendix A.4.5 provides more model and training details.

Table 1 summarizes the results. On RoboMimic, HT-Policies are on par with generative policies and substantially outperform MSE-based policies. On RoboCasa-GR1, HT-Policies outperform both MSE-Policies and Flow-Policies. Overall, HT-Policies are strong direct-regression algorithms for policy training from scratch. We provide per-task performance breakdowns in Appendix A.5.1 and additional comparisons with MIP policies (Pan et al., 2026) on RoboMimic in Appendix A.5.2.

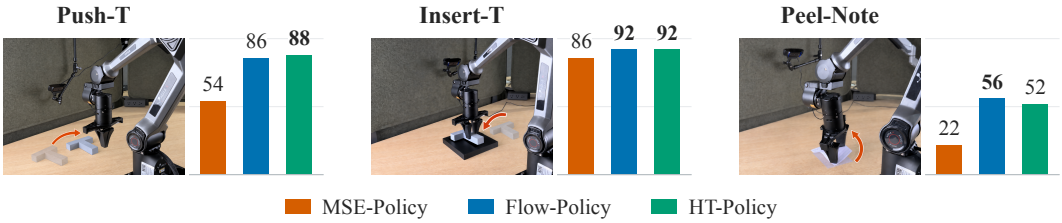

Figure 7: **Real-robot success rates (%).** We evaluate on 3 tasks: Push-T, Insert-T, and Peel-Note.

Model	Benchmark	Flow (Official)	Our Evaluations (10 seeds)		
			Flow	MSE	HT
$\pi_{0.5}$	LIBERO (4-suite avg.)	96.9	$\mathbf{97.09 \pm 0.04}$	96.72 ± 0.09	96.97 ± 0.07
GR00T N1.7	RoboCasa-GR1	44.5	42.34 ± 0.68	35.55 ± 0.67	$\mathbf{50.01 \pm 0.51}$
	SIMPLER / Bridge	62.3	62.06 ± 0.40	58.51 ± 0.37	$\mathbf{62.31 \pm 0.65}$
	SIMPLER / Fractal	72.5	$\mathbf{67.83 \pm 0.39}$	60.60 ± 0.43	65.25 ± 0.44
Cosmos 3	LIBERO-10	95.2	95.60 ± 0.90	95.10 ± 0.26	$\mathbf{96.50 \pm 0.45}$

Table 2: **Success rate (%) for fine-tuning from pretrained policies.** We report the mean success rate and its standard error of a single checkpoint across 10 evaluation seeds. We include official results for the flow-matching–based fine-tuning for reference. For $\pi_{0.5}$, we report the average performance in four LIBERO suites. Bold marks highlight the best-performing model in each row.

Real-robot experiment setup. Furthermore, we evaluate different policies on three real-world manipulation tasks: Push-T, Peel-Note, and Insert-T (Figure 7). Training and evaluation details are provided in Appendix A.4.1. On the real robot, HT-Policies largely close the performance gap between MSE-based and generative policies on all three tasks. The largest improvements occur on Push-T and Peel-Note, where MSE-Policies fail significantly.

Overall, in both simulation and real-robot tasks, HT-Policies perform on par with generative policies when the action model is trained from scratch. These gains are achieved with a simple feed-forward-only policy, highlighting the importance and effectiveness of residual modeling.

5.2 Fine-Tuning Pretrained Flow-Policies with HT Regression

We next investigate whether HT-Policies can be fine-tuned from pretrained VLA and WAM models to leverage existing advances in generalized policy learning. Following the procedure described in Section 4, the only architectural change is the addition of a randomly initialized residual scale prediction head $\sigma_\theta(o)$. We fine-tune the action predictor using the HT loss while retaining each model's original fine-tuning settings and hyperparameters. We show results on three models: $\pi_{0.5}$ (Intelligence et al., 2025), GR00T N1.7 (NVIDIA et al., 2025), and Cosmos 3 (NVIDIA et al., 2026), and on three benchmarks: LIBERO (Liu et al., 2023), SIMPLER (Li et al., 2024; Walke et al., 2023; Brohan et al., 2022), and RoboCasa-GR1 (NVIDIA et al., 2025). We evaluate each method across 10 evaluation seeds. These settings span VLA and WAM architectures and cover both fixed-arm and humanoid manipulation. We report the implementation details in Appendix A.4.

Shown in Table 2, HT-Policies achieve higher success rates than MSE-Policies in all five settings, and outperform Flow-Policies in three of them. These results are obtained using fine-tuning recipes adopted directly from Flow-Policies, rather than specifically tuned for HT-Policies. The only tuned hyperparameter is the learning rate of the newly added scale prediction head, which is trained from scratch. We provide per-task performance break-downs in Appendices A.5.3, A.5.4, and A.5.5.

5.3 Residual Modeling, Gradient Allocation, and Policy Performance

The results above show that changing the residual model makes regression-based policies competitive with generative policies. To further close the theory-implementation gap, we examine the data likelihood and gradient distributions of policies trained on real-world data. This analysis tests whether the principles identified in Section 3 regarding residual modeling and gradient allocation explain the observed performance gains.

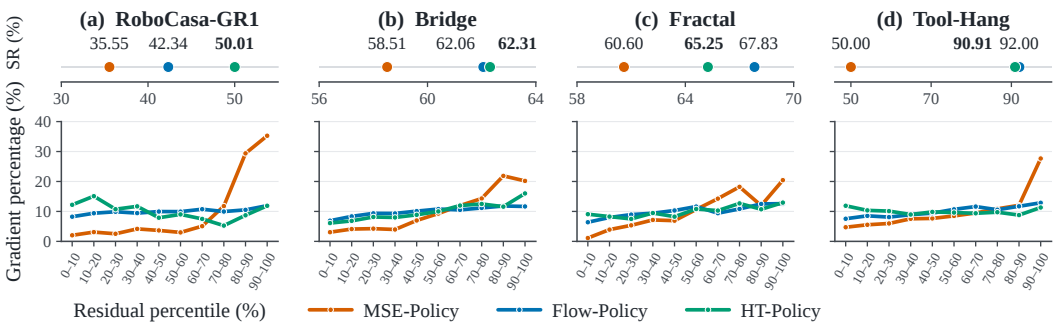

Figure 8: **Residual modeling and optimization. Top:** Policy success rates across four settings. **Bottom:** Each action-residual decile's share of summed per-sample parameter-gradient RMS under MSE, HT, and low-noise Flow ($t > 0.5$). Each decile contains approximately 10% of the samples.

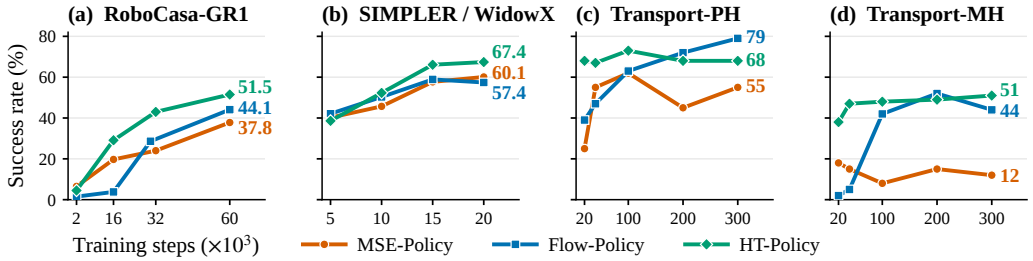

Figure 10: **Training curves for different models on different benchmarks.** HT-Policies reach strong task success rates with fewer training steps than Flow-Policies.

Gradient allocation across observations Figure 8 shows that the HT loss produces a more balanced allocation of gradients across observations: large-residual observations no longer dominate the gradients as they do under MSE. This allocation resembles that of low-noise flow-matching when $t > 0.5$. Appendix A.1.4 extends this comparison to different training stages.

Residual modeling and policy performance. Figure 9 examines whether data likelihood under different residual models is correlated with policy performance on RoboCasa-GR1. Beyond MSE and HT, we include six alternative models, such as models trained with the Huber loss. Across all eight objectives, calibrated mean log-likelihood is strongly correlated with policy success rate (Spearman $\rho = 0.86$; Pearson $r = 0.83$). HT-Policies achieve substantially higher likelihood than the conventional Gaussian residual model from MSE-based policies, and the highest success rate. Modeling either heavy tails with a Student-t distribution or state-dependent scales with a heteroscedastic Gaussian (HG) also improves likelihood and success over MSE, but neither matches HT, which incorporates both. Appendix A.7.2 provides the evaluation details. Together, these findings support the principles that improved residual modeling and more balanced gradient allocation yield better policy performance.

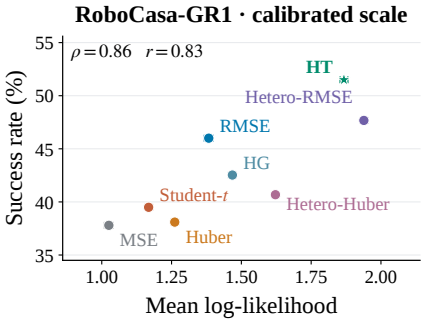

Figure 9: **Mean log-likelihood vs. policy success rate.**

5.4 Training and Inference Efficiency

A core advantage of regression policies for practitioners is their training and inference efficiency. We next evaluate whether HT-Policies retain this advantage while achieving strong task performance.

Training efficiency. Shown in Figure 10, we compare the training progress of HT-Policies, MSE-Policies, and Flow-Policies in two training-from-scratch and two fine-tuning settings. In all four settings, HT-Policies reach strong task success with fewer training steps than Flow-Policies.

Inference efficiency. HT-Policies require only a single feed-forward pass at inference. Table 3 reports batch-size-one inference runtime on an NVIDIA RTX 5090 GPU. HT achieves higher end-

Metric	GR00T N1.7			$\pi_{0.5}$			Cosmos3-Nano		
	Flow	HT	Speed-up	Flow	HT	Speed-up	Flow	HT	Speed-up
Head latency ↓	29.30	**7.92**	3.70×	62.50	**6.42**	9.74×	1266.10	**65.96**	19.19×
Total latency ↓	46.20	**24.52**	1.88×	123.58	**67.94**	1.82×	1271.76	**71.34**	17.83×
FPS ↑	21.65	**40.78**	1.88×	8.09	**14.72**	1.82×	0.79	**14.02**	17.83×

Table 3: **Inference efficiency benchmarking.** We evaluate all policies on a single NVIDIA RTX 5090 GPU with batch size 1. Latencies are in milliseconds.

to-end throughput than flow on all three models. Cosmos3-Nano shows the largest whole-model speedup of 17.83×, with our HT-Policy replacing the 30-step sampling procedure of Flow-Policies. Appendix A.6 provides additional implementation details and results on NVIDIA A800 GPUs.

5.5 Ablation Studies

Ablation on the degrees of freedom ν. In practice, we select ν in proportion to the action-chunk dimension d, parameterizing it as $\nu = cd$. Figure 11a shows that RoboCasa-GR1 performance peaks at an intermediate value of ν, favoring moderate downweighting of large residuals over either strong downweighting or the Gaussian limit. To evaluate this dimension-dependent choice across settings, Figure 11b compares performance as a function of $c = \nu/d$. RoboCasa-GR1, Bridge, and LIBERO all achieve their highest tested success rates at $c = 4$, supporting $\nu = 4d$ as a practical default across these settings. Appendix A.7.3 provides evaluation details.

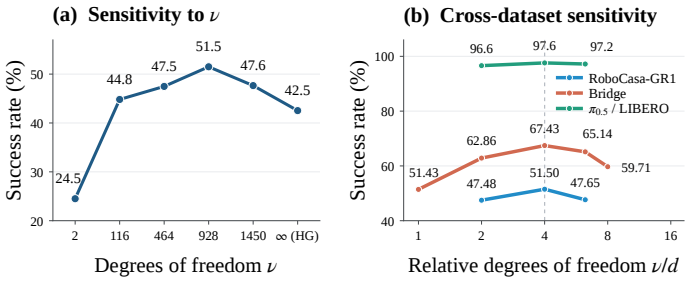

Figure 11: **Selection of ν. (a)** Success rate vs. ν on RoboCasa-GR1 and **(b)** Success rate vs. $c = \nu/d$ on three datasets, where d is the action-chunk dimension. $\nu \to \infty$ gives heteroscedastic Gaussian.

Comparison with alternative training objectives. Figure 9 summarizes the comparison among HT-Policies and alternative robust-regression objectives by calibrated likelihood and performance; Appendix A.7.1 defines these objectives (Table 28) and provides detailed results (Table 27). Overall, HT-Policies achieve the best performance.

6 Conclusion

In this paper, we revisit the performance gap between direct action-regression policies and generative policies through the lens of statistical modeling of prediction residuals and their effects on optimization. Contrary to the common hypothesis that generative policies perform better because they better model multimodal action distributions, our analysis suggests that the empirical performance gap is better explained by differences in how these methods handle network prediction residuals. We show that real-world demonstrations exhibit input-dependent residual scales and heavy tails. The fixed-scale residual model implicit in MSE regression leads to gradient imbalance, whereas the denoising objectives in generative policies better mitigate this. These findings motivate HT-Policies, a class of neural policies that are performant without iterative computation, can be trained from scratch or fine-tuned from pretrained models, and offer improved training and inference efficiency.

Limitations. So far, our analysis has focused on network fitting and prediction errors, without directly connecting them to control failures. In particular, it does not explain why small errors at a few states can catastrophically reduce overall success rate. Incorporating control-theoretic analysis is an important next step, such as in Tu et al. (2022). Second, HT-Policies can not explicitly represent multiple action modes. Our contribution offers an explanation for the observed gap between regression and generative policies and an efficient alternative for tasks with limited multimodality, rather than a universal solution to policy learning. Finally, although the appendix provides preliminary evidence that HT-Policies offer a promising starting point for RL due to their closed-form likelihood and action sampling, a systematic evaluation remains future work.

7 AI USE STATEMENT

In this work, we used generative AI tools for helping develop theoretical models or conceptual frameworks, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, and interpreting results. We have not used generative AI tools for proposing or refining scientific hypotheses. Generating synthetic datasets and formulating mathematical claims are not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures or images, suggest experimental parameters, create or edit software code, transcribe recordings of research material, search for information, summarize or analyze existing literature, identify relevant literature, suggest the structure of the manuscript, suggest a title or keywords, and draft and edit portions of the manuscript for clarity and readability. We have not used generative AI tools to formulate survey or interview questions, create research artifacts, discover research topics or identify research gaps, brainstorm, or format references.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

D. F. Andrews and C. L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(1):99–102, 1974. doi: 10.1111/j.2517-6161.1974.tb00989.x. URL https://doi.org/10.1111/j.2517-6161.1974.tb00989.x.

Jonathan T. Barron. A general and adaptive robust loss function, 2019. URL https://arxiv.org/abs/1701.03077.

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

Shang-Fu Chen, Co Yong, and Shao-Hua Sun. Restoring noisy demonstration for imitation learning with diffusion models. *IEEE Transactions on Neural Networks and Learning Systems*, 37:401–413, 2025. URL https://api.semanticscholar.org/CorpusID:281386356.

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

Kanishk Gandhi, Siddharth Karamcheti, Madeline Liao, and Dorsa Sadigh. Eliciting compatible demonstrations for multi-human imitation learning. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1981–1991. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/gandhi23a.html.

Ross Girshick. Fast r-cnn, 2015. URL https://arxiv.org/abs/1504.08083.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35:73–101, 1964. URL https://doi.org/10.1214%2Faoms%2F1177703732.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh,

Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6pPYRXKPpw.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision?, 2017. URL https://arxiv.org/abs/1703.04977.

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

Kenneth L. Lange, Roderick J. A. Little, and Jeremy M. G. Taylor. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association*, 84(408):881–896, 1989. ISSN 01621459, 1537274X. URL http://www.jstor.org/stable/2290063.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

D.A. Nix and A.S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pp. 55–60 vol.1, 1994. doi: 10.1109/ICNN.1994.374138.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

NVIDIA, :, Aditi, Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, Aarti Basant, Mukesh Beladiya, Mohammad Qazim Bhat, Zaid Pervaiz Bhat, Dan Blick, Vanni Brighella, Han Cai, Tiffany Cai, Eric Cameracci, Jiaxin Cao, Yulong Cao, Mark Carlson, Carlos Casanova, Ting-Yun Chang, Yan Chang, Yu-Wei Chao, Prithvijit Chattopadhyay, Roshan Chaudhari, Chieh-Yun Chen, Junyu Chen, Ke Chen, Qizhi Chen, Wenkai Chen, Xiaotong Chen, Yu Chen, An-Chieh Cheng, Click Cheng, Xiu Chia, Jeana Choi, Chaeyeon Chung, Wenyan Cong, Yin Cui, Magdalena Dadela, Nalin Dadhich, Wenliang Dai, Joyjit Daw, Alperen Degirmenci, Rodrigo Vieira Del Monte, Robert Denomme, Sameer Dharur, Marco Di Lucca, Ke Ding, Wenhao Ding, Yifan Ding, Yuzhu Dong, Nicole Drumheller, Yilun Du, Aigul Dzhumamuratova, Aleksandr Efitorov, Hamid Eghbalzadeh, Naomi Eigbe, Imad El Hanafi, Hassan Eslami, Benedikt Falk, Jiaojiao Fan, Jim Fan,

Amol Fasale, Sergiy Fefilatyev, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Vikram Fugro, Prashant Gaikwad, TJ Galda, Katelyn Gao, Yihuai Gao, Wenhang Ge, Sreyan Ghosh, Arushi Goel, Vivek Goel, Akash Gokul, Rama Govindaraju, Jinwei Gu, Miguel Guerrero, Elfie Guo, Aryaman Gupta, Siddharth Gururani, Hugo Hadfield, Song Han, Ankur Handa, Zekun Hao, Mohammad Harrim, Ali Hassani, Nathan Hayes-Roth, Yufan He, Chris Helvig, Cyrus Hogg, Madison Huang, Michael Huang, Sophia Huang, Yufan Huang, Jacob Huffman, DeLesley Hutchins, Suneel Indupuru, Boris Ivanovic, Arihant Jain, Joel Jang, Ryan Ji, Yanan Jian, Dongfu Jiang, Jingyi Jin, Atharva Joshi, Nikhilesh Joshi, Pranjali Joshi, Andy Ju, Jaehun Jung, Weiwei Kang, Scott Kassekert, Jan Kautz, Ashna Khetan, Julia Kiczka, Slawek Kierat, Gwanghyun Kim, Kuno Kim, Sunny Kim, Kezhi Kong, Xin Kong, Zhifeng Kong, Tomasz Kornuta, Egor Krivov, Hui Kuang, Saurav Kumar, Chia-Wen Kuo, George Kurian, Wojciech Kutak, JF Lafleche, Himangshu Lahkar, Omar Laymoun, Jayjun Lee, Sanggil Lee, Gabriele Leone, Boyi Li, Freya Li, Jiajun Li, Jinfeng Li, Ling Li, Pengcheng Li, Shangru Li, Tingle Li, Xiaolong Li, Xuan Li, Zhaoshuo Li, Zhiqi Li, Hao Liang, Maosheng Liao, Chen-Hsuan Lin, Tsung-Yi Lin, Ming-Yu Liu, Sifei Liu, Zihan Liu, Hai Loc Lu, Xiangyu Lu, Alice Luo, Ruipu Luo, Wenjie Luo, Jiangran Lyu, Martin Ding Ma, Nic Ma, Qianli Ma, Dawid Majchrowski, Louis Marcoux, Miguel Martin, Qing Miao, Ashkan Mirzaei, Shreyas Misra, Kaichun Mo, Durra Mohsin, Hyejin Moon, Pawel Morkisz, Saeid Motiian, Kirill Motkov, Seungjun Nah, Yashraj Narang, Deepak Narayanan, Thabang Ngazimbi, Julian Ouyang, Shubham Pachori, David Page, Yatian Pang, Sehwi Park, Mahesh Patekar, Mostofa Patwary, Marco Pavone, Trung Pham, Wei Ping, Soha Pouya, Shrimai Prabhumoye, Varun Praveen, Delin Qu, Hesam Rabeti, Morteza Ramezanali, Marilyn Reeb, Xuanchi Ren, Kristen Rumley, Wojciech Rymer, Jun Saito, Yeongho Seol, John Shao, Piyush Shekdar, Tianwei Shen, Humphrey Shi, Min Shi, Stella Shi, Kevin Shih, Mohammad Shoeybi, Mateusz Sieniawski, Shuran Song, Alexander Sotelo, Amir Sotoodeh, Sunil Srinivasa, Vignesh Srinivasakumar, Bartosz Stefaniak, Rahul Heinrich Steiger, Shangkun Sun, Jiaxiang Tang, Shitao Tang, Yangyang Tang, Yue Tang, Tolou Tavakkoli, Kayley Ting, Krzysztof Tomala, Wei-Cheng Tseng, Jibin Varghese, Sergei Vasilev, Thomas Volk, Raju Wagwani, Roger Waleffe, Andrew Z. Wang, Boxiang Wang, Haoxiang Wang, Qiao Wang, Shihao Wang, Shijie Wang, Ting-Chun Wang, Yan Wang, Yu Wang, Rohit Watve, David Wehr, Fangyin Wei, Xinshuo Weng, Jay Zhangjie Wu, Kedi Wu, Hongchi Xia, Summer Xiao, Tianjun Xiao, Kevin Xie, Daguang Xu, Jiashu Xu, Mengyao Xu, Ruqing Xu, Xingqian Xu, Yao Xu, Dinghao Yang, Dong Yang, Hans Yang, Xiaodong Yang, Xuning Yang, Yichu Yang, Yurong You, Zhiding Yu, Hao Yuan, Simon Yuen, Xiaohui Zeng, Pengcuo Zeren, Cindy Zha, Haotian Zhang, Jenny Zhang, Jing Zhang, Liangkai Zhang, Paris Zhang, Shun Zhang, Xuanmeng Zhang, Zhizheng Zhang, Ann Zhao, Yilin Zhao, Yuliya Zhautouskaya, Charles Zhou, Fengzhe Zhou, Shilin Zhu, Yuke Zhu, Dima Zhylko, and Artur Zolkowski. Cosmos 3: Omnimodal world models for physical ai, 2026. URL https://arxiv.org/abs/2606.02800.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Matthew Boffi, Guanya Shi, and Max Simchowitz. Much ado about noising: Dispelling the myths of generative robotic control. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LzWKuxTKuW.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=aPOpXlnV1T.

Maximilian Seitzer, Arash Tavakoli, Dimitrije Antic, and Georg Martius. On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks, 2022b. URL https://arxiv.org/abs/2203.09168.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone, 2022. URL https://arxiv.org/abs/2206.11251.

Modi Shi, Li Chen, Jin Chen, Yuxiang Lu, Chiming Liu, Guanghui Ren, Ping Luo, Di Huang, Maoqing Yao, and Hongyang Li. Is diversity all you need for scalable robotic manipulation? *IEEE Transactions on Robotics*, 42:1872–1883, 2026. doi: 10.1109/TRO.2026.3686184.

Voot Tangkaratt, Nontawat Charoenphakdee, and Masashi Sugiyama. Robust imitation learning from noisy demonstrations, 2021. URL https://arxiv.org/abs/2010.10181.

Stephen Tu, Alexander Robey, Tingnan Zhang, and Nikolai Matni. On the sample complexity of stability constrained imitation learning. In *Proceedings of the 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pp. 180–191. PMLR, 2022. URL https://proceedings.mlr.press/v168/tu22a.html.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

A Appendix

A.1 Residual and Gradient Diagnostics

A.1.1 Residual Definitions and Sampling

We analyze residuals and gradients on the training demonstrations, as in Section 3. The calibration splits below hold out observations only for scale estimation.

Policies and samples. Figures 3 and 6 use GR00T MSE-Policies, HG-Policies, and Flow-Policies trained for 60k updates on RoboCasa-GR1 and 20k updates on Bridge and Fractal. We select ten episodes from each of the 24 GR1 tasks. For Bridge and Fractal, we select the 40 most frequent instruction groups and take ten episodes per group. Each episode contributes four evenly spaced, complete eight-step action chunks. Every method uses the same selected observations and demonstrated actions. These groups describe the training demonstrations and are separate from the seven and six SIMPLER evaluation tasks. For Tool-Hang PH, we evaluate state-based MSE-Policies, HG-Policies, and Flow-Policies with Chi-UNet. All three methods share eight evenly spaced, complete 16-step chunks from each of the 200 demonstrations.

Table 4: **Residual-tail samples.** Bridge, Fractal, and Tool-Hang exclude the gripper channel from the plotted residuals. Each Flow-Policy contributes 64 independently evaluated action draws per observation.

Dataset	Episodes	Observations	d	MSE/HG coordinates
RoboCasa-GR1	240	960	232	222,720
Bridge	400	1,600	48	76,800
Fractal	400	1,600	48	76,800
Tool-Hang	200	1,600	144	230,400

Residual definition. We use the action residual $r = \hat{a}(o) - a$ defined in the main text, computed in normalized action space without augmentation or dropout. The chunk magnitude $\mathrm{RMS}(r) = \|r\|_2/\sqrt{d}$ is used for observation-level analyses and gradient grouping. The tail plots pool scalar residual coordinates, each normalized by its coordinate RMS as specified below. GR1 retains its 29 valid action channels. Bridge and Fractal retain six continuous channels and omit the gripper from the tail plots; the HG-Policy's predicted scale still uses its original training mask, including the gripper. Tool-Hang retains nine continuous channels with 6D rotation encoding. For MSE-Policies and HG-Policies, residuals subtract the demonstration from the predicted mean. For Flow-Policies, residuals subtract it from each sampled action, without averaging the generated actions first.

Fixed coordinate scales. For GR00T, the ten episodes within each task or instruction group are divided into five folds of two episodes. Tool-Hang uses five folds of 40 episodes each. For each evaluated fold, the other four folds estimate a fixed RMS for each chunk coordinate j:

$$q_j = \sqrt{\mathbb{E}_{\mathrm{cal}}[u_j^2]}, \qquad z_{ij} = u_{ij}/q_j. \tag{4}$$

Here u_{ij} is either the raw residual or the residual after division by the observation-dependent scale described below. For Flow-Policies, expectations also average over the designated action draws. There is no mean subtraction, residual trimming, or subsequent rescaling using the evaluated fold. All valid scalar coordinates are pooled to estimate $\Pr(|z| > x)$. The percentages in Figure 3 report $\Pr(|z| > 3)$: the fraction of scalar residuals exceeding three times their coordinate RMS in absolute value. The two-sided standard Gaussian reference is approximately 0.27% at this threshold.

Observation-dependent scales. For each HG-Policy, $u_{ij} = r_{ij}/\sigma_i$ uses its predicted scalar scale. For Flow-Policies, we draw 128 independent samples per observation with unit Gaussian initial noise, using the native four-step sampler for GR00T and nine Euler steps for Tool-Hang. The first 64 estimate spread and the remaining 64 enter the tail curves. Let v_{ij} be the unbiased sample variance of coordinate j in the first bank. The calibration folds define $b_j = \mathbb{E}_{\mathrm{cal}}[v_{ij}]$, and the local spread is

$$s_i = \sqrt{\frac{1}{d}\sum_j \frac{v_{ij}}{b_j}}. \tag{5}$$

The own-spread curve uses $u_{ij} = (A_{ij} - a_{ij})/s_i$, followed by Eq. (4). The separate sampling banks keep an extreme evaluated draw from increasing its own normalization scale.

A.1.2 Per-Observation Parameter Gradients

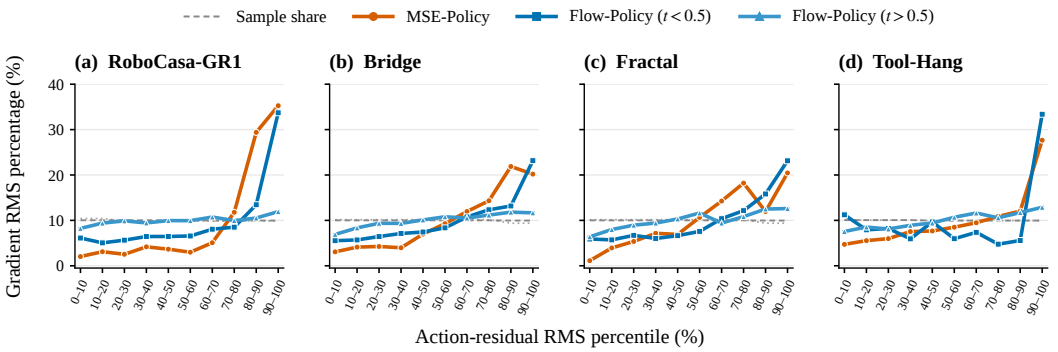

Figure 12: **Gradient imbalance across objectives and noise levels.** Curves show the relative gradient contribution (%) of each action-residual decile. For flow matching, gradient RMS is averaged over noise draws and timesteps separately for high noise ($t < 0.5$) and low noise ($t > 0.5$).

Parameter scope and sample counts. Figures 4 and 8 use gradients of individual observation losses before the optimizer, clipping, or weight decay. For GR00T, the parameter set contains the trainable action-head internals, excluding the final action and scale decoders and the frozen VLM. For Tool-Hang, it contains the native observation encoder and shared Chi-UNet trunk, excluding the output heads. Gradients of the HT loss combine the mean and scale paths before the norm is computed, retaining their cross term. Table 5 gives the number of observations evaluated for each method.

Table 5: **Observation counts for the parameter-gradient diagnostics.** GR00T is trained for 60k updates on GR1 and 20k updates on Bridge and Fractal. Tool-Hang uses state observations and Chi-UNet.

Setting	MSE	Flow	HT
RoboCasa-GR1	1,536	1,536	1,536
Bridge	2,048	2,048	2,048
Fractal	2,048	2,048	2,048
Tool-Hang	4,096	4,096	4,096

Residual ranking and gradient shares. Each policy's observations are ranked by its own action-chunk residual RMS and divided into ten groups of nearly equal size. All valid action channels, including grippers, enter this ranking. MSE-Policies and HT-Policies use their predicted means. Flow-Policies use the complete native sampler starting from zero initial noise. For raw parameter gradient g_{ikp} of observation i, draw k, and parameter p, the reported contribution is

$$m_i = \frac{1}{K}\sum_{k=1}^{K}\sqrt{\frac{1}{P}\sum_{p=1}^{P} g_{ikp}^2}, \qquad G_b = \frac{\sum_{i\in b} m_i}{\sum_i m_i}. \tag{6}$$

The figures display $100G_b$. Norms are computed before averaging draws, and observations are summed only afterwards. This is not the norm of an accumulated batch gradient.

Flow-matching noise and timestep groups. Flow-matching gradients use the native velocity MSE at 16 equally weighted midpoint quantiles of the training timestep distribution, $t_k = Q((k - \frac{1}{2})/16)$ for $k = 1, \ldots, 16$, with independent Gaussian action noise for each observation and draw. Each point represents a quantile interval with probability $1/16$. The high- and low-noise curves retain points with $t_k < 0.5$ and $t_k > 0.5$, respectively, before applying Eq. (6). Table 6 summarizes the resulting groups. The thresholds apply to the representative points, so the represented intervals need not meet at exactly 0.5. These two flow-matching curves share the same observations and residual bins. Gradients are measured with training-mode dropout and augmentation.

Table 6: **Flow-matching timestep groups for gradient diagnostics.** Ranges span the full quantile intervals represented by each group; percentages denote their training probability mass.

Dataset	Quantile function $Q(u)$	High noise ($t < 0.5$)	Low noise ($t > 0.5$)
RoboCasa-GR1, Bridge, Fractal	$0.999[1-(1-u)^{2/3}]$	Points 1–10 (62.5%) $[0,\ 0.4795)$	Points 11–16 (37.5%) $[0.4795,\ 0.999]$
Tool-Hang	u	Points 1–8 (50%) $[0,\ 0.5)$	Points 9–16 (50%) $[0.5,\ 1]$

A.1.3 Gaussian Tail Bounds for Flow-Matching Velocity Residuals

We examine the tail-bound hypothesis in Section 3 using the velocity residual $r_\theta = v_\theta(a_t, t, o) - (a - \epsilon)$. At six fixed low-noise timesteps ($t > 0.5$), we compare the empirical tail probabilities of $t\|r_\theta\|_2/\sqrt{d}$ with the Gaussian norm reference $\|\epsilon\|_2/\sqrt{d} \sim \chi_d/\sqrt{d}$. The largest empirical tail probability across these timesteps remains below the reference over the plotted range across datasets (Figure 13) and training stages on RoboCasa-GR1 (Figure 14), supporting the hypothesis in the evaluated low-noise settings.

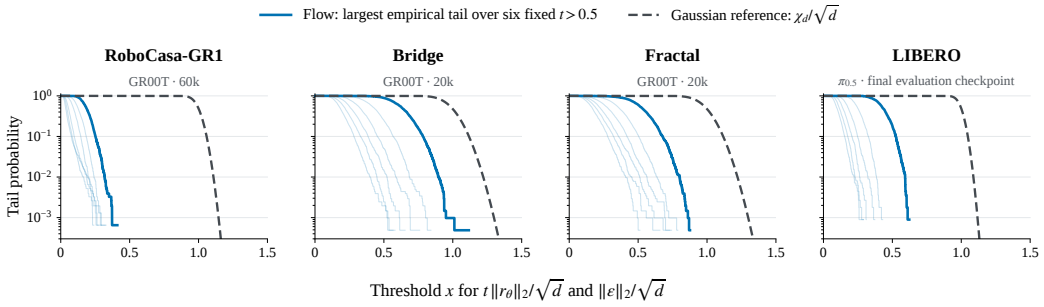

Figure 13: **Gaussian tail-bound comparison across datasets.** GR00T uses 60k updates on RoboCasa-GR1 and 20k on Bridge and Fractal; LIBERO results use $\pi_{0.5}$. Light curves show individual timesteps, the thick blue curve their pointwise maximum, and the dashed curve the Gaussian norm reference.

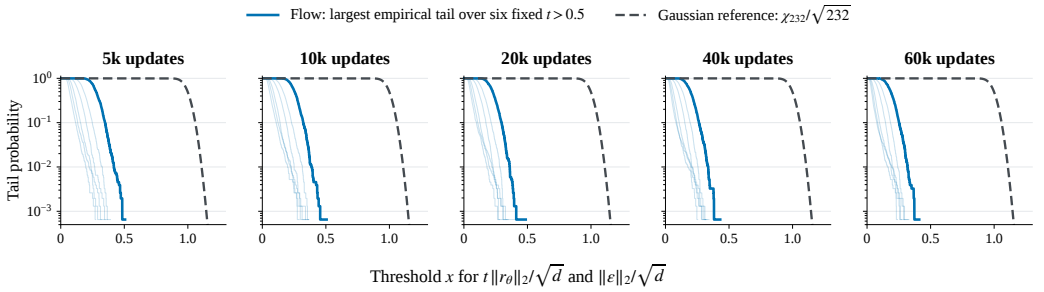

Figure 14: **Gaussian tail-bound comparison across training stages.** GR00T on RoboCasa-GR1 after 5k, 10k, 20k, 40k, and 60k updates, with $d = 232$. Curves follow the same convention as Figure 13.

A.1.4 Gradient Allocation across Training Stages

We examine whether the gradient patterns in Section 5.3 are already present early in training. Figure 15 compares MSE-Policies, Flow-Policies, and HT-Policies on RoboCasa-GR1 after 5k, 10k, 20k, 40k, and 60k optimizer updates. At each stage, it shows how the summed per-sample gradient RMS is distributed across action-residual deciles, together with the corresponding policy success rate. We follow the gradient-diagnostic protocol in Appendix A.1.2, including low-noise flow-matching timesteps ($t > 0.5$). These stage-wise results are reported separately from the main benchmark comparison.

The MSE loss assigns a disproportionate gradient share to the largest residuals at every evaluated stage, and its highest-residual decile becomes more dominant as training progresses. The flow-

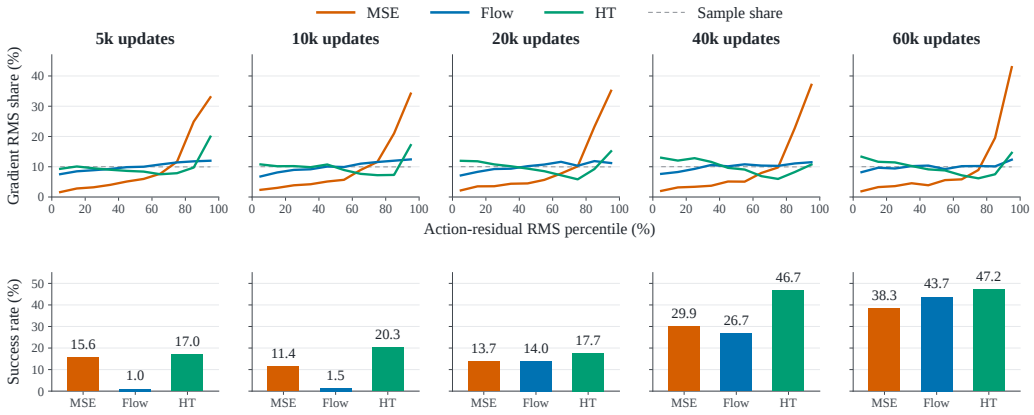

Figure 15: **Gradient allocation and policy success across training stages on RoboCasa-GR1.** Columns show results after 5k, 10k, 20k, 40k, and 60k updates. **Top:** Each action-residual decile's share of summed per-sample parameter-gradient RMS, with flow-matching gradients evaluated at low-noise timesteps ($t > 0.5$). The dashed line marks the 10% sample share of each decile. **Bottom:** Success rates of MSE-Policies, Flow-Policies, and HT-Policies at the corresponding training stages; HT-Policies have the highest success rate at all five stages.

matching and HT losses distribute gradient magnitude more evenly across residual deciles throughout the comparison. In particular, the HT loss already reduces the concentration of gradients in the residual tail at 5k updates, well before HT-Policies reach their strongest performance at 40k and 60k. The reduced tail influence is therefore visible early and persists across the evaluated training stages.

A.2 Action Averaging in Flow-Policies

We examine whether averaging actions sampled from a trained Flow-Policy affects task success. We keep each policy fixed and compare its native single-sample output ($K = 1$) with the mean of $K = 16$ complete action chunks. At each replanning step, we condition all samples on the current observation o and use distinct initial noise:

$$A^{(k)} = \mathrm{Flow}_\theta(o, \epsilon_k), \qquad \bar{A} = \frac{1}{16}\sum_{k=1}^{16} A^{(k)}. \tag{7}$$

Each sample uses the full native sampler: 4 steps for GR00T N1.7, 10 for $\pi_{0.5}$, and 30 for Cosmos3-Nano. We average the final chunks in normalized action space, including gripper and finger channels, before applying the original action postprocessing and execution. Both conditions use the same trained policy, observation preprocessing, and action execution length.

Evaluation. We evaluate all 24 RoboCasa-GR1 tasks, all 40 tasks in the four LIBERO suites for $\pi_{0.5}$, and all 10 LIBERO-10 tasks for Cosmos3-Nano. The requested rollout counts per task and condition are 20, 100, and 10, respectively; each query executes 8, 10, and 16 actions. In particular, $\pi_{0.5}$ executes only the first 10 actions of its 50-step prediction. Cosmos3-Nano retains joint video/action denoising for every sample and averages only the final actions. Its $K = 16$ mean includes the $K = 1$ sample and 15 additional noise initializations.

Each setting uses one evaluation run per condition, with $K = 1$ remeasured for this comparison. These results are separate from the 10-seed means in Table 2. RoboCasa-GR1 has 480 and 483 completed rollouts for $K = 1$ and $K = 16$, respectively. We compute each task's success rate using its actual rollout count and average task rates equally. For $\pi_{0.5}$, we average task rates within each suite and then average the four suites.

Table 7 shows no decrease in benchmark-average success under action averaging in these three evaluations, although individual tasks can improve or decline. These results suggest that averaging sampled action chunks need not harm performance in the tested policies; the single-run comparisons do not establish a consistent improvement across evaluation seeds. Tables 8–10 provide all task-level results.

Table 7: **Action averaging in fixed Flow-Policies.** Success rates (%) from one evaluation run per condition. Δ is the change in percentage points, computed before rounding.

Model / benchmark	$K = 1$	$K = 16$ mean	Δ (pp)
GR00T N1.7 / RoboCasa-GR1	40.63	43.23	$+2.61$
$\pi_{0.5}$ / LIBERO	97.250	97.325	$+0.075$
Cosmos3-Nano / LIBERO-10	96.00	98.00	$+2.00$

Table 8: **Action averaging on RoboCasa-GR1.** Entries give successes / completed rollouts, with success rates (%) in parentheses. Task names follow Table 22. The mean weights all 24 tasks equally; Δ is computed before rounding.

Task	$K = 1$	$K = 16$ mean	Δ (pp)
PnPBottle $\rightarrow$ CabinetClose	8/20 (40.00)	12/20 (60.00)	$+20.00$
PnPCan $\rightarrow$ DrawerClose	2/20 (10.00)	6/20 (30.00)	$+20.00$
PnPCup $\rightarrow$ DrawerClose	0/20 (0.00)	4/20 (20.00)	$+20.00$
PnPMilk $\rightarrow$ MicrowaveClose	3/20 (15.00)	1/20 (5.00)	-10.00
PnPPotato $\rightarrow$ MicrowaveClose	4/20 (20.00)	1/20 (5.00)	-15.00
PnPWine $\rightarrow$ CabinetClose	3/20 (15.00)	3/21 (14.29)	-0.71
Cuttingboard $\rightarrow$ Basket	14/20 (70.00)	15/20 (75.00)	$+5.00$
Cuttingboard $\rightarrow$ Cardboardbox	8/20 (40.00)	6/20 (30.00)	-10.00
Cuttingboard $\rightarrow$ Pan	15/20 (75.00)	14/20 (70.00)	-5.00
Cuttingboard $\rightarrow$ Pot	14/20 (70.00)	13/20 (65.00)	-5.00
Cuttingboard $\rightarrow$ Tieredbasket	6/20 (30.00)	10/20 (50.00)	$+20.00$
Placemat $\rightarrow$ Basket	12/20 (60.00)	14/20 (70.00)	$+10.00$
Placemat $\rightarrow$ Bowl	3/20 (15.00)	7/20 (35.00)	$+20.00$
Placemat $\rightarrow$ Plate	8/20 (40.00)	9/20 (45.00)	$+5.00$
Placemat $\rightarrow$ Tieredshelf	1/20 (5.00)	3/20 (15.00)	$+10.00$
Plate $\rightarrow$ Bowl	8/20 (40.00)	14/20 (70.00)	$+30.00$
Plate $\rightarrow$ Cardboardbox	9/20 (45.00)	4/21 (19.05)	-25.95
Plate $\rightarrow$ Pan	9/20 (45.00)	6/20 (30.00)	-15.00
Plate $\rightarrow$ Plate	16/20 (80.00)	13/20 (65.00)	-15.00
Tray $\rightarrow$ Cardboardbox	10/20 (50.00)	14/20 (70.00)	$+20.00$
Tray $\rightarrow$ Plate	13/20 (65.00)	15/20 (75.00)	$+10.00$
Tray $\rightarrow$ Pot	15/20 (75.00)	13/20 (65.00)	-10.00
Tray $\rightarrow$ Tieredbasket	10/20 (50.00)	8/20 (40.00)	-10.00
Tray $\rightarrow$ Tieredshelf	4/20 (20.00)	3/21 (14.29)	-5.71
Mean success (%)	40.63	43.23	$+2.61$

Table 9: **Action averaging for $\pi_{0.5}$ on all four LIBERO suites.** Entries give successes / completed rollouts. Task IDs are listed in Tables 20 and 21.

	LIBERO-10		LIBERO-Goal		LIBERO-Object		LIBERO-Spatial	
ID	$K = 1$	$K = 16$	$K = 1$	$K = 16$	$K = 1$	$K = 16$	$K = 1$	$K = 16$
0	96/100	93/100	99/100	99/100	100/100	100/100	100/100	100/100
1	98/100	98/100	98/100	100/100	100/100	100/100	100/100	99/100
2	99/100	96/100	89/100	91/100	100/100	100/100	100/100	100/100
3	100/100	100/100	94/100	94/100	97/100	98/100	97/100	100/100
4	99/100	98/100	100/100	100/100	100/100	99/100	98/100	98/100
5	100/100	100/100	99/100	99/100	100/100	100/100	80/100	84/100
6	95/100	95/100	96/100	95/100	100/100	100/100	100/100	99/100
7	99/100	98/100	100/100	100/100	99/100	100/100	99/100	96/100
8	72/100	79/100	100/100	100/100	100/100	100/100	99/100	97/100
9	97/100	95/100	94/100	94/100	99/100	99/100	98/100	100/100
Suite SR (%)	95.50	95.20	96.90	97.20	99.50	99.60	97.10	97.30
Δ (pp)	-0.30		$+0.30$		$+0.10$		$+0.20$	

Table 10: **Action averaging for Cosmos3-Nano on LIBERO-10.** Entries give successes / completed rollouts. Task IDs follow the LIBERO-10 ordering in Table 21.

ID	Task	$K = 1$	$K = 16$ mean
0	Put both the alphabet soup and the tomato sauce in the basket.	$8/10$	$9/10$
1	Put both the cream cheese box and the butter in the basket.	$10/10$	$10/10$
2	Turn on the stove and put the moka pot on it.	$9/10$	$10/10$
3	Put the black bowl in the bottom drawer of the cabinet and close it.	$10/10$	$10/10$
4	Put the white mug on the left plate and the yellow and white mug on the right plate.	$10/10$	$10/10$
5	Pick up the book and place it in the back compartment of the caddy.	$10/10$	$10/10$
6	Put the white mug on the plate and the chocolate pudding to the right of the plate.	$10/10$	$9/10$
7	Put both the alphabet soup and the cream cheese box in the basket.	$10/10$	$10/10$
8	Put both moka pots on the stove.	$9/10$	$10/10$
9	Put the yellow and white mug in the microwave and close it.	$10/10$	$10/10$
Mean success (%)		96.00	98.00

A.3 Exploration with Controllable Sampling Noise

We evaluate stochastic exploration on state-based Tool-Hang with Chi-Transformer. We compare native Flow-Policy sampling with Gaussian perturbations of HT-Policy action predictions, scaled either by the learned $\sigma(o)$ or by a fixed scale matched to the noiseless baseline. We measure exploration using pass@k, the estimated probability of at least one successful rollout within k attempts.

Table 11: **Exploration pass@k (%) on state-based Tool-Hang with Chi-Transformer.** Estimates are computed per initial state from eight rollouts and averaged over 100 states. α controls HT-Policy sampling noise. Dashes indicate unreported values or an inapplicable noise multiplier.

Method	α	pass@1	pass@2	pass@4	pass@8
Flow-Policy (native)	—	76.25	93.68	99.67	100.00
HT-Policy (no noise)	0	81.00	—	—	—
HT-Policy (σ-scaled)	0.125	79.38	95.61	99.76	100.00
HT-Policy (fixed scale)	0.125	78.88	95.39	99.81	100.00
HT-Policy (σ-scaled)	0.25	78.63	95.82	99.90	100.00
HT-Policy (fixed scale)	0.25	76.88	94.25	99.56	100.00
HT-Policy (σ-scaled)	0.5	78.13	94.68	99.64	100.00
HT-Policy (fixed scale)	0.5	79.38	95.79	99.93	100.00
HT-Policy (σ-scaled)	1	75.88	94.43	99.76	100.00
HT-Policy (fixed scale)	1	78.75	95.11	99.60	100.00
HT-Policy (σ-scaled)	2	67.13	89.46	99.16	100.00
HT-Policy (fixed scale)	2	54.88	79.18	95.71	100.00

Table 11 shows that modest noise gives HT-Policies pass@k comparable to Flow-Policies. Both stochastic HT-Policy variants reach 100% at pass@8 across the tested noise strengths, showing that direct action prediction can support successful stochastic exploration in this setting.

A.4 Policy Training and Evaluation Settings

This section specifies the data, trainable modules, and evaluation protocols for Sections 5.1 and 5.2. Throughout this appendix, H is the supervised action-chunk length, D_a is the number of valid action channels, and $d = HD_a$.

For Table 2, we evaluate each trained model using ten evaluation seeds and report the mean success rate and standard error across seeds. Table 14 lists the rollout budgets per task and seed, executed action horizons, and Flow-Policy sampling steps. MSE-Policies and HT-Policies predict actions in one forward pass.

A.4.1 REAL-ROBOT EXPERIMENTS

Push-T, Peel-Note, and Insert-T use a PiperX robot with a front-view Logitech C920s camera. Each task has 100 demonstrations collected using Quest 3-assisted VR teleoperation. All methods use the same demonstrations and train U-Net policies from scratch. The 640×480 RGB images are cropped to 480×480 and resized to 256×256. Policies also receive a ten-dimensional state: end-effector position, 6D orientation, and gripper opening width.

In Push-T, the robot pushes a T-shaped object into alignment with a target pose. The object's initial position is sampled within a 10×10 cm bounding box, with an in-plane rotation sampled from $[-30°, 30°]$.

In Insert-T, the robot grasps and lifts a T-shaped object, then places it into a holder. The object's initial position is sampled within a 16×5 cm region to the right of the holder, with an in-plane rotation sampled from $[-45°, 45°]$.

In Peel-Note, the robot uses its gripper to peel a note off the tabletop. The note's initial position is sampled within a 5×5 cm region.

The control loop executes actions at 30 Hz. Each policy call predicts a 16-step chunk, of which the first eight actions are executed before replanning. Push-T uses the gripper instead of DP's cylindrical end effector. For a fair comparison, all methods use the same gripper during training and evaluation. Each method is evaluated in 50 rollouts per task.

A.4.2 SCALE PREDICTION

Each HT-Policy predicts one positive scalar scale $\sigma_\theta(o)$ for each observation and action chunk, shared across its d valid action coordinates. The scale head is trained jointly with the action predictor through the HT loss. At inference, only the predicted actions are used.

$\pi_{0.5}$**.** The scale head first pools its raw outputs into a scalar and then applies softplus. Its final-layer bias is initialized to -0.396.

GR00T N1.7. The scale head applies softplus before pooling. Let $z_j(o)$ be its raw output for action coordinate j, and let m_j indicate whether that coordinate is supervised. The scalar scale is

$$\sigma_\theta(o) = \frac{\sum_j m_j \,\mathrm{softplus}(z_j(o))}{\sum_j m_j} + 10^{-3}.$$

The average uses only supervised coordinates. We initialize the scale to the action-residual RMS before fine-tuning, with biases -1.088, -0.4535, and -0.5093 for RoboCasa-GR1, Bridge, and Fractal, respectively.

Cosmos3-Nano. The scale head maps action features to a positive scalar scale through softplus. Its final-layer weights are initialized to zero. We calibrate the initial bias on 64 examples to match the residual RMS, giving $b_0 = -0.17432$.

Learning rate. The scale head shares the network's learning rate in all settings except Fractal. There, it uses 1.5×10^{-4}, while all other parameter groups use 10^{-4}.

A.4.3 $\pi_{0.5}$ ON LIBERO

Table 12: **Training budgets for the large-model experiments.** Batch size is global. Cosmos3-Nano batch size depends on packing.

Model / dataset	Updates	Batch	Base LR	Warmup	ν
$\pi_{0.5}$ / LIBERO	30,000	32	2.5×10^{-5}	1,000	1,400
GR00T / RoboCasa-GR1	60,000	512	10^{-4}	3,000	928
GR00T / Bridge	20,000	1,024	10^{-4}	1,000	224
GR00T / Fractal	20,000	1,024	10^{-4}	1,000	224
Cosmos3-Nano / LIBERO-10	2,000	$\leq 2,048$	5×10^{-5}	500	640

Table 13: **Training data and effective action dimensions.** Episode and frame counts refer to the datasets used for training.

Dataset	Episodes	Frames	H	D_a	d
LIBERO, four suites ($\pi_{0.5}$)	1,693	273,465	50	7	350
RoboCasa-GR1	24,000	5,820,277	8	29	232
Bridge	53,192	1,893,026	8	7	56
Fractal	87,212	3,786,400	8	7	56
LIBERO-10 (Cosmos3-Nano)	379	101,469	16	10	160

Training. We train MSE-Policies and HT-Policies using the LeRobot fine-tuning recipe. Fine-tuning updates the full vision, language, and action-expert network with global batch size 32, BF16 precision, and no EMA. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.01, and gradient clipping at norm 1. After 1,000 warmup steps, the learning rate decays from 2.5×10^{-5} to 2.5×10^{-6} over the 30,000-step run.

Inputs and action prediction. Training covers all 40 LIBERO tasks (Table 13). Inputs contain two RGB views and an eight-dimensional state. Images are resized and padded to 224×224; this recipe disables training image augmentation. State and action channels use quantile normalization. The action chunk contains 50 seven-dimensional actions, so $d = 350$ and the main HT loss setting is $\nu = 1400$. HT-Policies and MSE-Policies use zero action input and timestep zero to predict clean actions in one forward pass. The Flow-Policy uses ten Euler steps.

Table 14: **Evaluation settings for the 10-seed comparison in Table 2.** Rollout budgets are per task and evaluation seed; actual GR00T rollout counts can slightly exceed these budgets. "Execute" is the number of actions applied before replanning. MSE-Policies and HT-Policies each use a single forward pass.

Model / benchmark	Tasks	Rollout budget/ task/seed	Execute	Flow steps	Step limit
$\pi_{0.5}$ / Spatial and Object	2×10	100	10	10	280
$\pi_{0.5}$ / Goal	10	100	10	10	300
$\pi_{0.5}$ / LIBERO-10	10	100	10	10	520
GR00T / RoboCasa-GR1	24	20	8	4	720
GR00T / Bridge–WidowX	7	50	4	4	300
GR00T / Fractal–Google Robot	6	100	1	4	300
Cosmos3-Nano / LIBERO-10	10	10	16	30	520

Evaluation. Evaluation covers LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10, with ten tasks in each suite. The environment uses the official initial-state banks and ten no-op settling steps. Each task is evaluated in 100 rollouts per evaluation seed. The policy executes ten actions before replanning. Episode limits are 280 steps for Spatial and Object, 300 for Goal, and 520 for LIBERO-10. The reported four-suite score averages the task rates within each suite and then averages the four suites, giving equal weight to all 40 tasks.

A.4.4 GR00T N1.7 ON ROBOCASA-GR1 AND SIMPLER

Trainable parameters. GR00T N1.7 uses a Cosmos-Reason2-2B vision-language backbone. The pretrained vision-language backbone is frozen. Training updates the action transformer, state and action projections, action decoder, and associated normalization layers.

Optimization. Training uses AdamW, cosine learning-rate decay, weight decay 10^{-5}, and gradient clipping at norm 1. We use BF16 precision. Table 12 gives the batch sizes, budgets, and warmup lengths. The state dropout probabilities are 0.2 on RoboCasa-GR1, 0.8 on Bridge, and 0.5 on Fractal. The HT loss uses $\nu = 4d$.

RoboCasa-GR1 data and actions. Training uses 1,000 demonstrations for each of 24 tasks, for a total of 24,000 episodes at 20 Hz. Each observation contains the current ego-view image, robot state, and task text. Each action has 29 supervised channels: seven for each arm, six for each hand, and three for the waist. Arm and hand actions are expressed relative to the current state; waist actions remain absolute. The supervised action chunk has eight steps, giving $d = 8 \times 29 = 232$.

SIMPLER data and actions. BridgeData V2 and Fractal/RT-1 provide the demonstrations for WidowX and Google Robot, respectively (Table 13). Both policies receive one RGB view, robot state, and instruction text. Both predict eight actions with seven channels each: three translation components, three rotation components, and gripper control. Their effective dimension is $d = 56$. Both datasets retain their incremental-action conventions during training and execution.

Preprocessing. GR00T uses percentile normalization, outlier clipping, and statistics computed from the fine-tuning data. Images are resized to a shortest edge of 256 pixels and cropped to 95% of that size. Training uses random crops and color jitter; evaluation uses center crops and the VLM's image processor. The full training datasets are used, with no separate policy validation dataset.

Closed-loop evaluation. On RoboCasa-GR1, each evaluation seed targets 20 rollouts per task, allows up to 720 environment steps, and executes eight actions before replanning.

SIMPLER uses the standard benchmark environments. The rollout budgets per task and seed are 50 for the seven Bridge tasks and 100 for the six Fractal tasks. Both use a 300-step limit, but WidowX executes four predicted actions before replanning and Google Robot executes one.

Actual rollout counts can slightly exceed the requested budgets. Each task's success rate uses the number of completed rollouts as its denominator. We average task rates equally within each seed, then report the mean and standard error across the ten evaluation seeds in Table 2.

WidowX images are resized to 256×256, and Google Robot images to 256×320, before policy preprocessing. We follow the benchmark action and gripper conventions. HT-Policies and MSE-Policies execute their predicted means; Flow-Policies use four sampling steps.

A.4.5 From-Scratch Policies on RoboCasa-GR1

The from-scratch experiments in Section 5.1 use the same GR00T N1.7 architecture, data, optimization, and evaluation protocols described in Appendix A.4.4, with a randomly initialized action head. The pretrained vision-language backbone remains frozen. Thus, "from scratch" refers to learning the action predictor from fixed VLM features, as in the main text.

A.4.6 Cosmos3-Nano on LIBERO-10

Joint video and action training. We retain the video flow-matching objective and train the action branch with the HT or MSE loss. The action branch receives zero action input at timestep zero and predicts clean normalized actions. Both branch losses have weight 10, with the standard auxiliary terms unchanged. Training updates the shared generation expert and modality projections, so video supervision also affects action prediction.

Optimization. Training uses FusedAdam with BF16 precision, base learning rate 5×10^{-5}, and an action learning-rate multiplier of five. The schedule has 500 warmup steps and a 16,000-step cycle. We train for 2,000 updates with effective batch size up to 2,048 (Table 12).

Actions and evaluation. The policy uses agent-view and wrist images from LIBERO-10. Actions have ten valid channels: translation, a six-dimensional rotation representation, and gripper control. A 16-step chunk gives $d = 160$ and $\nu = 640$. At inference, one joint forward pass uses the observed inputs, zero action input, and video noise to predict actions. No future ground-truth video or video generation is required. Video noise can affect actions through joint attention, so the policy remains stochastic despite using a single action-prediction pass.

The protocol uses the official LIBERO-10 initial states, ten settling steps and ten rollouts per task and evaluation seed. All 16 predicted actions are executed before replanning, with a 520-step episode limit. The Flow-Policy uses its native 30-step sampler.

A.5 Additional Comparisons with Generative Policies

A.5.1 Comparison with Diffusion Policy

Tables 15 and 16 give the task-level results for DP and HT-Policies with state and image observations used in Table 1. DP-C and DP-T are the U-Net and Transformer variants reported by Chi et al. (2023); HT-Policies use Chi-UNet and Chi-Transformer, respectively. The HT-Policy runs use three training seeds and 22 rollouts per evaluation, with fixed initial conditions. DP also used 22 initial

states for RoboMimic, as documented in the evaluation-code footnote in Section 5.2 of its extended paper.

Each cell reports best/last10: the highest evaluation score during training and the mean of the final ten evaluations, averaged over three training seeds. The RoboMimic mean covers its nine tasks; Push-T is listed separately and reports normalized coverage. Table 1 averages the state and image results equally for each backbone. Figure 1 further averages the best scores across the two backbones. All averages are computed before rounding.

Table 15: **DP and HT with state observations.** Bold marks the highest value per row and metric across both backbones, before rounding.

Task	U-Net		Transformer	
	DP-C best/last10	HT best/last10	DP-T best/last10	HT best/last10
Lift-PH	**1.00**/0.98	**1.00/1.00**	**1.00/1.00**	**1.00**/0.93
Lift-MH	**1.00**/0.97	**1.00/1.00**	**1.00/1.00**	**1.00**/0.96
Can-PH	**1.00**/0.96	**1.00/1.00**	**1.00/1.00**	**1.00**/0.94
Can-MH	**1.00**/0.96	**1.00/0.98**	**1.00**/0.94	**1.00/0.98**
Square-PH	**1.00**/0.93	**1.00/0.97**	**1.00**/0.89	0.97/0.88
Square-MH	0.97/0.82	**0.98/0.88**	0.95/0.81	0.88/0.74
Transport-PH	0.94/0.82	0.91/0.78	**1.00/0.84**	0.70/0.48
Transport-MH	0.68/**0.46**	**0.79**/0.45	0.62/0.35	0.36/0.14
Tool-Hang	0.50/0.30	0.88/0.65	**1.00/0.87**	0.97/0.80
RoboMimic mean	0.899/0.800	0.951/**0.857**	**0.952**/0.856	0.876/0.761
Push-T	0.95/0.91	0.96/0.93	0.95/0.79	**1.00/0.98**

Table 16: **DP and HT with image observations.** Bold marks the highest value per row and metric across both backbones, before rounding.

Task	U-Net		Transformer	
	DP-C best/last10	HT best/last10	DP-T best/last10	HT best/last10
Lift-PH	**1.00/1.00**	**1.00**/0.95	**1.00/1.00**	**1.00/1.00**
Lift-MH	**1.00/1.00**	**1.00**/0.95	**1.00**/0.99	**1.00**/0.99
Can-PH	**1.00**/0.97	**1.00**/0.94	**1.00**/0.98	**1.00/1.00**
Can-MH	**1.00**/0.96	**1.00**/0.92	**1.00/0.98**	**1.00**/0.95
Square-PH	0.98/0.92	**1.00**/0.95	**1.00**/0.90	**1.00/0.96**
Square-MH	0.98/0.84	**1.00/0.85**	0.94/0.80	0.95/**0.85**
Transport-PH	**1.00/0.93**	**1.00**/0.92	0.98/0.81	0.95/0.90
Transport-MH	**0.89/0.69**	0.82/0.56	0.73/0.50	0.86/0.63
Tool-Hang	**0.95**/0.73	0.77/0.65	0.76/0.47	0.91/**0.79**
RoboMimic mean	**0.978**/0.893	0.954/0.854	0.934/0.826	0.963/**0.897**
Push-T	0.91/0.84	0.93/0.89	0.78/0.66	**0.95/0.91**

A.5.2 Comparison with MIP

Comparison setup. We compare HT-Policies with Flow-Policies, MSE-Policies, Straight Flow, and MIP on nine RoboMimic tasks and Push-T. All baseline scores are taken from Tables 11 and 12 of Pan et al. (2026). Our tables label that work's Regression baseline as MSE. The RoboMimic labels PH and MH denote proficient-human and multi-human demonstrations. Each method is evaluated with three backbones: Sudeep-DiT, Chi-Transformer, and Chi-UNet. Tables 17 and 18 report results for state and image observations. Each cell gives best/last5: the highest evaluation score during training and the mean of the final five evaluations, averaged over three training seeds. The Mean column reports the average score over the ten tasks. RoboMimic uses binary success, while Push-T uses normalized coverage.

Performance comparison. HT-Policies use a single-pass regression head and perform on par with MIP overall. The bottom block of each table reports the best score across backbones for each method, task, and metric. Averaged over the ten tasks and both observation types, these scores are $91.3\%/85.5\%$ for HT-Policies and $91.5\%/86.1\%$ for MIP. HT-Policies score slightly lower on state observations and slightly higher on image observations.

Table 17: **Scores on state-based tasks.** Entries report best/last5, averaged over three seeds. RoboMimic reports success; Push-T reports normalized coverage. Baselines follow Table 11 of Pan et al. (2026). Bold marks the best value per task and metric across all methods and backbones, before rounding.

Method	Lift		Can		Square		Transport		Tool-Hang	Push-T	Mean
	PH	MH	PH	MH	PH	MH	PH	MH			
Sudeep-DiT											
Flow	**1.00/1.00**	**1.00**/0.99	**1.00/1.00**	**1.00**/0.94	**1.00/0.94**	0.88/0.75	0.80/0.70	0.40/0.27	0.86/0.75	**0.98**/0.95	0.89/0.83
MSE	**1.00/1.00**	**1.00**/0.99	**1.00**/0.98	0.92/0.90	0.94/0.86	0.72/0.53	0.50/0.44	0.12/0.06	0.52/0.39	0.92/0.83	0.76/0.70
Straight Flow	**1.00/1.00**	**1.00**/0.98	**1.00**/0.99	0.96/0.90	0.96/0.93	0.72/0.66	0.56/0.48	0.20/0.14	0.70/0.59	0.90/0.86	0.80/0.75
MIP	**1.00/1.00**	**1.00**/0.99	**1.00/1.00**	0.98/0.95	0.98/**0.94**	0.90/0.81	0.76/0.68	0.44/0.38	**0.92/0.88**	0.95/0.92	0.89/0.86
HT (ours)	**1.00**/0.98	**1.00**/1.00	**1.00**/0.97	0.98/0.90	0.89/0.82	0.72/0.63	0.54/0.45	0.33/0.21	0.65/0.56	0.97/**0.95**	0.81/0.75
Chi-Transformer											
Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.98	**1.00**/0.93	0.96/0.89	0.78/0.74	**0.88**/0.64	0.44/0.34	0.68/0.54	0.91/0.89	0.87/0.80
MSE	**1.00**/0.99	**1.00**/0.99	**1.00**/0.96	0.98/0.92	0.92/0.85	0.74/0.61	0.68/0.51	0.28/0.20	0.40/0.36	0.93/0.88	0.79/0.73
Straight Flow	**1.00/1.00**	**1.00**/0.99	**1.00**/0.99	0.98/0.92	0.96/0.89	0.68/0.58	0.62/0.54	0.24/0.16	0.60/0.55	0.94/0.90	0.80/0.75
MIP	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	0.96/0.95	0.96/0.89	0.86/0.73	0.80/0.68	0.42/0.37	0.76/0.69	0.94/0.92	0.87/0.82
HT (ours)	**1.00**/0.98	**1.00**/0.99	**1.00**/0.96	**1.00**/0.95	0.95/0.86	0.72/0.60	0.63/0.48	0.33/0.22	0.88/0.81	0.96/0.93	0.85/0.78
Chi-UNet											
Flow	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/0.98**	0.98/**0.94**	0.90/0.78	0.80/**0.73**	0.52/0.40	0.84/0.70	**0.98**/0.94	0.90/0.85
MSE	**1.00/1.00**	**1.00/1.00**	**1.00**/0.99	**1.00**/0.96	**1.00**/0.91	**0.94/0.82**	0.64/0.55	0.22/0.16	0.68/0.64	0.95/0.89	0.84/0.79
Straight Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.99	**1.00**/0.92	0.98/0.90	**0.94**/0.79	0.64/0.52	0.22/0.15	0.50/0.00	0.93/0.88	0.82/0.72
MIP	**1.00/1.00**	**1.00/1.00**	**1.00**/0.99	**1.00/0.98**	**1.00/0.94**	0.92/0.81	0.80/0.69	0.62/0.46	0.80/0.64	0.97/0.95	0.91/0.85
HT (ours)	**1.00**/1.00	**1.00/1.00**	**1.00**/0.99	**1.00**/0.97	0.98/0.92	0.87/0.74	0.78/0.67	**0.63/0.49**	0.72/0.56	0.93/0.91	0.89/0.82
Best backbone											
Flow	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/0.98**	**1.00/0.94**	0.90/0.78	**0.88/0.73**	0.52/0.40	0.86/0.75	**0.98**/0.95	0.91/0.85
MSE	**1.00/1.00**	**1.00/1.00**	**1.00**/0.99	**1.00**/0.96	**1.00**/0.91	**0.94/0.82**	0.68/0.55	0.28/0.20	0.68/0.64	0.95/0.89	0.85/0.80
Straight Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.99	**1.00**/0.92	0.98/0.93	**0.94**/0.79	0.64/0.54	0.24/0.16	0.70/0.59	0.94/0.90	0.84/0.78
MIP	**1.00/1.00**	**1.00/1.00**	**1.00/1.00**	**1.00/0.98**	**1.00/0.94**	0.92/0.81	0.80/0.69	0.62/0.46	**0.92/0.88**	0.97/0.95	0.92/0.87
HT (ours)	**1.00**/1.00	**1.00/1.00**	**1.00**/0.99	**1.00**/0.97	0.98/0.92	0.87/0.74	0.78/0.67	**0.63/0.49**	0.88/0.81	0.97/**0.95**	0.91/0.85

Table 18: **Scores on image-based tasks.** Entries report best/last5, averaged over three seeds. RoboMimic reports success; Push-T reports normalized coverage. Baselines follow Table 12 of Pan et al. (2026). Bold marks the best value per task and metric across all methods and backbones, before rounding.

Method	Lift		Can		Square		Transport		Tool-Hang	Push-T	Mean
	PH	MH	PH	MH	PH	MH	PH	MH			
Sudeep-DiT											
Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.99	0.96/0.94	0.96/**0.94**	0.82/0.76	0.84/0.83	0.32/0.20	**0.78**/0.57	**0.92/0.89**	0.86/0.81
MSE	**1.00/1.00**	**1.00**/0.99	**1.00/1.00**	0.92/0.81	0.94/0.84	0.74/0.67	0.74/0.56	0.14/0.08	0.28/0.18	0.83/0.77	0.76/0.69
Straight Flow	**1.00**/0.99	**1.00**/0.99	**1.00**/0.98	0.98/0.95	**1.00**/0.93	0.82/0.72	0.86/0.83	0.26/0.19	0.46/0.40	0.85/0.79	0.82/0.78
MIP	**1.00/1.00**	**1.00**/0.99	**1.00**/0.98	**1.00**/0.96	**1.00**/0.92	0.90/0.83	0.90/0.84	0.50/0.31	0.76/**0.66**	0.91/0.87	0.90/0.84
HT (ours)	**1.00/1.00**	**1.00**/0.99	**1.00**/0.98	0.99/0.94	0.96/0.88	0.83/0.73	0.90/0.83	0.72/0.58	0.75/0.66	0.91/0.87	0.91/0.85
Chi-Transformer											
Flow	**1.00/1.00**	**1.00**/0.99	**1.00**/0.96	0.98/0.92	0.98/0.93	0.70/0.66	0.80/0.77	0.24/0.22	0.54/0.40	0.89/0.85	0.81/0.77
MSE	**1.00**/0.98	**1.00**/0.98	**1.00**/0.96	**1.00**/0.94	0.98/0.90	0.76/0.70	0.94/0.87	0.40/0.27	0.44/0.36	0.85/0.81	0.84/0.78
Straight Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.98	**1.00**/0.95	0.98/**0.94**	0.90/0.78	0.86/0.70	0.32/0.25	0.36/0.28	0.86/0.80	0.83/0.77
MIP	**1.00/1.00**	**1.00**/0.98	**1.00**/0.98	0.96/0.91	0.90/0.04	0.72/0.21	0.86/0.69	0.18/0.06	0.60/0.48	0.87/0.83	0.81/0.62
HT (ours)	**1.00**/0.99	**1.00**/0.99	**1.00**/0.97	0.99/0.93	0.98/0.90	0.85/0.75	0.90/0.83	**0.78/0.66**	0.54/0.38	0.89/0.82	0.89/0.82
Chi-UNet											
Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.98	**1.00/0.97**	0.96/0.90	0.90/0.79	0.78/0.61	0.24/0.16	0.48/0.37	**0.92**/0.87	0.83/0.77
MSE	**1.00**/0.99	**1.00**/0.96	0.98/0.87	0.84/0.70	0.94/0.86	0.74/0.66	0.66/0.64	0.18/0.10	0.30/0.23	0.90/0.85	0.75/0.69
Straight Flow	**1.00**/0.99	**1.00**/0.94	**1.00**/0.96	0.98/0.93	0.92/0.62	0.72/0.68	0.50/0.22	0.00/0.00	0.06/0.02	0.87/0.82	0.71/0.62
MIP	**1.00/1.00**	**1.00/1.00**	**1.00**/0.98	**1.00**/0.95	0.96/0.91	**0.92/0.84**	**0.96/0.91**	0.52/0.37	0.56/0.50	0.83/0.78	0.88/0.82
HT (ours)	**1.00**/0.97	**1.00**/0.94	0.97/0.88	0.93/0.85	0.90/0.81	0.76/0.64	0.87/0.76	0.63/0.49	0.62/0.47	0.85/0.80	0.85/0.76
Best backbone											
Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.99	**1.00/0.97**	0.98/**0.94**	0.90/0.79	0.84/0.83	0.32/0.22	**0.78**/0.57	**0.92/0.89**	0.87/0.82
MSE	**1.00/1.00**	**1.00**/0.99	**1.00/1.00**	**1.00**/0.94	0.98/0.90	0.76/0.70	0.94/0.87	0.40/0.27	0.44/0.36	0.90/0.85	0.84/0.79
Straight Flow	**1.00/1.00**	**1.00/1.00**	**1.00**/0.98	**1.00**/0.95	**1.00/0.94**	0.90/0.78	0.86/0.83	0.32/0.25	0.46/0.40	0.87/0.82	0.84/0.80
MIP	**1.00/1.00**	**1.00/1.00**	**1.00**/0.98	**1.00**/0.96	**1.00**/0.92	**0.92/0.84**	**0.96/0.91**	0.52/0.37	0.76/**0.66**	0.91/0.87	0.91/0.85
HT (ours)	**1.00/1.00**	**1.00**/0.99	**1.00**/0.98	0.99/0.94	0.98/0.90	0.85/0.75	0.90/0.83	**0.78/0.66**	0.75/0.66	0.91/0.87	0.92/0.86

A.5.3 RESULTS FOR $\pi_{0.5}$ ON LIBERO

Table 19 compares the official OpenPI Flow-Policy scores with our Flow-Policy and HT-Policy results across the four LIBERO suites. The policies use the LeRobot training recipe. Tables 20 and 21 give the Flow-Policy and HT-Policy scores for all 40 tasks as mean $\pm$ standard error across 10 evaluation seeds for each trained model. The overall means match Table 2.

Table 19: **Success rates (%) of $\pi_{0.5}$ across LIBERO suites.** Flow and HT report mean $\pm$ standard error across 10 evaluation seeds for each trained model. Bold marks the higher mean of Flow and HT.

Suite	Flow (OpenPI)	Flow	HT
LIBERO-Spatial	98.80	97.13 ± 0.11	$\mathbf{97.71 \pm 0.13}$
LIBERO-Object	98.20	99.44 ± 0.06	$\mathbf{99.80 \pm 0.00}$
LIBERO-Goal	98.00	96.55 ± 0.19	$\mathbf{96.94 \pm 0.23}$
LIBERO-10	92.40	$\mathbf{95.23 \pm 0.14}$	93.44 ± 0.06
Mean	96.85	$\mathbf{97.09 \pm 0.04}$	96.97 ± 0.07

Table 20: **Task-level success rates (%) of $\pi_{0.5}$ on LIBERO-Spatial and LIBERO-Object.** Task scores are mean $\pm$ standard error over 10 evaluation seeds for each trained model. Bold marks the higher mean for each task.

ID	Task	Flow	HT
LIBERO-Spatial			
0	Pick up the black bowl between the plate and the ramekin and place it on the plate.	$\mathbf{99.90 \pm 0.10}$	$\mathbf{99.90 \pm 0.10}$
1	Pick up the black bowl next to the ramekin and place it on the plate.	$\mathbf{99.60 \pm 0.22}$	99.30 ± 0.21
2	Pick up the black bowl from table center and place it on the plate.	$\mathbf{99.70 \pm 0.21}$	$\mathbf{99.70 \pm 0.15}$
3	Pick up the black bowl on the cookie box and place it on the plate.	$\mathbf{99.40 \pm 0.22}$	98.80 ± 0.36
4	Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate.	$\mathbf{94.20 \pm 0.39}$	94.10 ± 0.64
5	Pick up the black bowl on the ramekin and place it on the plate.	81.40 ± 0.75	$\mathbf{87.70 \pm 0.97}$
6	Pick up the black bowl next to the cookie box and place it on the plate.	$\mathbf{100.00 \pm 0.00}$	99.80 ± 0.13
7	Pick up the black bowl on the stove and place it on the plate.	99.30 ± 0.26	$\mathbf{99.50 \pm 0.17}$
8	Pick up the black bowl next to the plate and place it on the plate.	98.60 ± 0.31	$\mathbf{99.00 \pm 0.33}$
9	Pick up the black bowl on the wooden cabinet and place it on the plate.	99.20 ± 0.25	$\mathbf{99.30 \pm 0.21}$
Suite mean		97.13 ± 0.11	$\mathbf{97.71 \pm 0.13}$
LIBERO-Object			
0	Pick up the alphabet soup and place it in the basket.	99.50 ± 0.22	$\mathbf{100.00 \pm 0.00}$
1	Pick up the cream cheese and place it in the basket.	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
2	Pick up the salad dressing and place it in the basket.	99.50 ± 0.22	$\mathbf{100.00 \pm 0.00}$
3	Pick up the BBQ sauce and place it in the basket.	97.60 ± 0.56	$\mathbf{98.00 \pm 0.00}$
4	Pick up the ketchup and place it in the basket.	99.10 ± 0.18	$\mathbf{100.00 \pm 0.00}$
5	Pick up the tomato sauce and place it in the basket.	99.70 ± 0.15	$\mathbf{100.00 \pm 0.00}$
6	Pick up the butter and place it in the basket.	99.70 ± 0.15	$\mathbf{100.00 \pm 0.00}$
7	Pick up the milk and place it in the basket.	99.80 ± 0.13	$\mathbf{100.00 \pm 0.00}$
8	Pick up the chocolate pudding and place it in the basket.	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
9	Pick up the orange juice and place it in the basket.	99.50 ± 0.22	$\mathbf{100.00 \pm 0.00}$
Suite mean		99.44 ± 0.06	$\mathbf{99.80 \pm 0.00}$

Table 21: **Task-level success rates (%) of $\pi_{0.5}$ on LIBERO-Goal and LIBERO-10.** Task scores are mean ± standard error over 10 evaluation seeds for each trained model. Bold marks the higher mean for each task.

ID	Task	Flow	HT
LIBERO-Goal			
0	Open the middle drawer of the cabinet.	96.10 ± 0.46	$\mathbf{99.40 \pm 0.16}$
1	Put the bowl on the stove.	98.20 ± 0.53	$\mathbf{98.50 \pm 0.48}$
2	Put the wine bottle on top of the cabinet.	89.90 ± 0.60	$\mathbf{90.30 \pm 1.16}$
3	Open the top drawer and put the bowl inside.	92.50 ± 0.48	$\mathbf{95.40 \pm 0.85}$
4	Put the bowl on top of the cabinet.	99.40 ± 0.16	$\mathbf{99.50 \pm 0.27}$
5	Push the plate to the front of the stove.	$\mathbf{98.80 \pm 0.36}$	98.30 ± 0.37
6	Put the cream cheese in the bowl.	97.70 ± 0.37	$\mathbf{98.70 \pm 0.37}$
7	Turn on the stove.	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
8	Put the bowl on the plate.	98.90 ± 0.31	$\mathbf{99.70 \pm 0.15}$
9	Put the wine bottle on the rack.	$\mathbf{94.00 \pm 0.94}$	89.60 ± 0.86
Suite mean		96.55 ± 0.19	$\mathbf{96.94 \pm 0.23}$
LIBERO-10			
0	Living room 2: Put both the alphabet soup and the tomato sauce in the basket.	$\mathbf{94.50 \pm 0.54}$	92.00 ± 0.00
1	Living room 2: Put both the cream cheese box and the butter in the basket.	$\mathbf{98.30 \pm 0.26}$	96.00 ± 0.00
2	Kitchen 3: Turn on the stove and put the moka pot on it.	96.80 ± 0.39	$\mathbf{97.90 \pm 0.38}$
3	Kitchen 4: Put the black bowl in the bottom drawer of the cabinet and close it.	99.00 ± 0.33	$\mathbf{99.40 \pm 0.16}$
4	Living room 5: Put the white mug on the left plate and put the yellow and white mug on the right plate.	97.60 ± 0.16	$\mathbf{100.00 \pm 0.00}$
5	Study 1: Pick up the book and place it in the back compartment of the caddy.	$\mathbf{99.30 \pm 0.30}$	98.50 ± 0.34
6	Living room 6: Put the white mug on the plate and put the chocolate pudding to the right of the plate.	$\mathbf{94.80 \pm 0.59}$	94.00 ± 0.00
7	Living room 1: Put both the alphabet soup and the cream cheese box in the basket.	96.40 ± 0.34	$\mathbf{98.00 \pm 0.00}$
8	Kitchen 8: Put both moka pots on the stove.	$\mathbf{80.10 \pm 1.43}$	69.40 ± 0.87
9	Kitchen 6: Put the yellow and white mug in the microwave and close it.	$\mathbf{95.50 \pm 0.58}$	89.20 ± 0.71
Suite mean		$\mathbf{95.23 \pm 0.14}$	93.44 ± 0.06

A.5.4 TASK-LEVEL RESULTS FOR GR00T N1.7

Tables 22, 23, and 24 compare Flow-Policies and HT-Policies on RoboCasa-GR1, SIMPLER / Bridge, and SIMPLER / Fractal, with officially reported Flow-Policy scores as reference. These experiments fine-tune GR00T N1.7 from pretrained action-head weights. The official Flow-Policy averages match those in Table 2. We report Flow-Policy and HT-Policy scores as mean ± standard error across 10 evaluation seeds for each trained model, matching Table 2.

Table 22: **Task-level success rates (%) on RoboCasa-GR1.** Flow and HT report mean ± standard error across 10 evaluation seeds for each trained model. Bold marks the higher mean of Flow and HT.

Task	Flow (official)	Flow	HT
PnPBottle → CabinetClose	70.00	42.00 ± 4.23	$\mathbf{58.26 \pm 5.53}$
PnPCan → DrawerClose	70.00	23.90 ± 2.68	$\mathbf{50.64 \pm 4.51}$
PnPCup → DrawerClose	35.00	8.40 ± 1.83	$\mathbf{23.58 \pm 3.32}$
PnPMilk → MicrowaveClose	45.00	11.00 ± 2.56	$\mathbf{25.00 \pm 4.22}$
PnPPotato → MicrowaveClose	40.00	6.00 ± 1.25	$\mathbf{26.31 \pm 3.84}$
PnPWine → CabinetClose	65.00	24.88 ± 2.98	$\mathbf{49.74 \pm 1.60}$
Cuttingboard → Basket	10.00	$\mathbf{78.50 \pm 3.50}$	73.00 ± 3.18
Cuttingboard → Cardboardbox	30.00	31.32 ± 2.10	$\mathbf{40.50 \pm 2.63}$
Cuttingboard → Pan	40.00	$\mathbf{83.50 \pm 3.25}$	78.07 ± 2.52
Cuttingboard → Pot	45.00	$\mathbf{73.50 \pm 2.69}$	72.50 ± 3.27
Cuttingboard → Tieredbasket	25.00	28.61 ± 3.42	$\mathbf{34.50 \pm 1.89}$
Placemat → Basket	40.00	63.50 ± 2.99	$\mathbf{67.50 \pm 2.71}$
Placemat → Bowl	40.00	35.40 ± 2.48	$\mathbf{44.00 \pm 2.45}$
Placemat → Plate	40.00	$\mathbf{42.50 \pm 3.52}$	41.00 ± 2.33
Placemat → Tieredshelf	18.20	12.06 ± 2.47	$\mathbf{20.49 \pm 2.42}$
Plate → Bowl	50.00	39.00 ± 2.33	$\mathbf{42.50 \pm 3.27}$
Plate → Cardboardbox	35.00	$\mathbf{35.40 \pm 2.48}$	32.71 ± 4.16
Plate → Pan	40.00	40.71 ± 3.40	$\mathbf{48.76 \pm 3.32}$
Plate → Plate	75.00	$\mathbf{76.21 \pm 3.34}$	73.00 ± 3.09
Tray → Cardboardbox	60.00	67.05 ± 4.03	$\mathbf{70.50 \pm 2.93}$
Tray → Plate	50.00	67.50 ± 3.44	$\mathbf{72.00 \pm 3.89}$
Tray → Pot	45.00	69.50 ± 4.25	$\mathbf{76.50 \pm 2.89}$
Tray → Tieredbasket	55.00	39.81 ± 4.38	$\mathbf{53.00 \pm 2.60}$
Tray → Tieredshelf	45.00	15.81 ± 3.66	$\mathbf{26.17 \pm 4.24}$
Mean	44.50	42.34 ± 0.68	$\mathbf{50.01 \pm 0.51}$

Table 23: **Task-level success rates (%) on SIMPLER / Bridge.** Flow and HT: mean ± standard error over 10 evaluation seeds. Mean averages seven tasks; bold marks the higher mean of Flow and HT.

Task	Flow (official)	Flow	HT
carrot on plate	58.00	$\mathbf{63.00 \pm 1.94}$	29.80 ± 1.17
close drawer	97.00	98.40 ± 0.50	$\mathbf{98.80 \pm 0.53}$
put eggplant in basket	53.00	50.80 ± 2.33	$\mathbf{93.03 \pm 1.24}$
put eggplant in sink	2.00	2.00 ± 0.84	$\mathbf{65.00 \pm 1.82}$
open drawer	100.00	$\mathbf{100.00 \pm 0.00}$	72.60 ± 1.19
spoon on towel	78.00	$\mathbf{87.02 \pm 1.13}$	61.40 ± 1.98
stack cube	48.00	$\mathbf{33.20 \pm 1.50}$	15.57 ± 0.66
Mean	62.29	62.06 ± 0.40	$\mathbf{62.31 \pm 0.65}$

A.5.5 RESULTS FOR COSMOS3-NANO ON LIBERO-10

Table 25 reports task-level success as mean ± standard error across 10 evaluation seeds for each trained model, using ten rollouts per task and seed. The overall results match Table 2.

Table 24: **Task-level success rates (%) on SIMPLER / Fractal.** Flow and HT: mean ± standard error over 10 evaluation seeds. Mean averages six tasks; bold marks the higher mean of Flow and HT.

Task	Flow (official)	Flow	HT
pick coke can	100.00	$\mathbf{97.90 \pm 0.41}$	97.80 ± 0.57
pick object	94.00	$\mathbf{90.11 \pm 0.98}$	72.30 ± 1.01
move near	100.00	$\mathbf{98.50 \pm 0.50}$	93.10 ± 0.87
open drawer	65.00	$\mathbf{52.10 \pm 1.21}$	46.10 ± 1.33
close drawer	69.00	58.84 ± 0.97	$\mathbf{66.20 \pm 1.19}$
place in closed drawer	7.00	9.50 ± 1.01	$\mathbf{16.00 \pm 0.54}$
Mean	72.50	$\mathbf{67.83 \pm 0.39}$	65.25 ± 0.44

Table 25: **Task-level success rates (%) of Cosmos3-Nano on LIBERO-10.** Mean ± standard error over 10 evaluation seeds; bold marks the higher mean per row.

ID	Task	Flow	HT
0	Put both the alphabet soup and the tomato sauce in the basket.	92.00 ± 4.16	$\mathbf{95.00 \pm 2.24}$
1	Put both the cream cheese box and the butter in the basket.	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
2	Turn on the stove and put the moka pot on it.	94.00 ± 2.21	$\mathbf{99.00 \pm 1.00}$
3	Put the black bowl in the bottom drawer of the cabinet and close it.	94.00 ± 2.21	$\mathbf{97.00 \pm 1.53}$
4	Put the white mug on the left plate and put the yellow and white mug on the right plate.	95.00 ± 1.67	$\mathbf{96.00 \pm 2.21}$
5	Pick up the book and place it in the back compartment of the caddy.	$\mathbf{99.00 \pm 1.00}$	98.00 ± 1.33
6	Put the white mug on the plate and put the chocolate pudding to the right of the plate.	$\mathbf{92.00 \pm 3.27}$	88.00 ± 2.00
7	Put both the alphabet soup and the cream cheese box in the basket.	$\mathbf{99.00 \pm 1.00}$	$\mathbf{99.00 \pm 1.00}$
8	Put both moka pots on the stove.	$\mathbf{97.00 \pm 2.13}$	96.00 ± 1.63
9	Put the yellow and white mug in the microwave and close it.	94.00 ± 2.21	$\mathbf{97.00 \pm 1.53}$
Mean		95.60 ± 0.90	$\mathbf{96.50 \pm 0.45}$

A.6 Training Curves and Inference Timing

Training-progress measurements. Figure 10 plots success against optimizer updates, rather than elapsed training time. GR1 is evaluated after 2k, 16k, 32k, and 60k updates; WidowX is evaluated after 5k, 10k, 15k, and 20k updates. Each evaluation uses a single seed, so these scores may differ from the 10-seed means in Table 2. The Transport runs use state observations, absolute actions, Chi-UNet, eight-step observation history, and eight executed actions. They train for 300k updates and evaluate every 20k updates, with 40 completed episodes per evaluation. Transport-PH has three seeds per method. Transport-MH has three Flow-Policy seeds, two MSE-Policy seeds, and seven HT-Policy seeds. The Transport curves average scores across seeds at each training stage. Flow-Policies use nine Euler steps for these Transport evaluations.

Hardware and measurement setup. Table 3 reports batch-one inference on an NVIDIA RTX 5090; Table 26 reports additional NVIDIA A800-SXM4-80GB measurements. Measurements use the same architectures as the policy evaluations, with BF16 mixed precision and no gradient computation or compilation. Latency measures a policy query; throughput is $1000/\mathrm{latency}_{\mathrm{ms}}$, measured in action chunks per second.

Repeated measurements. For A800 measurements, methods within each model use the same 16 observations. After 20 warmup calls, we time ten repetitions per observation and report the mean over 160 synchronized measurements.

Timing boundaries. Whole-model latency includes observation encoding and action prediction from preprocessed GPU inputs. It excludes input/output processing, environment execution, and video decoding. Head latency measures action prediction with observation features already computed. GR00T and $\pi_{0.5}$ skip scale prediction when evaluating HT-Policies.

Table 26: **Inference efficiency on an NVIDIA A800.** Batch size one. All latency values are reported in milliseconds. NFE denotes the number of action-head evaluations per action chunk. Speedup is the Flow-to-HT whole-model latency ratio.

		Action head		Whole model		
Model	Method	NFE↓	Latency↓	Latency↓	FPS↑	Speedup↑
GR00T N1.7	Flow	4	89.2	139.1	7.19	**1.94**×
	HT (ours)	**1**	**24.3**	**71.7**	**13.95**	
$\pi_{0.5}$	Flow	10	169.7	260.8	3.83	**2.42**×
	HT (ours)	**1**	**17.1**	**107.6**	**9.29**	
Cosmos3-Nano	Flow	30	1368.2	1377.5	0.73	**16.74**×
	HT (ours)	**1**	**72.3**	**82.3**	**12.15**	

Cosmos3-Nano timing configuration. For Cosmos3-Nano, the Flow-Policy uses 30 sampling steps, while the HT-Policy predicts actions in a single forward pass through the joint video/action network.

A.7 Additional Ablations and Settings

A.7.1 Regression Objectives

Scale adaptation and robust losses. The objective ablations in Table 27 show that learning an observation-dependent scale improves success for every tested objective. We attribute this consistent benefit to the heteroscedastic residuals observed in Section 3, whose typical magnitudes vary across observations. Different robust objectives (Barron, 2019) also improve over MSE, with heteroscedastic Student-t achieving the highest success. The only exception is fixed-scale L_1. L_1 penalizes coordinates independently, while RMSE uses a joint norm that makes each coordinate's gradient depend on the full residual vector. Replacing L_1 with RMSE improves success under both fixed and learned scales, supporting the value of treating the action residual jointly.

Table 27: **Regression loss ablation.** GR00T N1.7 on RoboCasa-GR1, 60k steps, 24 tasks. Upper/lower blocks omit/use input-dependent scales. Both Student-t variants use $\nu = 928$.

Objective	SR (%) ↑
MSE	37.80
L_1	27.00
RMSE	46.00
Huber Huber (1964)	38.10
Student-t	39.50
Heteroscedastic Gaussian	42.50
Heteroscedastic L_1	44.50
Heteroscedastic RMSE	47.67
Heteroscedastic Huber	40.70
Heteroscedastic Student-t (HT)	**51.50**

Common training setting. The objective comparison uses GR00T N1.7 on RoboCasa-GR1 after 60,000 updates, with the data, mask, and optimizer described in Appendix A.4. Success rates in Table 27 use a single evaluation seed per model, so they may differ from the 10-seed means in Table 2. Every action chunk has $d = 232$ valid coordinates. Both Student-t variants use $\nu = 928$. Table 28 specifies the implemented penalties. Scale-dependent terms are included during training; only constants independent of the optimized parameters are omitted.

Table 28: **Implemented regression penalties.** Penalties are shown before division by the number of valid action coordinates. Let $S = \|r\|_2^2$, $R = \|r\|_2$, $Q = \|r\|_1$, and $T_\nu(z) = \frac{\nu+d}{2}\log(1 + z^2/\nu)$. The learned scale $\sigma = \sigma(o)$ is scalar.

Family	Fixed-scale penalty	Learned-scale penalty
Gaussian	S	$S/(2\sigma^2) + d\log\sigma$
L_1	Q	$Q/\sigma + d\log\sigma$
Radial norm (RMSE)	R	$R/\sigma + d\log\sigma$
Huber	$h_\delta(R)$	$h_\delta(R/\sigma) + d\log\sigma$
Student-t	$T_\nu(R/\sigma_0) + d\log\sigma_0$	$T_\nu(R/\sigma) + d\log\sigma$

Coordinate and radial penalties. L_1 sums absolute errors independently across coordinates. The norm and Huber losses act on the whole chunk radius. The "RMSE" label denotes the radial-norm baseline: its implementation minimizes R/d, which differs from the conventional $R/\sqrt{d}$ by a fixed factor for this experiment. The corresponding likelihood is a spherical radial-exponential density, rather than a product of coordinate-wise Laplace densities. Huber uses $h_\delta(z) = z^2/2$ for $z \le \delta$ and $h_\delta(z) = \delta z - \delta^2/2$ otherwise. Its threshold is 2.5 times an exponential moving average of the batch median radius, with decay 0.99. The heteroscedastic version applies the same rule to the standardized radius.

Scale initialization. Fixed-scale Student-t uses $\sigma_0 \simeq 0.291346$. Heteroscedastic RMSE (Hetero-RMSE) requires a different scale convention from Gaussian or Student-t regression. We multiply its softplus scale, including the 10^{-3} floor, by a constant chosen on the initial training batch so that its mean equals $\mathbb{E}[R]/d$. The scale head remains trainable, while this multiplier stays fixed. This initialization is separate from the post-training likelihood calibration in Appendix A.7.2.

A.7.2 Likelihood Calibration

The success rates paired with likelihoods in Figure 9 use a single evaluation seed per model, whereas Table 2 averages 10 evaluation seeds.

A single global scale multiplier. For each frozen policy, we replace its residual scale $\sigma(o)$ with $g\sigma(o)$ and choose

$$g^* = \arg\max_{g>0} \mathbb{E}_{\text{cal}} \left[\log p\big(a \mid \mu(o), g\sigma(o)\big) \right]. \tag{8}$$

We use the full joint log-likelihood with all log-scale terms and normalization constants, using radial densities for RMSE and Huber. One g is shared across all tasks, observations, and coordinates, preserving relative scales. Policy weights, distribution family, ν, and Huber thresholds remain fixed. Calibration neither uses nor changes policy success rates.

Calibration and scoring data. We sample 512 demonstration episodes per GR1 task, totaling 12,288. Each contributes one eight-step chunk with 232 valid coordinates and four forward passes with training augmentation and dropout. We use four task-stratified folds, grouping all passes from each episode together. Each fold is scored with g fitted on the other three folds. These demonstrations were used in policy training; the held-out split applies only to scale calibration.

Reported likelihood. The horizontal axis in Figure 9 is $\mathbb{E}[\log p(a \mid o)]/232$, the mean joint log-likelihood per action dimension in nat/dim, averaged over forward passes and held-out folds. Passes are scored separately and do not define a mixture density. The Gaussian special case has $g^2 = \mathbb{E}_{\text{cal}}[\|r\|_2^2/\sigma(o)^2]/d$. For fixed-scale MSE, the calibrated standard deviation is the calibration residual RMS.

Huber thresholds and uncertainty. For likelihood calibration, the Huber threshold is fixed at 1.87205 for fixed-scale Huber and 34.1826 for heteroscedastic Huber (Hetero-Huber). Confidence intervals use 2,000 task-stratified episode bootstrap samples with each fitted g fixed. They reflect episode sampling, rather than variation across training seeds or control rollouts.

A.7.3 Degrees-of-Freedom Sweep Settings

Table 29 lists the ν values and training budgets for the points shown in Figure 11. Each point uses a single evaluation seed, so its success rate may differ from the 10-seed mean in Table 2. All variants learn observation-dependent scales. The GR1 and Bridge runs follow the corresponding GR00T recipes; the $\pi_{0.5}$ runs use 30k updates on all four LIBERO suites. All Bridge evaluations use the environment's default random initialization.

Table 29: **Degrees-of-freedom settings for Figure 11.** Bold marks the choice $\nu = 4d$ ($c = 4$), which gives the highest tested success in each setting. HG denotes the Gaussian limit.

Setting	Updates	d	Reported ν
RoboCasa-GR1	60,000	232	$2, 116, 464, \mathbf{928}, 1450, \infty$ (HG)
Bridge	20,000	56	$56, 112, \mathbf{224}, 350, 448$
$\pi_{0.5}$ / LIBERO	30,000	350	$700, \mathbf{1400}, 2187.5$