FLOWING WITH PURPOSE: LATENT ACTION GUIDED FLOW MATCHING POLICIES FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow matching has recently become a new standard for behavior cloning in robotic manipulation. However, state-of-the-art flow matching policies suffer from a systematic structural mismatch: they rely on a globally fixed isotropic source distribution despite the strongly fragmented and heteroscedastic structure of robotic action spaces. This agnostic initialization forces the model to learn highly entangled vector fields, bottlenecking training efficiency and limiting overall policy performance. To address this limitation, we introduce Latent Action Guided Flow Matching (LAFM), a novel framework that replaces the monolithic Gaussian with an adaptive library of learned prior distributions. By grounding these distributions using a latent action model, LAFM maps current observations to discrete motion primitives, selecting a specialized base distribution that provides an informed, structurally aligned initialization for the denoising process. This dynamic adaptivity naturally accommodates heteroscedasticity in human demonstrations and makes transport trajectories shorter and less entangled, reducing the number of training steps required to reach the final error of standard flow matching by 32.5%. Empirically, LAFM substantially outperforms standard flow matching formulations, increasing task success rates by 23.4 percentage points in real-world robotic deployments, 10.4 points on LIBERO-90, and 12 to 15 points on the more challenging LIBERO-Plus and RoboTwin 2.0 benchmarks. Code and model weights will be publicly released upon acceptance of the publication.

1 INTRODUCTION

Modeling visuomotor policy learning as a generative denoising process has recently emerged as a highly effective paradigm for robotic manipulation (Chi et al., 2023; Black et al., 2024; NVIDIA et al., 2025), with diffusion (Ho et al., 2020; Sohl-Dickstein et al., 2015) and flow matching (Lipman et al., 2023; Liu, 2022) policies establishing new performance standards across both simulated and real-world benchmarks. Among these approaches, flow matching offers a particularly attractive formulation by directly learning a continuous transport field from a simple source distribution to the target action distribution, enabling faster inference and more stable optimization than diffusion-based alternatives.

Despite their empirical success, current flow matching policies rely on a strong structural assumption: all action trajectories are generated from a single isotropic Gaussian prior. While this assumption is convenient, it becomes restrictive in robotic manipulation, where the conditional action distribution is inherently fragmented and heterogeneous. Under nearly identical visual observations, multiple valid motor futures may coexist, corresponding to different grasp strategies, motion phases, or object interaction intents. As a result, the model must learn transport fields that connect a single unstructured source distribution to sharply separated action modes, forcing nearby noise samples to evolve toward distant targets and yielding unnecessarily entangled vector fields.

This limitation is further amplified by the heteroscedastic nature of manipulation demonstrations. Certain motion patterns, such as repetitive reaching or grasp closure, exhibit low variance and strong consistency, whereas others—particularly long-horizon object interactions—remain substan-

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.

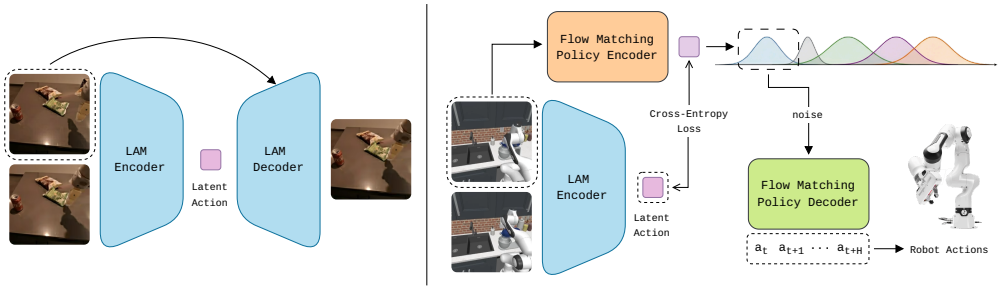


Figure 1: **LAFM overview.** Left: We train a LAM to extract latent actions from consecutive frames. Right: We divide our policy into two parts: The encoder predicts the next latent action for a given frame. The predicted latent action is used to condition the noise sampling, guiding the flow matching process in the decoder.

tially more diverse. A single fixed prior cannot capture such mode-dependent uncertainty, implicitly imposing homogeneous variance across fundamentally different motion regimes.

These observations suggest that the source distribution itself should be structured before transport begins. Rather than initiating all denoising trajectories from the same global Gaussian, we argue that flow matching should start from priors already aligned with coarse motion intent. A suitable partition variable for this purpose must capture high-level dynamics rather than instantaneous controls. Latent action models (LAM) (Bruce et al., 2024), recently introduced as unsupervised abstractions of temporal transitions, naturally provide such a representation by compressing visual dynamics into discrete motion primitives.

Building on this idea, we introduce Latent Action Guided Flow Matching (LAFM), a framework that replaces the monolithic Gaussian prior with a library of learned prior distributions indexed by latent actions. Instead of forcing the model to transport generic noise toward every possible action mode, LAFM first predicts a latent motion primitive from the current observation and then samples from a prior specialized for that primitive. This produces a more structured initialization for the generative process, yielding smoother transport trajectories and reducing vector field entanglement during training. An overview of the method is shown in Figure 1.

Our work makes four main contributions:

- We introduce LAFM, a new formulation of conditional flow matching in which the source distribution is no longer fixed but selected from a library of learnable priors. Each prior is associated with a latent motion primitive, allowing the generative process to begin from a distribution aligned with coarse action intent rather than from an uninformative global Gaussian.
- We propose a simple yet effective mechanism to ground these priors using a LAM. A policy encoder predicts discrete latent actions directly from current observations, and the predicted latent action determines which prior distribution parameterizes the noisy initialization of the decoder. This transforms latent actions from a pretraining signal into a direct structural bias on the geometry of the transport process.
- We show that structuring the source distribution substantially enhances final control performance in both simulation and real-world robotic manipulation settings. LAFM improves the success rate over standard flow matching by 23.4 percentage points in real-world experiments and 10.4 points on LIBERO-90, being competitive with significantly larger vision-language-action models (VLA) in both settings. It also yields gains of 15.1 and 12.3 points on the more challenging LIBERO-Plus and RoboTwin 2.0 benchmarks.
- We provide theoretical and empirical evidence that latent-conditioned priors reduce the transport burden of flow matching. LAFM reduces the kinetic energy of the learned flow by 67% and reaches the final action prediction error of standard flow matching with 32.5% fewer training steps.

2 METHODOLOGY

2.1 PRELIMINARY: THE FLOW MATCHING POLICY

We follow the flow matching policy formulation proposed by Black et al. (2024). Formally, the objective is to model the data distribution $p(\mathbf{A}_t|\mathbf{o}_t)$ of future actions $\mathbf{A}_t = [\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H-1}]$, conditioned on the current observation $\mathbf{o}_t$. The subscripts refer to robot control timesteps and H denotes the action horizon. A neural network parametrized by θ is optimized to minimize the conditional flow matching objective (Lipman et al., 2023; Liu, 2022):

$$\mathcal{L}^{\tau}_{FM}(\theta) = \mathbb{E}_{p(\mathbf{A}_t|\mathbf{o}_t), q(\mathbf{A}^{\tau}_t|\mathbf{A}_t)} ||\mathbf{v}_\theta(\mathbf{A}^{\tau}_t, \mathbf{o}_t) - \mathbf{u}(\mathbf{A}^{\tau}_t|\mathbf{A}_t)||^2 \tag{1}$$

The optimal transport probability path is defined as $q(\mathbf{A}^{\tau}_t|\mathbf{A}_t) = \mathcal{N}(\tau\mathbf{A}_t, (1-\tau)^2\mathbf{I})$, where $\tau \in [0, 1]$ represents the flow matching step. This path is realized by sampling standard Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to construct noisy action targets through linear interpolation: $\mathbf{A}^{\tau}_t = \tau\mathbf{A}_t + (1-\tau)\epsilon$. The flow matching step τ is sampled from a Beta distribution that prioritizes noisier samples (see Appendix I). The network output $\mathbf{v}_\theta(\mathbf{A}^{\tau}_t, \mathbf{o}_t)$ is explicitly optimized to regress the denoising vector field $\mathbf{u}(\mathbf{A}^{\tau}_t|\mathbf{A}_t) = \mathbf{A}_t - \epsilon$.

During inference, actions are predicted by integrating the vector field from $\tau = 0$ to $\tau = 1$. This is implemented using Euler integration rule: $\mathbf{A}^{\tau+\delta}_t = \mathbf{A}^{\tau}_t + \delta\mathbf{v}_\theta(\mathbf{A}^{\tau}_t, \mathbf{o}_t)$, where δ is the integration step size. Starting with $\mathbf{A}^0_t = \epsilon$, $1/\delta$ integration steps are performed to obtain the final prediction $\mathbf{A}^1_t$.

2.2 ADAPTIVE FLOW MATCHING WITH MULTIPLE PRIORS

Instead of using a single prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$, we propose using a library of K learned prior distributions $\{\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^K$. For each data sample, the network predicts a latent mode distribution $\pi(\mathbf{o}_t)$ conditioned on the current observation, where $\pi_k(\mathbf{o}_t)$ denotes the probability that the action belongs to latent mode k. An index k is then selected, and noise is sampled from the corresponding distribution: $\epsilon_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Because both the prior parameters and the selection mechanism are learned, the resulting probability path is adaptive and parameter-dependent: $q_\theta(\mathbf{A}^{\tau}_t|\mathbf{A}_t) = \mathcal{N}\left(\tau\mathbf{A}_t + (1-\tau)\boldsymbol{\mu}_k, (1-\tau)^2\boldsymbol{\Sigma}_k\right)$. As shown in Equation equation 2, the objective is to regress the adaptive denoising vector field $\mathbf{u}_\theta(\mathbf{A}^{\tau}_t|\mathbf{A}_t) = \mathbf{A}_t - \epsilon_k$.

$$\mathcal{L}^{\tau}_{AFM}(\theta) = \mathbb{E}_{p(\mathbf{A}_t|\mathbf{o}_t), q_\theta(\mathbf{A}^{\tau}_t|\mathbf{A}_t)} ||\mathbf{v}_\theta(\mathbf{A}^{\tau}_t, \mathbf{o}_t) - \mathbf{u}_\theta(\mathbf{A}^{\tau}_t|\mathbf{A}_t)||^2 \tag{2}$$

This structured source distribution admits a simple geometric interpretation: when each latent prior aligns with a distinct mode of the conditional action distribution, the expected transport burden imposed on the learned vector field is reduced. We formalize this intuition with Proposition 1.

Proposition 1 ***Expected transport reduction under latent-conditioned priors***

Let $p(\mathbf{A}_t|\mathbf{o}_t)$ denote a conditional action distribution decomposable into K latent motion modes: $p(\mathbf{A}_t|\mathbf{o}_t) = \sum_{k=1}^K \pi_k(\mathbf{o}_t)p_k(\mathbf{A}_t|\mathbf{o}_t)$, where each component $p_k(\mathbf{A}_t|\mathbf{o}_t)$ has mean $\mathbf{m}_k(\mathbf{o}_t)$. Consider two source constructions: a single isotropic prior ($\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) and a latent-conditioned prior ($\epsilon_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$) selected according to latent mode k. If each prior satisfies $\boldsymbol{\mu}_k \approx \mathbb{E}_{\mathbf{o}_t}[\mathbf{m}_k(\mathbf{o}_t)]$ and $\mathrm{Tr}(\boldsymbol{\Sigma}_k) \leq \mathrm{Tr}(\mathbf{I})$, then the expected squared transport distance satisfies $\mathbb{E}||\mathbf{A}_t - \epsilon_k||^2 \leq \mathbb{E}||\mathbf{A}_t - \epsilon||^2$ under perfect mode coupling.

In practice, exact alignment $\boldsymbol{\mu}_k = \mathbb{E}_{\mathbf{o}_t}[\mathbf{m}_k(\mathbf{o}_t)]$ is not required. Latent supervision only needs to induce a coarse partition of the action space such that samples within each partition are more transport-aligned than samples across partitions. A proof sketch for Proposition 1 is provided in Appendix K.

Importantly, the proposed priors are not used as output mixture components, but as source distributions defining distinct transport paths during training and inference. Unlike mixture-of-Gaussians conditioning, our method modifies the generative trajectory itself rather than only increasing output expressiveness.

2.3 LAFM

To ensure the prior distributions are informative and provide a structured initialization for the denoising process, we leverage a LAM, which is pre-trained to compress the dynamics between two temporal observations, $\mathbf{o}_t$ and $\mathbf{o}_{t+T}$, into a discrete latent space $\mathcal{C} = \{1, \dots, C\}$. We set the number of prior distributions K equal to the codebook size C, establishing a direct correspondence between each discrete latent action $\ell_a \in \mathcal{C}$ and a specific prior $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. The policy network predicts a categorical distribution $P_{\theta_{cls}}(k \mid \mathbf{o}_t) \equiv \pi_k(\mathbf{o}_t)$, which serves as the latent mode distribution defined in Section 2.2. The model is trained to predict the latent action index $\ell_a \coloneqq \phi_e(\mathbf{o}_t, \mathbf{o}_{t+T})$, where ϕ_e represents the pre-trained LAM encoder. The predicted distribution $P_{\theta_{cls}}(k|\mathbf{o}_t)$ is supervised using the discrete latent action ℓ_a via a categorical cross-entropy (CCE) loss:

$$\mathcal{L}_{CCE}(\theta_{cls}) = -\mathbb{E}_{(\mathbf{o}_t, \ell_a)} \left[\log P_{\theta_{cls}}(k = \ell_a|\mathbf{o}_t)\right] \tag{3}$$

where $\theta_{cls} \subset \theta$. The predicted index k is selected with an operation $\arg\max$, and the corresponding prior is used for flow matching. In this way, the network learns to bias the denoising process toward the specific motion primitive identified by the latent action. An analysis of the trained classifier is given on Appendix E.

All learned prior distributions are initialized as standard normal distributions, $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This initialization strategy is crucial for training stability; early in the training process, the model's latent action predictions are largely uninformative, and a uniform starting point prevents the flow matching objective from diverging. As training progresses, the priors naturally specialize, adapting their means and covariances to capture the specific spatial regions of their corresponding motion primitives. However, over-specialized priors could lead to overfitting, particularly for underrepresented motion primitives. To prevent this, we introduce a Kullback-Leibler (KL) divergence regularization term. This auxiliary loss ensures that the learned priors do not deviate excessively from a manageable search space, by penalizing the divergence between the selected prior distribution and a standard normal distribution. The KL divergence term is computed analytically in closed-form as:

$$\mathcal{R}_{KL}(\theta) = \mathbb{E}_{k \sim P_{\theta_{cls}}(k|\mathbf{o}_t)} \left[\frac{1}{2} \sum_{d=1}^{D} \left(\boldsymbol{\mu}_{k,d}^2 + \boldsymbol{\sigma}_{k,d}^2 - 1 - 2 \log \boldsymbol{\sigma}_{k,d} \right) \right] \tag{4}$$

The overall objective for training a model with LAFM integrates the three aforementioned components:

$$\mathcal{L}_{LAFM}(\theta) = \mathcal{L}_{AFM}^{\tau}(\theta) + \lambda_1 \mathcal{L}_{CCE}(\theta_{cls}) + \lambda_2 \mathcal{R}_{KL}(\theta) \tag{5}$$

where λ_1 and λ_2 are scalar weighting hyperparameters. The KL regularization weight λ_2 controls the degree of deviation of the learned priors from the base isotropic distribution. In particular, as $\lambda_2 \to \infty$, minimizing $\mathcal{R}_{KL}$ enforces $\boldsymbol{\mu}_k \to \mathbf{0}$ and $\boldsymbol{\Sigma}_k \to \mathbf{I}$ for all k. In this limit, the adaptive probability path becomes indistinguishable from the standard flow matching path, making the optimization virtually equivalent to solving the classical flow matching objective (Equation equation 1) alongside the latent classification task (Equation equation 3). Conversely, lowering λ_2 relaxes this constraint, granting the network the flexibility to learn highly specialized, localized prior distributions. Thus, tuning λ_2 provides a continuous interpolation between a constrained, single-prior baseline and a fully adaptive, multi-prior ensemble. A comprehensive ablation study analyzing the empirical impact of varying λ_2 is detailed in Appendix D.

2.4 Implementation details

In order to validate our method, we propose an encoder-decoder architecture to implement the flow matching policy, as depicted in Figure 2. For all of our experiments, we consider an observation $\mathbf{o}_t = [\mathbf{I}_t^1, ..., \mathbf{I}_t^n, l_t, \mathbf{q}_t]$, which incorporates one or more RGB images ($\mathbf{I}_t^1, ..., \mathbf{I}_t^n$), a description of the target task given in natural language (l_t), and the proprioceptive state of the robot ($\mathbf{q}_t$). Task instructions are embedded via a frozen text encoder, whereas images are processed by a pre-trained image encoder, which is further optimized during training. These preprocessed features, together

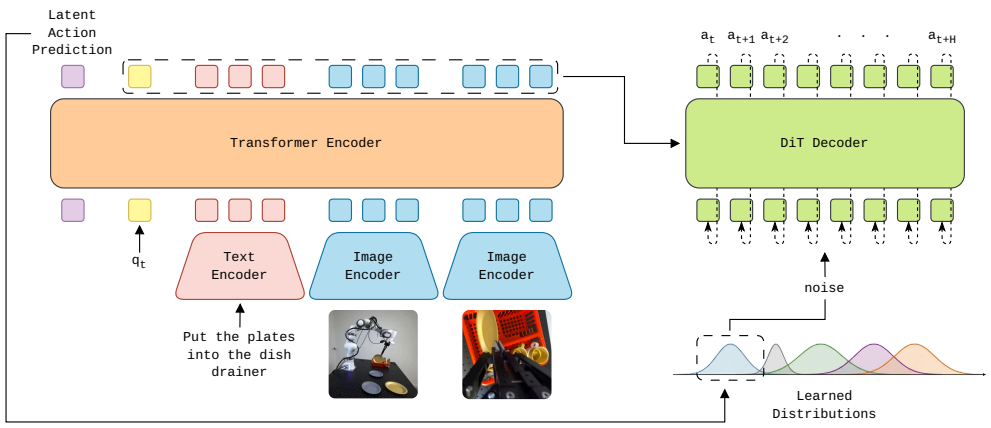


Figure 2: **Implemented architecture.** The images, task description, and robot's proprioception are given to the policy encoder. The images and text are first processed by a specialized encoder before being fed to a Transformer. The output of the Transformer encoder is given via cross-attention to the DiT decoder. The latent action predicted by the encoder is used to choose the distribution from which the noise is sampled.

with the proprioceptive state vector, are projected to a unified dimension, forming a multi-modal token sequence. An additional dedicated prompt token is added to the beginning of the sequence, which is processed by a randomly initialized Transformer (Vaswani et al., 2017) using full bidirectional attention (Devlin et al., 2019).

The hidden state corresponding to the prompt token is linearly projected to the codebook dimension C to produce classification logits (Equation equation 3). The predicted index k acts as a lookup key for the prior library, which is implemented using two embedding layers: $\mathbf{E}_{\boldsymbol{\mu}}, \mathbf{E}_{\log \boldsymbol{\sigma}} \in \mathbb{R}^{C \times D}$, where C is the number of priors and D is the dimensionality of the action space. These embeddings store the learnable parameters $\boldsymbol{\mu}_k$ and $\log \boldsymbol{\sigma}_k$ for each motion primitive. To sample the contextual noise ϵ_k, we first draw a sample from a standard normal distribution $\zeta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then apply a location-scale transformation using the retrieved embedding vectors: $\epsilon_k = \boldsymbol{\mu}_k + \boldsymbol{\sigma}_k \odot \zeta$, where $\boldsymbol{\sigma}_k = \exp(\log \boldsymbol{\sigma}_k)$ and $\odot$ denotes the element-wise product.

The decoder is parameterized as a Diffusion Transformer (DiT) (Peebles & Xie, 2023), which is conditioned on the output of the encoder through cross-attention. The input to the decoder consists of the noisy action sequence $\mathbf{A}_t^\tau$. Consequently, the input sequence comprises H tokens, which are linearly projected from the original action space to the hidden dimension of the decoder. Analogous to the encoder architecture, we do not employ attention masking. The model is optimized using the LAFM loss formulated in Equation equation 5.

LAM implementation details. A comprehensive description of the deployed LAM is provided in Appendix G, while Appendix H presents ablation studies investigating the impact of LAM training data and codebook size on policy performance.

3 Experiments

We evaluate our method in both real-world robotic manipulation settings and simulated environments. We design experiments to address the following questions:

- Does our method improve the standard flow matching policy formulation and is the improvement significant?
- Are multiple priors necessary, or is latent action prediction supervision sufficient?
- Does LAFM decrease the transport burden and consequently increases convergence speed over standard flow matching?

3.1 MAIN RESULTS

To evaluate our proposed method, we establish a direct comparison with a standard flow matching baseline (FM). To ensure a fair evaluation, the FM model shares the exact architecture and hyperparameters of LAFM, as described in Section 2.4, but omits the latent action prediction module. Instead, it relies on sampling noise from a single fixed normal distribution, being optimized via the standard conditional flow matching loss (Equation equation 1). Additionally, to contextualize our performance within the broader landscape of manipulation policies, we benchmark LAFM against a comprehensive suite of state of the art methods (Chi et al., 2023; Ghosh et al., 2024; Kim et al., 2024; Black et al., 2024; Shukor et al., 2025; Bu et al., 2025; Zhao et al., 2023; Lee et al., 2024; Mete et al., 2024).

3.1.1 REAL-WORLD EXPERIMENTS

We evaluate our policies on four distinct manipulation tasks, using 50 demonstrations per task. These demonstrations are collected with a GELLO teleoperation system (Wu et al., 2024) to control a Franka Emika Panda robotic arm. The tasks consist of: (1) picking up all **plates** from the workspace and placing them in a dish drying rack; (2) opening a **drawer**, picking up a screwdriver, stowing it inside, and closing the drawer; (3) picking up all **cans** from the table and depositing them into a trash bin; and (4) stacking all **bowls** on the table into a single pile. Figure 10 provides a visualization of each task.

Due to the prohibitive time and resource costs associated with real-world training and evaluation, we restrict our comparative analysis to two primary baselines. First, we consider ACT (Zhao et al., 2023) as a comparably sized model which is trained from randomly initialized weights. Note that while LAFM leverages a pretrained LAM for dataset labeling, the policy model is also learned from scratch. Second, we include π_0 (Black et al., 2024) to assess how our approach compares to a VLA with substantially more parameters and extensive policy pre-training. Detailed training configurations for each policy are provided in Appendix I.

We report two primary metrics for our real-world evaluations: success rate and completion score. Success rate serves as a strict binary criterion for task completion. In contrast, the completion score is a continuous metric ranging from 0 to 1 that indicates the fraction of the task successfully executed. Comprehensive task descriptions and the precise criteria used to compute completion scores are detailed in Appendix J. We conduct 15 evaluation trials per task for each model, randomizing object positions for every trial.

Table 1: **Results on real-world.** Success rates (%) and completion score (CS) (%) for various policies. VLA Pt refers to policy pretraining on robotics data.

Policy (# Params)	**VLA Pt.**	**Success Rate (%)**					**CS (%)**
		Plates	**Drawer**	**Cans**	**Bowls**	**Avg.**	**Avg.**
π_0 (3.3B)	Yes	86.7	60.0	66.7	73.3	71.7	82.9
ACT (0.05B)	No	40.0	13.3	60.0	46.7	40.0	51.7
FM (0.11B)	No	73.3	46.7	66.7	66.7	63.3	75.0
LAFM (0.11B)	No	**100.0**	**80.0**	**86.7**	**80.0**	**86.7**	**93.9**

Our method improves the average success rate by 23.4% over FM and 15.0% over π_0, with corresponding completion score gains of 18.9% and 11.0%, respectively.

3.1.2 LIBERO

To ease reproducibility and provide a rigorous comparison against diverse baselines, we evaluate our method on LIBERO (Liu et al., 2023), a simulated robotic manipulation environment comprising five benchmark suites: Goal, Object, Spatial, Long (also referred to as LIBERO-10), and LIBERO-90. While several recent works (Kim et al., 2024; Shukor et al., 2025; Bu et al., 2025) restrict their evaluations to the first four benchmarks, we advocate for a comprehensive evaluation across all five. However, to ensure a fair and direct comparison with these aforementioned works, we partition our

empirical study into two phases. First, following Shukor et al. (2025), we train a single unified model across the four combined benchmarks (Spatial, Object, Goal, and Long). Subsequently, we conduct an independent evaluation solely on LIBERO-90.

LIBERO Spatial, Object, Goal and Long. Each of the four benchmarks comprises 10 tasks. We train all models using a preprocessed version of the dataset introduced by Kim et al. (2024), which yields a refined set of 1,693 episodes encompassing all 40 tasks. For all LIBERO evaluations, we execute 50 trials per task and average the results across 3 random seeds. We report the standard deviation with Bessel's correction. Baseline metrics are sourced directly from Kim et al. (2024); Shukor et al. (2025); Bu et al. (2025), with the exception of ACT, which we independently train and evaluate.

The evaluation results are detailed in Table 2. It is important to note that Diffusion Policy (DP) (Chi et al., 2023), Octo (Ghosh et al., 2024), OpenVLA (Kim et al., 2024), and UniVLA (Bu et al., 2025) train benchmark-specific models, resulting in a distinct set of weights for each of the four suites. Consequently, the reported averages for these baselines reflect the mean performance of specialized models rather than a single, unified policy. Conversely, SmolVLA (Shukor et al., 2025) and π_0 train a single model concurrently on all four benchmarks. We adopt this unified training paradigm for all models trained in our study.

Table 2: **Results on LIBERO Spatial, Object, Goal, and Long.** Success rates (%) for various policies. VLA Pt refers to policy pretraining on robotics data. Baseline scores from Kim et al. (2024); Shukor et al. (2025); Bu et al. (2025). $\pm$ denotes standard deviation. † denotes average performance of four distinct expert policies.

Policy (# Params)	**VLA Pt.**	**Success Rate (%)**				
		Spatial	**Object**	**Goal**	**Long**	**Avg.**
DP (0.26B)	No	78.3	92.5	68.3	50.5	72.4†
Octo (0.09B)	Yes	78.9	85.7	84.6	51.1	75.1†
OpenVLA (7B)	Yes	84.7	88.4	79.2	53.7	76.5†
π_0 (3.3B)	Yes	90	86	95	73	86.0
SmolVLA (2.25B)	No	93	94	91	77	88.8
UniVLA (7B)	Yes	96.5	96.8	95.6	92.0	95.2†
ACT (0.05B)	No	96.3 ± 0.5	$\mathbf{99.5} \pm 0.1$	97.0 ± 0.7	90.8 ± 0.9	95.9 ± 0.5
FM (0.11B)	No	96.1 ± 1.0	97.9 ± 0.3	96.5 ± 0.5	91.2 ± 0.2	95.4 ± 0.3
LAFM (0.11B)	No	$\mathbf{97.2} \pm 0.7$	99.2 ± 0.0	$\mathbf{97.2} \pm 0.5$	$\mathbf{92.3} \pm 0.3$	$\mathbf{96.5} \pm 0.3$

Empirically, FM demonstrates strong baseline performance, remaining highly competitive with significantly larger models that leverage large-scale pre-training. By incorporating our method, we achieve a consistent, albeit marginal, performance gain of approximately 1% in success rate across all benchmarks, allowing our model to surpass all baselines. While this improvement is systematic across settings, the near-ceiling success rate of FM intrinsically limits the margin for further performance gains.

LIBERO-90. The LIBERO-90 suite comprises 90 distinct manipulation tasks. Applying the same data preprocessing protocol introduced by Kim et al. (2024), we filter the original dataset to obtain a final corpus of 3,959 episodes encompassing all 90 tasks. Baseline results are sourced from Mete et al. (2024), with the exception of ACT and π_0, which we train and evaluate ourselves.

This experiment more clearly highlights the efficacy of our approach, yielding an absolute gain of 10.4% in success rate over FM. Furthermore, our method systematically outperforms all considered baselines, establishing state-of-the-art results on the LIBERO-90 benchmark.

Similar gains hold on two other benchmarks that are less saturated than the standard LIBERO suites, with LAFM improving over FM by 15.1 points on LIBERO-Plus (Fei et al., 2026) and 12.3 points on RoboTwin 2.0 (Chen et al., 2026) (see Appendix A).

Table 3: **Results on LIBERO-90.** Success rates (%) for various policies. VLA Pt refers to policy pretraining on robotics data. Baseline scores from Mete et al. (2024). $\pm$ denotes standard deviation.

Policy (# Params)	VLA Pt.	Success Rate (%)
DP (0.26B)	No	75.4
VQ-BeT	No	81.3
QueST	No	88.6
π_0 (3.3B)	Yes	92.1 ± 0.1
ACT (0.05B)	No	86.2 ± 0.2
FM (0.11B)	No	82.6 ± 0.2
FM + LAC (0.11B)	No	85.4 ± 0.1
LAFM (0.11B)	No	$\mathbf{93.0} \pm 0.2$

3.2 Impact of auxiliary supervision

To isolate the efficacy of source-distribution structuring from the benefits of auxiliary representation learning, we compare our proposed LAFM against a standard flow matching policy augmented with an identical latent action prediction objective. Specifically, we evaluate a variant denoted as FM + LAC (Latent Action Conditioning). In this setup, the model continues to sample from a single standard Gaussian prior, and the predicted latent action is explicitly provided to the decoder via cross-attention.

We conduct this experiment on LIBERO-90, with results reported in Table 3. Adding latent action prediction as auxiliary supervision improves the performance of the standard flow matching policy. However, this improvement does not fully account for the gains o LAFM, highlighting the importance of structured priors in our approach. We explore the use of multiple priors with guidance other than latent actions in Appendix F.

3.3 Transport burden and convergence speed

Proposition 1 predicts a reduction of the transport cost under latent-conditioned priors. We quantify the transport burden by the kinetic energy of the generated trajectory, $\sum_\tau \|\mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t)\|^2 \delta$, which is minimized by straight, constant-speed paths. Measuring it therefore captures both shorter transport and a less entangled field in a single number. We report the mean and standard deviation over the entire held-out validation set of LIBERO-90: 136.08 ± 0.96 for flow matching against 44.97 ± 0.30 for LAFM, a reduction of 67.0%.

A reduced kinetic energy should in turn lead to faster convergence, as shorter and less entangled vector fields are easier to learn. Since the LAFM and FM training losses are not on the same scale, we compare instead the L1 action prediction error during training: at regular intervals, we run inference by integrating the learned field from the sampled prior with a single function evaluation. As shown in Figure 3, LAFM matches FM's final validation L1 after only 135k steps, a 32.5% reduction in training steps. It also converges to a lower error overall, ending 20% below FM's final L1. A qualitative analysis is presented in Appendix C.

4 Related works

4.1 Diffusion and flow matching policies

Diffusion Policy (Chi et al., 2023) introduced the concept of modeling a robot's policy as a conditional denoising diffusion process, quickly establishing itself as a standard approach for action prediction in robotic manipulation (Ghosh et al., 2024). Building upon this foundation, π_0 (Black et al., 2024) replaced the diffusion objective with a flow matching formulation—a shift adopted by subsequent research (NVIDIA et al., 2025) that has since emerged as the new state of the art. While diffusion models generate data by learning to reverse a gradual, stochastic noise-adding process step-

by-step, flow matching achieves this by directly learning a continuous vector field that transports a simple base distribution straight to a complex target distribution, allowing faster inference using fewer denoising steps. Beyond architectural differences, our work extends flow matching policies by learning distinct prior distributions, whereas these previous methods sample noise from a single, fixed distribution.

4.2 Noise optimization for flow matching policies

DSRL (Wagenmaker et al., 2025) uses Reinforcement Learning (RL) to optimize the noise space of pre-trained diffusion and flow matching policies, adapting them to downstream scenarios without altering the models' weights. Parrot (Singh et al., 2021) employs a similar strategy, but uses a policy based on normalizing flow. Similarly, we also adjust the prior noise distribution of the policy to improve its performance. But unlike these methods, which use RL to adapt trained policies, our approach operates directly during policy training, learning different noise distributions based on the bias introduced by a LAM.

4.3 Latent action models applied to robotics

Latent action models (LAM) function as unlabeled world models that learn directly from video, without requiring ground-truth action labels. Genie (Bruce et al., 2024) uses a LAM to label videos, mapping the learned latent actions to user inputs to create interactive environments. In the context of 2D video games, ILPO (Edwards et al., 2019) and LAPO (Schmidt & Jiang, 2024) learn a policy within the latent action space before mapping those latent actions to real actions. LAPA (Ye et al., 2025) utilizes latent actions to pre-train a VLA, which establishes a unified action space across different embodiments and allows the model to leverage unlabeled videos (i.e., human manipulation videos) as part of its pre-training dataset. Subsequent studies have also adopted latent actions as a labeling technique for VLA pre-training, using them either as the sole pre-training targets (Chen et al., 2025) (Bu et al., 2025) or alongside other training objectives (NVIDIA et al., 2025). Unlike prior latent action based policy learning methods, which use latent actions simply as pretraining supervision targets, our method leverages latent actions as a guiding bias to generative policy learning, using them to directly reshape the source distribution, thereby altering the geometry of the learned transport field.

5 Conclusion

In this work, we introduced LAFM, a novel generative framework designed to overcome the structural inefficiencies inherent in standard flow matching policies for robotic manipulation. By transitioning from a monolithic Gaussian prior to an adaptive library of learned distributions grounded by a LAM, our approach provides a highly informative initialization for the denoising process. This structurally informed starting point leads to more direct transport trajectories, disentangles the underlying vector fields, and naturally accommodates the heteroscedasticity of multi-modal human demonstration data.

Our extensive empirical evaluations demonstrate that LAFM yields substantial improvements over standard flow matching formulations. Consistent with our transport analysis, LAFM reduces the kinetic energy of the learned flow by 67% and reaches the final error of standard flow matching with 32.5% fewer training steps. In real-world robotic deployments, our method increased the task success rate by 23.4 percentage points, and by 10.4 points on the LIBERO-90 benchmark, with gains of 15.1 and 12.3 points on the more challenging LIBERO-Plus and RoboTwin 2.0 benchmarks.

Despite these strong results, our framework presents distinct avenues for future exploration. Currently, our reliance on video-based LAMs means that the latent representations can occasionally capture extraneous visual dynamics, such as moving backgrounds or severe camera shifts, rather than strictly isolating the robot's kinematics. Future work will focus on refining the information bottleneck to better disentangle agent-specific movements from environmental noise. Additionally, integrating our multi-prior flow matching formulation directly into the large-scale pre-training pipelines of foundational VLA architectures represents a highly promising next step. Ultimately, LAFM establishes a robust, efficient, and scalable paradigm for teaching robots complex behaviors through structured generative modeling.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164. 1, 3, 6, 8, 23, 24

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander 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 S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H 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 *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.025. 20, 21, 22

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bruce24a.html. 2, 9, 20

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to Act Anywhere with Task-centric Latent Actions. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.014. 6, 7, 9, 20, 22

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024. 22

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=itonej9GIV. 7, 15

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19752–19763, October 2025. 9

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026. 1, 6, 7, 8

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848. 22

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/. 5

Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1755–1763. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/edwards19a.html. 9

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 38574–38583, June 2026. 7, 15

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090. 6, 7, 8

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18995–19012, 2022. 22

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 22

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf. 1

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.120. 21

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin

Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ZMnD6QZAE6. 6, 7

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26991–27008. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lee24y.html. 6, 19

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t. 1, 3

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf. 6

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport, 2022. URL https://arxiv.org/abs/2209.14577. 1, 3

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7. 21, 23, 24

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 4062–4089. Curran Associates, Inc., 2024. doi: 10.52202/079017-0133. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/076c3e48fa502c660902105965fdd9f6-Paper-Conference.pdf. 6, 7, 8

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. 1, 8, 9

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad

Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. 22

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers . In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, Los Alamitos, CA, USA, October 2023. IEEE Computer Society. doi: 10.1109/ICCV51070.2023.00387. URL https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.00387. 5

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9379–9395, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/27985d21f0b751b933d675930aa25022-Paper-Conference.pdf. 9

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844. 6, 7

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ysuv-WOFeKR. 9

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings*

of Machine Learning Research, pp. 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html. 1

Mohammad Hassan Vali and Tom Bäckström. Nsvq: Noise substitution in vector quantization for machine learning. *IEEE Access*, 10:13598–13610, 2022. doi: 10.1109/ACCESS.2022.3147670. 20

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. 5

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=jU7AbGq3se. 9

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2526–2547, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.127. URL https://aclanthology.org/2025.acl-long.127/. 22

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12156–12163, 2024. doi: 10.1109/IROS58592.2024.10801581. 6

Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, and Hongkai Xiong. Spatial-temporal transformer networks for traffic flow forecasting, 2021. URL https://arxiv.org/abs/2001.02908. 20

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VYOe2eBQeh. 9

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016. 6, 23

A ADDITIONAL BENCHMARKS

On the four standard LIBERO suites, the FM baseline already reaches an average success rate of 95.4% (Table 2), which leaves little room to measure the benefit of our method. To further study how LAFM improves over the standard flow matching formulation, we therefore evaluate both policies on two less saturated benchmarks: LIBERO-Plus (Fei et al., 2026) and RoboTwin 2.0 (Chen et al., 2026). As in the main experiments, FM shares the exact architecture and hyperparameters of LAFM, but omits the latent action prediction module and samples noise from a single standard normal distribution.

LIBERO-Plus. LIBERO-Plus is a robustness benchmark built on four of the LIBERO suites (Spatial, Object, Goal, and Long). It applies controlled perturbations along seven dimensions: object layout, camera viewpoints, robot initial states, language instructions, lighting conditions, background textures, and sensor noise, resulting in a total of 10,030 perturbed task instances.

RoboTwin 2.0. RoboTwin 2.0 is a simulated benchmark of 50 dual-arm manipulation tasks, which applies strong domain randomization over scene clutter, lighting conditions, background textures, tabletop height, and language instructions. It complements LIBERO with bimanual manipulation and a substantially different task distribution.

We follow the standard evaluation protocol of each benchmark. On LIBERO-Plus, we evaluate on the full perturbation set, which amounts to approximately 250 evaluations per task. On RoboTwin 2.0, we perform 100 trials per task. We report the mean success rate and its standard deviation over 3 random seeds.

Table 4: **FM and LAFM across simulated benchmarks.** Success rates (%). Benchmarks are sorted by decreasing FM success rate. LIBERO results are reproduced from Tables 2 and 3, where the first row is the average over the Spatial, Object, Goal, and Long suites. $\pm$ denotes standard deviation. Δ denotes the absolute gain of LAFM over FM.

Benchmark	**FM**	**LAFM**	**Δ**
LIBERO (Spatial, Object, Goal, Long)	95.4 ± 0.3	$\mathbf{96.5} \pm 0.3$	$+1.1$
LIBERO-90	82.6 ± 0.2	$\mathbf{93.0} \pm 0.2$	$+10.4$
RoboTwin 2.0	53.7 ± 0.3	$\mathbf{66.0} \pm 0.2$	$+12.3$
LIBERO-Plus	50.4 ± 0.2	$\mathbf{65.5} \pm 0.1$	$+15.1$

On LIBERO-Plus, LAFM achieves a success rate of $65.5 \pm 0.1\%$, against $50.4 \pm 0.2\%$ for FM, an absolute gain of 15.1 points. Per category, LAFM is particularly robust to perturbations of lighting, background textures, camera viewpoints, and sensor noise, with success rates around 90%. It obtains intermediate performance under object layout perturbations (57%) and degrades under perturbations of the robot initial state and of the language instructions (around 28%). Regarding these two weak categories, robot initial-state perturbations are among the most challenging for all models evaluated by Fei et al. (2026): the strongest of them in this category reaches only 30.3%. Language perturbations, in turn, probe language grounding. Our architecture uses a compact frozen text encoder rather than a VLM backbone, so its generalization to unseen instruction phrasings is limited by design. This limitation is orthogonal to our contribution, which concerns the generative process. On RoboTwin 2.0, LAFM achieves an average success rate of $66.0 \pm 0.2\%$, against $53.7 \pm 0.3\%$ for FM, an absolute gain of 12.3 points.

Table 4 summarizes the gains of LAFM over FM across all simulated benchmarks. The gains on LIBERO-Plus and RoboTwin 2.0 are more than ten times larger than the 1.1-point improvement on the near-saturated LIBERO Spatial, Object, Goal, and Long suites, and consistent with the 10.4-point gain on LIBERO-90. More generally, the lower the success rate of the FM baseline, the larger the improvement brought by LAFM. This supports our analysis in Section 3.1.2 that the ceiling effect of the standard suites masks the margin of improvement.

B CONVERGENCE ANALYSIS FIGURE

We provide Figure 3, which illustrates the faster convergence of LAFM compared to the standard flow matching formulation.

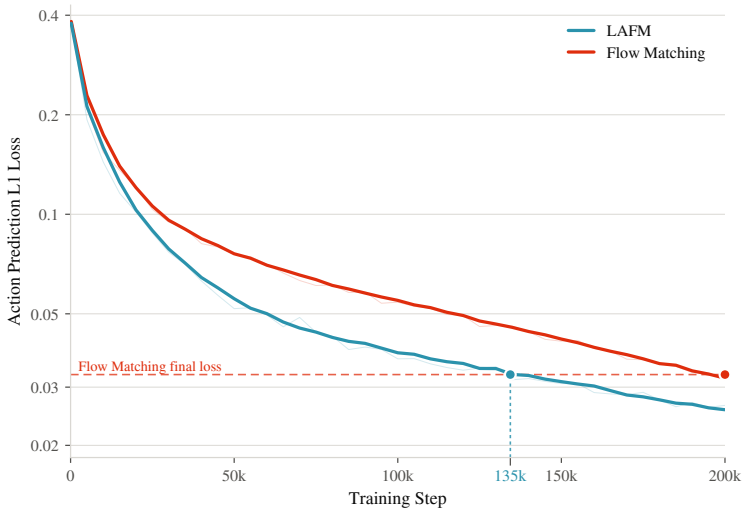


Figure 3: **Convergence speed.** We compare the action prediction L1 loss of LAFM and FM on LIBERO-90 during training.

C QUALITATIVE ANALYSIS

C.1 LEARNED PRIORS ANALYSIS

Figure 4 shows a t-SNE visualization of the learned prior distributions for our model trained on LIBERO. We display the projected means, representing the variance by color and size of the circles. To better understand the correspondence of the learned distributions and the latent actions, we take a single image as input and generate an output image using each possible latent action and feeding them to the LAM's decoder. We then compute the optical flow between the generated images and the original image. We display the original image with a superposed optical flow representation for a couple of latent actions that have similar learned priors.

We can observe that our model learns similar distributions for latent actions that represent similar movements. Another interesting thing to note is that latent actions which represent larger range movements have corresponding priors with larger variances, while smaller range movements have more specialized priors, with smaller variance. This reinforces our intuition that latent actions are suitable to capture the heteroscedasticity of human demonstrations, making them appropriate for guiding our priors.

C.2 VECTOR FIELDS VISUALIZATION

To have an intuition of how our method simplifies flow matching, we perform the inference for our model and FM for the same random batch of 512 items from LIBERO-90. We use $\delta = 0.1$ and store the intermediate predictions for each denoising step. We use PCA to reduce the dimension of the action space, computing it based on the target actions. We then project the original noise points and the model's predictions into the first and second principal components, interpolating the 9 intermediate predictions to represent the denoising vector field. To allow for a visualization of the transport process, we shift the mean of the prior distributions by adding an offset to all the noise points, moving them to the left. For the outputs of the model, the offset is proportional to $1 - \tau$, being zero for the final prediction.

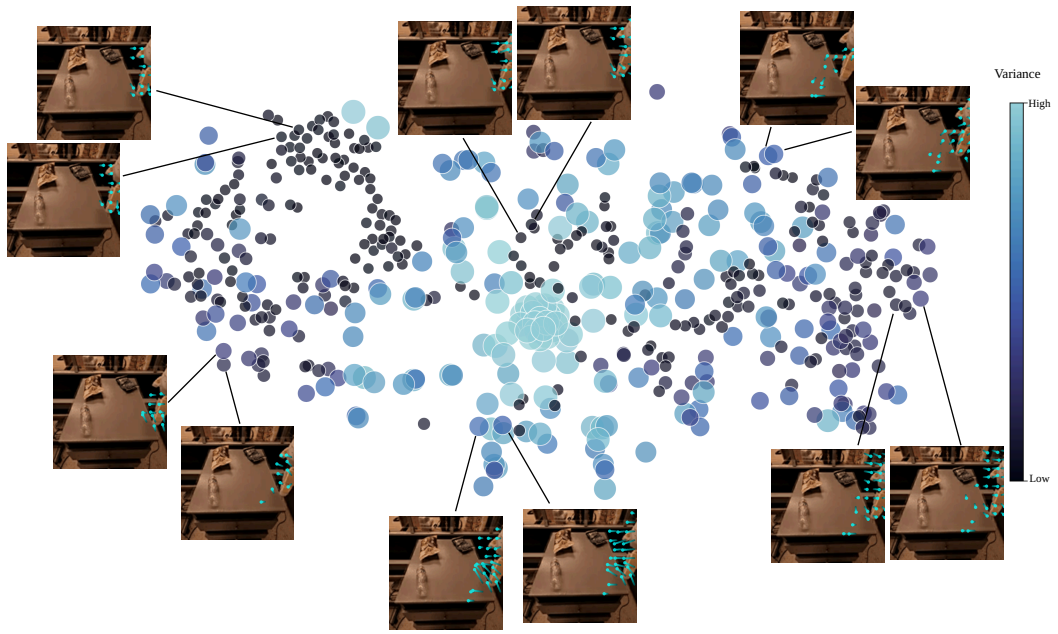


Figure 4: **Learned prior distributions visualization.** We plot the t-SNE of the learned means. The learned variances are represented by the color and size of the circles. We display the optical flow generated by the LAM decoder while reconstructing the same image using different latent actions.

The resulting graphs are shown in Figure 5, which empirically illustrates Proposition 1: latent-conditioned priors reduce the initial displacement between source and target samples, producing shorter and less entangled transport trajectories, considerably reducing the transport cost and the final prediction error.

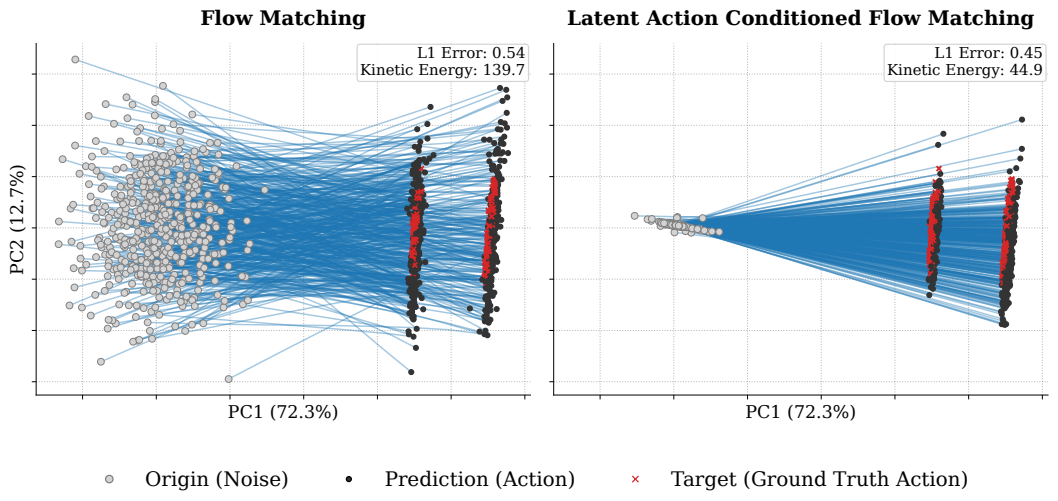


Figure 5: **Vector fields visualization.** We compare the denoising process of the standard and latent action guided flow matching policies. The gray dots are the sampled noise points, the black dots are the final predicted actions, and the blue lines connecting the dots represents the flow matching vector fields. The target actions are represented by the red crosses, being the same on both plots. We present the average L1 distance of the final predicted action to the target action and the average transport cost for the batch. The explained variance ratio is given for both principal components.

D λ_2 SWEEP

To verify our hypothesis that λ_2 interpolates between a single-prior baseline with additional latent action supervision and a multi-prior adaptive ensemble, we sweep its value over 16 orders of magnitude on LIBERO-90. Given the scale of the experiment, we limit our evaluations to a single seed per run, otherwise retaining the protocol used in the other experiments. The results are shown in Figure 6.

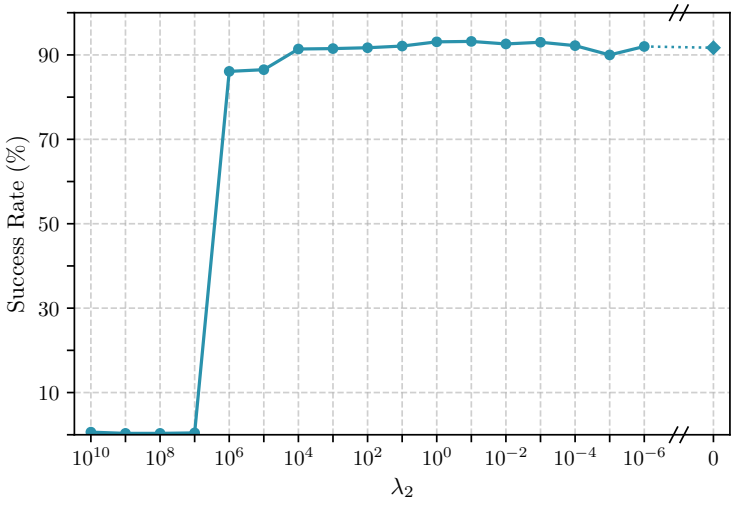


Figure 6: **λ_2 ablation.** We sweep λ_2 and evaluate the resulting policy performance on LIBERO-90.

Three regimes emerge. (1) For $\lambda_2 > 10^7$, training collapses. We note that this does not contradict the limiting argument of Section 2.3, which describes the loss landscape as $\lambda_2 \to \infty$. The collapse is instead an optimization artifact: the KL term dominates all gradients and, because it is an expectation over $k \sim P(k|\mathbf{o}_t)$, its gradients also flow through the classifier and shared encoder, preventing any feature learning. (2) For $\lambda_2 \in [10^5, 10^6]$, the priors are effectively pinned near $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and performance (86.1–86.5%) closely matches the FM + LAC ablation (85.4%), empirically confirming the equivalence predicted in Section 2.3. (3) For $\lambda_2 \leq 10^4$, performance is remarkably robust across 10 orders of magnitude (90.0–93.2%), peaking for $\lambda_2 \in [10^{-3}, 10^{-1}]$. Notably, even $\lambda_2 = 0$ achieves a success rate of 91.7%, demonstrating that the structured-prior mechanism does not depend on the regularizer, which rather contributes a modest refinement by preventing over-specialization of rarely selected priors.

E CLASSIFIER ANALYSIS

In order to validate the proper functioning of the latent-action selection mechanism, we report a few relevant metrics, computed on LIBERO-90.

Prediction accuracy. The policy encoder achieves 73.1% top-1 and 97.4% top-5 accuracy in predicting the LAM-derived latent actions. We hypothesize that most errors correspond to codes representing similar movements. To verify this, we compute, over misclassified samples, the KL divergence from the ground-truth code's prior to the predicted code's prior: it is on average 9.3 times smaller than the mean pairwise inter-prior divergence (ratio 0.107). Since KL is non-negative, at least $\sim 89\%$ of misclassifications involve priors closer than an average pair, hence errors predominantly select priors close to the correct one. This is consistent with the qualitative analysis in Appendix C.1, where codes representing similar movements exhibit similar learned priors.

Evenness. To quantify whether a small subset of codes dominates, we report evenness: the entropy of the marginal code distribution normalized by its maximum, where 1 indicates perfectly uniform usage and 0 indicates collapse onto a single code. The ground-truth LAM labels have an evenness

of 0.753 and the policy's predictions 0.731, indicating that usage is spread over a large fraction of the codebook rather than concentrated in a few dominant codes. Moreover, the similar evenness of predictions and labels indicates that the classifier reproduces the LAM's usage distribution.

Entropy. While the marginal distribution is broad, the per-observation predictive distribution is sharp: the mean predictive entropy is 1.06 nats (against a maximum of $\log 512$ = 6.24), i.e., conditioned on an observation, the classifier hesitates among only $e^{1.06} \approx 2.9$ candidate codes on average. Moreover, the gap between the marginal entropy of the code distribution (4.70 nats) and the classifier's cross-entropy (0.81 nats) yields a variational lower bound of 3.89 nats on the mutual information between the observation and the upcoming latent action, meaning that the current observation resolves $\sim 83\%$ of the code uncertainty. Finally, the cross-entropy (0.81 nats) lies slightly below the mean predictive entropy (1.06 nats). Since the two coincide under perfect calibration, the classifier operates in a mildly underconfident regime.

Codebook usage. Regarding raw codebok usage, since the LAM is trained on different data, not all codes are activated on the target dataset. The LAM assigns 91.4% of codes at least once, which is the upper-bound achievable policy usage. The policy predicts 84.4% of codes, i.e., 92% of the reachable codes.

Performance under oracle latent actions. To verify wether mispredictions by the classifier jeopardize the method, we aim to evaluate the performance of the policy under oracle latent actions. Nevertheless, a closed-loop oracle success rate is not directly measurable: the oracle latent is produced by the LAM from the frame pair, and only exists after the policy has acted, using the very action the oracle latent is meant to condition. We therefore evaluate the open-loop counterpart on a held-out validation set of LIBERO-90, where future frames are available from the demonstrations. We compare the mean squared error of generated actions when the prior is selected by the oracle latent computed by the LAM from the true future frame, and the policy's predicted latent. The oracle achieves 0.00735 and the predicted latent 0.00719. Predicted latents thus marginally surpass oracle selection. The small inversion is explained by the training setup: since we do not use teacher forcing, the prior used during training is selected by the policy's own prediction, so predicted latents are on-distribution for the decoder at inference, while oracle latents are mildly off-distribution. Latent misprediction therefore costs essentially nothing, and there is no unexploited oracle headroom.

Together, these results indicate that the latent-mode selection mechanism operates over a diverse, well-utilized prior library, accurately predicting the correct, or a similar, latent mode, and not being a bottleneck for model's performance.

F MOTION PRIMITIVE SOURCE ABLATION

To verify the importance of latent actions to our method, we conduct a study using different sources of motion primitives. The goal is to assess whether latent action guidance is necessary, as opposed to guidance derived directly from actions.

We consider three action-derived partition variants: (a) K-means over flattened action chunks, (b) K-means over cumulative chunk displacement, and (c) a VQ-VAE over action chunks, analogous to the one used in VQ-BeT (Lee et al., 2024). In all cases we keep the architecture, training, and evaluation configuration identical, including $K = 512$ and the loss of Equation equation 5. The only change is to the CCE supervision targets: the policy encoder now predicts the action-derived cluster index from $\mathbf{o}_t$, exactly as it predicts the latent action in LAFM. Figure 7 reports the success rates of the resulting policies on LIBERO-90.

Two conclusions follow. First, even naive action-space partitions improve over the single-prior baseline ($+6.9$ points for flattened K-means), which confirms the generality of our framework: consistent with Proposition 1, any reasonably transport-aligned partition reduces the transport cost. Second, the LAM-derived partition remains clearly the strongest ($+3.5$ points over the best action-derived variant). We attribute this to observation predictability: at inference time the mode must be predicted from $\mathbf{o}_t$ alone, and latent actions are grounded in observed visual dynamics by construction, whereas action-space clusters may group kinematically similar chunks arising from visually distinct contexts (and vice versa), which increases classification ambiguity.

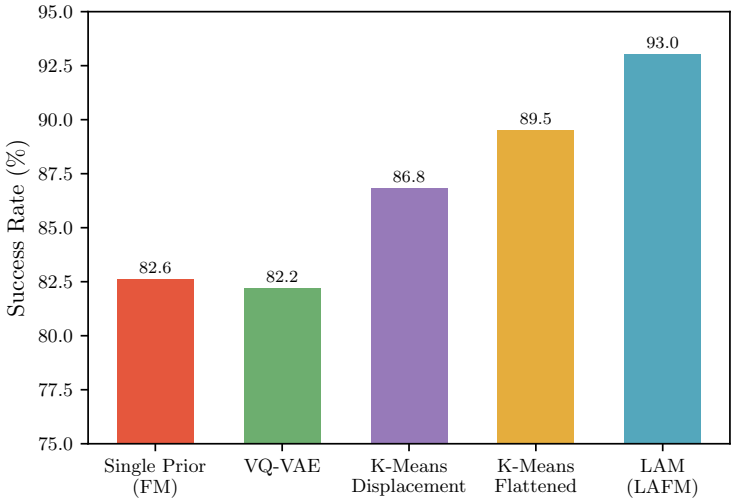


Figure 7: **Motion primitive source ablation.** We ablate different motion primitives label sources, and report the resulting success rate on LIBERO-90.

The VQ-VAE variant provides essentially no benefit over FM. We attribute this to a mismatch between the partition geometry and the prior family: K-means minimizes the within-cluster squared distance in action space, which is precisely the transport quantity of Proposition 1, and it operates in the same space in which our Gaussian priors are defined. VQ-VAE codes, in contrast, are optimized for reconstruction in a learned latent space, so the action chunks assigned to a given code need not form a compact region of action space, yielding Gaussian priors whose means and variances poorly summarize the corresponding action modes.

G Latent action model

The latent action model (LAM) architecture pairs an inverse dynamics encoder—which compresses two or more consecutive frames into a latent action—with a forward dynamics decoder that uses this latent action alongside historical frames to predict future frames, as shown in Figure 1 (left). Crucially, to prevent the encoder from simply memorizing the target frames rather than extracting meaningful actions, an information bottleneck must be applied. This is typically enforced by discretizing the latent action space or by regularizing a continuous latent action space. Although the ideal aim of a latent action is to isolate and represent the behavior of specific agents, in practice, these models capture all visual dynamics within a scene, including extraneous factors like camera motion or background movement. Nonetheless, because the video dynamics in robotic manipulation datasets arise almost exclusively from the robot's movements, this is not necessarily a limitation for our use case.

Following Bruce et al. (2024), we implement a Spatial-Temporal Transformer (Xu et al., 2021) as our LAM encoder. The LAM decoder is implemented as a traditional Transformer. Since the decoder processes a single input frame, temporal attention layers are not required (Bu et al., 2025). To enforce the information bottleneck, the latent action space is discretized using NSVQ (Vali & Bäckström, 2022). All LAMs are trained with the configuration given in Table 5. Each training takes around 40 hours and is performed using a single NVIDIA H100 GPU. For all our experiments, with the exception of the ablations detailed in Appendix H, we use the same LAM, which is trained in the Fractal (Brohan et al., 2023) dataset, using a codebook size of 512.

To ensure that latent actions capture coarse movements rather than low-level actions, we train the model using frame pairs sampled one second apart. For datasets with multiple cameras, a single camera view $\mathbf{I}^j$ is chosen and all latent actions are predicted using this view: $\ell_a \coloneqq \phi_e(\mathbf{I}_t^j, \mathbf{I}_{t+T}^j)$ where T corresponds to the frequency of the dataset. Since all datasets used in our experiments are recorded at frequencies higher than 1 Hz, a single latent action effectively corresponds to a chunk of real actions. After training, the model is used to automatically annotate a target dataset with latent

Table 5: LAM hyperparameters.

Model Dimension	512
Feed-Forward Dimension	2048
Encoder Layers	8
Decoder Layers	8
Attention Heads	8
Dropout	0.1
Image Size	128
Patch Size	16
Learning Rate	$5 \cdot 10^{-5}$
Batch Size	128
Weight Decay	10^{-4}
Gradient Steps	300,000
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning Rate Scheduler	Cosine Decay
Warm-up Steps	-

actions. LIBERO and the real-world dataset alike have two camera views, one external and another attached to the end-effector of the robot. For both cases, we extract the latent actions using the external view.

H LATENT ACTION MODEL ABLATIONS

In this section, we analyze how different design choices in the LAM affect the performance of the final policy. All ablations are conducted on LIBERO-90. We keep the policy architecture and training configuration fixed across experiments, and only vary the LAM used to label the dataset.

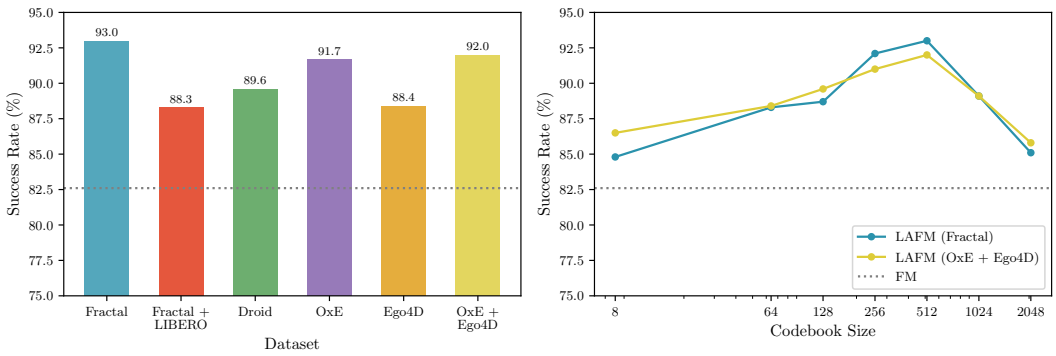


Figure 8: **Impact of different LAM design choices on the final policy performance.** We ablate the dataset (left) and the codebook size (right) used during LAM training, and and report the resulting success rate on LIBERO-90.

H.1 TRAINING DATASET

We investigate how the choice of dataset used to train the LAM influences downstream performance. Our hypothesis is that, as long as the dataset contains sufficient motion diversity to capture the relevant motion primitives, its exact distribution should have limited impact.

To test this, we first consider single-source manipulation datasets: Fractal (Brohan et al., 2023), which differs in embodiment from the target setting, and Droid (Khazatsky et al., 2024), which

shares the same embodiment but differs in visual distribution. LIBERO alone lacks sufficient diversity to train a meaningful LAM, the decoder can memorize future frames from the visual input alone, making the latent actions learned uninformative. We instead approximate an in-distribution setting by training on a mixture of Fractal and LIBERO.

We further evaluate a subset of OpenX Embodiment (OxE) (O'Neill et al., 2024), using the sampling distribution proposed by Bu et al. (2025), as well as Ego4D (Grauman et al., 2022), a large-scale human manipulation video dataset. Finally, we consider a combination of OxE and Ego4D, where Ego4D constitutes 3.0% of the mixture following Bu et al. (2025).

The results in Figure 8 show that adding Ego4D to the OxE mixture results in a small performance improvement. Nevertheless, using Fractal by itself outperforms both large dataset combinations. This suggests that quality is more important than quantity. Having a consistent dataset that covers the motion priors of interest seems to be more important than having a larger but less consistent data distribution. We posit that the lower performance when using Droid is due to the large number of different camera views, which makes it more challenging to extract consistent motion primitives.

The lower performance on Ego4D is unsurprising, as it exhibits the largest distribution gap from our target datasets. Much of the apparent dynamics in Ego4D videos arises from camera motion, leading to learned priors that are not useful for our policy. Additionally, the inherent shakiness of egocentric footage introduces substantial visual noise, which complicates latent action learning. Finally, mixing Fractal with LIBERO results in the worst performance. Given the high predictability of the LIBERO data, even though mixing it with Fractal allows for stable LAM training, the resulting learned motion priors are worse, given that the model is able to predict the future frame regardless of the latent action for a part of its training data.

H.2 Codebook size

An important design choice is the size of the LAM codebook, as it directly determines the number of priors used in our method. Ideally, this size should match the number of motion primitives present in the target dataset. To study this effect, we train all LAMs on the Fractal dataset (Brohan et al., 2023) and on the OxE + Ego4D mixture (O'Neill et al., 2024; Grauman et al., 2022; Bu et al., 2025), using the standard configuration described in Appendix G, while varying the codebook size.

Figure 8 shows the performance of the resulting policies. We observe that both excessively small and excessively large codebooks degrade performance, suggesting that under- and over-partitioning the latent action space are both harmful. Intuitively, these results are coherent with what we would expect. Not having enough priors to cover all motion primitives in the target dataset does not exploit the full potential of our method, while having too many priors leads to redundant distributions. Nevertheless, our method consistently outperforms standard flow matching across all prior counts, having the optimal codebook size consistent for both datasets analyzed.

I Policy training details and hyperparameters

We give the implementation details for all our policy models and baselines. All models are trained in a single NVIDIA H100 GPU. Batch size is chosen to maximize GPU memory usage and the learning rate is adapted accordingly. Each training takes around 20 hours. All policies are trained and evaluated using the LeRobot library (Cadene et al., 2024).

I.1 LAFM and FM

For all experiments, we use the same configuration for LAFM and FM. We use ModernBERT (Warner et al., 2025) as frozen text encoder. To reduce the computational overhead, we implement a caching system, saving the embeddings of each task and re-using them when the tasks reappear. The image encoder is a ResNet18 (He et al., 2016) pretrained on ImageNet-1k (Deng et al., 2009). We use a chunk size (H) of 10 actions for LIBERO and 20 actions for the real world experiments. We execute all 10 actions of the chunk when evaluating in LIBERO and execute 15 out of the 20 on real-world experiments. During inference, we use $\delta = 0.1$, performing 10 denoising steps for all experiments. Nevertheless, both LAFM and FM are quite robust to this hyperparam-

eter, being able to maintain equivalent performance even with a single denoising step, as shown in Figure 9. The other model and training hyperparameters are given in Table 6.

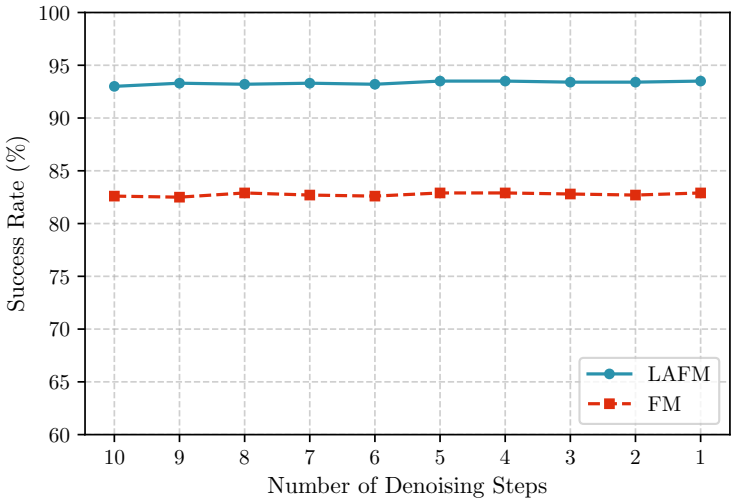


Figure 9: **Ablation on the number of denoising steps.** We plot the success rate on LIBERO-90 for LAFM and FM policies using different numbers of denoising steps during inference.

Table 6: LAFM and FM model and training hyperparameters.

Model Dimension	512
Feed-Forward Dimension	3200
Encoder Layers	8
DiT Layers	8
Attention Heads	8
Dropout	0.1
Learning Rate	$8 \cdot 10^{-5}$
Batch Size	448
Weight Decay	10^{-2}
Gradient Steps	200,000
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning Rate Scheduler	Cosine Decay
Warm-up Steps	10,000

Flow matching step sampling. During training, we follow Black et al. (2024) and sample the flow matching step τ from a scaled Beta distribution that prioritizes noisier samples and adds a cutoff value of 0.999. Specifically:

$$u \sim \text{Beta}(1, 1.5), \quad \tau = 0.999\, u, \quad \tau \in [0, 0.999]$$

LAFM loss scalars. We set both λ_1 and λ_2 from Equation equation 5 to 10^{-3}, which empirically seems to work the best in our experiments.

I.2 ACT

For ACT, we use the same configuration as in the original implementation (Zhao et al., 2023). We use the same training and inference chunk sizes as for LAFM for all experiments. Table 7 lists the remaining hyperparameters used.

Table 7: ACT model and training hyperparameters.

Model Dimension	512
Feed-Forward Dimension	3200
VAE Encoder Layers	4
Policy Encoder Layers	4
Policy Decoder Layers	1
Attention Heads	8
Dropout	0.1
Learning Rate	$8 \cdot 10^{-5}$
Batch Size	512
Weight Decay	10^{-2}
Gradient Steps	200,000
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning Rate Scheduler	-
Warm-up Steps	-

I.3 π_0

For π_0, we follow the exact same configuration as the original implementation (Black et al., 2024), initializing the model from the released pre-trained weights. The original chunk size of 50 is kept during training, but we execute the same number of actions as for the LAFM experiments during inference. Training hyperparameters are shown in Table 8.

Table 8: π_0 training hyperparameters.

Learning Rate	$5 \cdot 10^{-5}$
Batch Size	8
Gradient Steps	200,000
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning Rate Scheduler	Cosine Decay
Warm-up Steps	1,000

Given the large scale of π_0, the maximum batch size allowed in a single GPU is much smaller than for the other polices studied. Nevertheless, we opt for a compute equivalent approach between all models, training them for the same number of gradient steps, which results in approximately the same number of GPU-hours used per training. Note that π_0 is being fine-tuned, while the other policies are being trained from scratch.

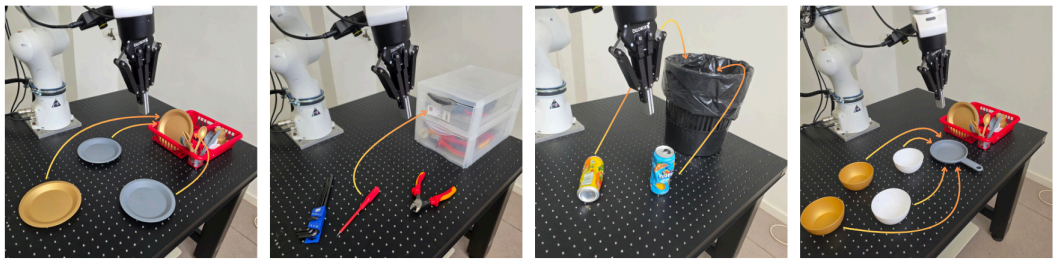
J Real-world experiments details

Figure 10 illustrates the four real-world tasks proposed for our experiments.

J.1 Completion Score (CS) criteria

For each task, we define a set of landmarks. The completion score is computed by adding the number of landmarks achieved during the episode and dividing it by the total number of landmarks in that same episode.

Plates. One to four plates are spread across the table. The position of each plate is chosen randomly. The dish dryer rack position does not change. The landmarks used to compute the completion score are: picking up a plate; placing a plate in the dish dryer rack. The score is therefore



(1) Plates **(2) Drawer** **(3) Cans** **(4) Bowls**

Figure 10: Real-world tasks examples.

relative to the number of plates in a given scene. In order to compute a fair and comparable score across all models, we perform the same number of evaluation episodes per plate count with each model (e.g. five episodes with four plates, five episodes with three plates, and five episodes with two plates). We do the same for the cans and bowls task, which also have a variable number of possible objects in the scene.

Drawer. The screwdriver is placed in a random position on the table. The drawer cabinet position is kept fixed. The completion score landmarks are: opening the drawer; picking up the screwdriver; placing the screwdriver inside the drawer; closing the drawer.

Cans. One to two cans are placed in a random position on the table. The trash can position is kept fixed. The completion score landmarks are: picking up a can; throwing a picked can in the trash.

Bowls. One to four bowls are placed randomly on the table. The position of the pan in which the bowls should be stacked is kept fixed. The completion score landmarks are: picking up a bowl; placing a bowl on top of the pan, for the first bowl, or placing a bowl on top of the bowl pile, for the others.

K Proof sketch for Proposition 1

We compare the expected squared transport distance under two source distributions: (i) a single isotropic Gaussian prior, and (ii) a latent-conditioned prior aligned with action modes.

Setup. Let the conditional action distribution decompose as

$$p(\mathbf{A}_t|\mathbf{o}_t) = \sum_{k=1}^{K} \pi_k(\mathbf{o}_t)\, p_k(\mathbf{A}_t|\mathbf{o}_t),$$

where each component $p_k(\mathbf{A}_t|\mathbf{o}_t)$ has mean $\mathbf{m}_k(\mathbf{o}_t)$ and covariance $\mathbf{S}_k$.

We analyze the expected squared distance $\mathbb{E}\|\mathbf{A}_t - \epsilon\|^2$ under the two constructions.

Case 1: Single isotropic prior. Let $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon\|^2 = \mathbb{E}\|\mathbf{A}_t\|^2 + \mathbb{E}\|\epsilon\|^2 - 2\,\mathbb{E}[\mathbf{A}_t^\top \epsilon].$$

Since ϵ is independent of $\mathbf{A}_t$ and has zero mean, the cross-term vanishes:

$$\mathbb{E}[\mathbf{A}_t^\top \epsilon] = 0.$$

Thus:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon\|^2 = \mathbb{E}\|\mathbf{A}_t\|^2 + \mathrm{Tr}(\mathbf{I}).$$

Using the mixture decomposition:

$$\mathbb{E}\|\mathbf{A}_t\|^2 = \sum_k \pi_k(\mathbf{o}_t)\left(\|\mathbf{m}_k(\mathbf{o}_t)\|^2 + \mathrm{Tr}(\mathbf{S}_k)\right).$$

Case 2: Latent-conditioned prior. Let $\epsilon_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ with $k \sim \pi(\mathbf{o}_t)$. Assume perfect mode coupling, i.e., the same latent index k is used for both $\mathbf{A}_t$ and ϵ_k. Then:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon_k\|^2 = \sum_k \pi_k(\mathbf{o}_t)\, \mathbb{E}_{\epsilon_k, \mathbf{A}_t \sim p_k} \|\mathbf{A}_t - \epsilon_k\|^2.$$

Expanding the squared norm:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon_k\|^2 = \mathbb{E}\|\mathbf{A}_t - \boldsymbol{\mu}_k\|^2 + \mathbb{E}\|\epsilon_k - \boldsymbol{\mu}_k\|^2 - 2\,\mathbb{E}[(\mathbf{A}_t - \boldsymbol{\mu}_k)^\top (\epsilon_k - \boldsymbol{\mu}_k)].$$

Again, since ϵ_k is independent of $\mathbf{A}_t$ and centered at $\boldsymbol{\mu}_k$, the cross-term vanishes, yielding:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon_k\|^2 = \mathbb{E}\|\mathbf{A}_t - \boldsymbol{\mu}_k\|^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_k).$$

We further decompose:

$$\mathbb{E}\|\mathbf{A}_t - \boldsymbol{\mu}_k\|^2 = \|\mathbf{m}_k(\mathbf{o}_t) - \boldsymbol{\mu}_k\|^2 + \mathrm{Tr}(\mathbf{S}_k).$$

Thus:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon_k\|^2 = \sum_k \pi_k(\mathbf{o}_t) \left(\|\mathbf{m}_k(\mathbf{o}_t) - \boldsymbol{\mu}_k\|^2 + \mathrm{Tr}(\mathbf{S}_k) + \mathrm{Tr}(\boldsymbol{\Sigma}_k) \right).$$

Comparison. For the isotropic prior:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon\|^2 = \sum_k \pi_k(\mathbf{o}_t) \left(\|\mathbf{m}_k(\mathbf{o}_t)\|^2 + \mathrm{Tr}(\mathbf{S}_k) \right) + \mathrm{Tr}(\mathbf{I}).$$

For the latent-conditioned prior:

$$\mathbb{E}\|\mathbf{A}_t - \epsilon_k\|^2 = \sum_k \pi_k(\mathbf{o}_t) \left(\|\mathbf{m}_k(\mathbf{o}_t) - \boldsymbol{\mu}_k\|^2 + \mathrm{Tr}(\mathbf{S}_k) + \mathrm{Tr}(\boldsymbol{\Sigma}_k) \right).$$

If $\boldsymbol{\mu}_k \approx \mathbb{E}_{\mathbf{o}_t}[\mathbf{m}_k(\mathbf{o}_t)]$ and $\mathrm{Tr}(\boldsymbol{\Sigma}_k) \leq \mathrm{Tr}(\mathbf{I})$, then

$$\mathbb{E}\|\mathbf{A}_t - \epsilon_k\|^2 \;\leq\; \mathbb{E}\|\mathbf{A}_t - \epsilon\|^2,$$

with strict inequality whenever the mode means are non-zero.

Interpretation. The isotropic prior requires transporting samples from a single global reference point to all action modes, while the latent-conditioned prior initializes samples closer to their corresponding modes. This reduces both the average displacement and the complexity of the learned transport field. This analysis focuses on squared Euclidean transport and provides an intuitive proxy for transport complexity; empirical results (Section 3.3 and Appendix C.2) corroborate that similar reductions occur in learned flow trajectories.

Remark on covariance assumption. If the action space is normalized to $[-1, 1]^d$, then $\mathrm{Tr}(\boldsymbol{\Sigma}_k) \leq d = \mathrm{Tr}(\mathbf{I})$ follows naturally, so the condition on the covariance is mild in practice.

Remark on perfect mode coupling assumption. The result assumes perfect mode coupling, i.e., that the latent index k used to sample ϵ_k matches the component generating $\mathbf{A}_t$. In practice, this condition may not hold. However, as long as the correct mode is selected with high probability, the expected transport cost remains reduced, since most samples are initialized near their corresponding modes.

BROADER IMPACTS STATEMENT

Although the advancement of capable robotic manipulation through methods such as LAFM can enhance productivity and assist in hazardous tasks, it also carries the risk of labor displacement in sectors reliant on physical work. Ensuring a net positive impact requires a proactive focus on workforce retraining, education, and ensuring equitable access to these technologies to prevent exacerbating already excessive socioeconomic inequality.

REPRODUCIBILITY STATEMENT

We describe the LAFM formulation and training objective in Sections 2.2 and 2.3, and the policy architecture in Section 2.4. Appendix I lists all training and inference hyperparameters for all trained models (Tables 6, 7, and 8), together with the flow matching step sampling, the loss weights, the hardware, and the software library used. Appendix G gives the architecture, training data, and training configuration of the latent action model (Table 5). All simulated experiments use publicly available benchmarks. Section 3.1.2 describes the LIBERO data preprocessing and evaluation protocol, including the number of trials per task, the number of random seeds, and how standard deviations are computed; Appendix A does the same for LIBERO-Plus and RoboTwin 2.0. For the real-world experiments, Section 3 describes the robot setup, data collection, and evaluation protocol, and Appendix J details each task and the criteria used to compute the completion score. To facilitate reproduction during review, we provide an anonymized version of our code as supplementary material, and the full code and model weights will be publicly released upon acceptance.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing the paper. Specifically, we used a large language model (LLM) to draft parts of the text from notes and results written by the authors, to edit the manuscript for clarity, concision, and grammar, to shorten it to fit the page limit, and to retrieve and check descriptive and bibliographic details of the benchmarks and prior works we cite. The authors reviewed and approved all AI-assisted text, and verified every reported number, citation, and factual claim against our experiments and the original sources. We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate mathematical claims or contribute to their proofs, propose or refine hypotheses, design or provide feedback on the research methodology or experiments, implement methods, translate text, or clean, reformat, or analyze data. The authors take full responsibility for the content of this paper.