LAFP: Preserving Latent Action Structure in Latent Policy Learning via Flow Matching

Jiexi Lyu[*] Fudan University jxlv23@m.fudan.edu.cn

Xizhou Bu[*] Fudan University xzbu24@m.fudan.edu.cn

Qingqiu Huang Morphi Robot draco.huang@morphi.com

Chufeng Tang Morphi Robot felix.tang@morphi.com

Xiaoshuai Hao Xiaomi EV haoxiaoshuai714@163.com

Hongbo Wang Fudan University Wanghongbo@fudan.edu.cn

Wei Li[†] Fudan University fd_liwei@fudan.edu.cn

Abstract

Learning high quality latent actions from large-scale unlabeled videos, coupled with limited real-world interaction data for training an action decoder, has emerged as a promising paradigm for scalable latent policy learning. However, existing approaches typically rely on behavior cloning, which tends to collapse inherently multimodal action distributions into unimodal ones, thereby degrading the pretrained latent action structure. While flow matching provides a potential alternative, directly applying it leads to a misalignment between latent actions and physical actions during action decoder training, due to the stochastic nature of the learned policy. To address these, we propose Latent Action Flow Policy (LAFP), which leverages flow matching for latent policy learning and introduces an inference-time interpolation mechanism to mitigate stochasticity-induced misalignment. Experimental results demonstrate that LAFP consistently outperforms prior methods on most downstream imitation learning tasks, achieving an 8.2% improvement in average success rate while incurring less than 1× additional inference overhead.

1 Introduction

In recent years, the severe scarcity of high quality action data has emerged as a fundamental bottleneck in advancing embodied intelligence. This has motivated a promising paradigm that learns latent actions from large-scale unlabeled video data, which can be naturally formulated as a three-stage pipeline [19, 20, 30]. In the pretraining stage, approaches such as the Latent Action Policies (LAPO) [23] adopt an autoencoding framework, where an inverse dynamics model (IDM) and a forward dynamics model (FDM) are jointly optimized under a reconstruction objective, encouraging latent actions to capture the temporal dynamics between consecutive observations. During the distillation stage, a latent policy is trained to predict latent actions produced by a frozen IDM, while in post-training, an action decoder is learned from limited real-world interaction data to map latent actions into executable physical actions.

Despite the effectiveness of this paradigm, existing works have primarily focused on improving the pretraining stage, emphasizing the learning of expressive latent action representations [13, 21, 16, 8, 26, 12] and their application to downstream such as vision-language-action (VLA)

[*]Equal contribution.

[†]Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

models [29, 3, 5, 6]. In contrast, the distillation stage, serving as a critical bridge between representation learning and control, remains insufficiently explored, particularly in preserving the intrinsic structure of the pretrained latent action space. The core issue is that the IDM is trained on a relatively simple, often unimodal and non-causal inverse dynamics problem, whereas the latent policy must capture inherently multimodal future uncertainties. Prior work [23, 9, 21] typically relies on behavior cloning (BC) [1] to distill knowledge from the IDM into the latent policy, which tends to collapse inherently multimodal action distributions into unimodal ones. A natural alternative is flow matching (FM) [15, 17, 14] for modeling multimodal action distributions, but naively applying it introduces a new challenge, as the stochasticity of the learned latent policy causes misalignment between latent actions and physical actions during post-training, degrading the effectiveness of the action decoder.

To address these, we propose Latent Action Flow Policy (LAFP), which leverages flow matching for the latent policy learning, preserving the well-structured representation learned in the pretraining phase. Additionally, in the post-training stage, we introduce an inference-time interpolation mechanism that performs flow matching inference by interpolating between the source Gaussian noise distribution and the target latent action distribution to train the action decoder, mitigating the misalignment between latent actions and physical actions. Our main contributions are as follows:

(1) The results on 16 PROCGEN tasks demonstrate that learning latent policies via flow matching can explicitly preserve the intrinsic structure of the pretrained latent action space, leading to consistent improvements in downstream imitation learning performance.

(2) we propose an inference time interpolation mechanism for training the action decoder under a flow-matching inference paradigm, enabling the resulting policy to achieve performance comparable to an action decoder directly trained with IDM supervision;

(3) Through ablation studies, we identify an optimal number of inference steps, which achieves an 8.2% improvement in average success rate while incurring less than 1× additional training and inference overhead.

2 Related work

2.1 Latent action model

Recent works have explored learning latent actions from videos through unsupervised or weakly supervised paradigms [7]. LAPO [23] introduces an unsupervised framework for latent action learning, while LAPA [29] extends this paradigm to VLA settings. To mitigate shortcut learning, CoMo [28] replaces the future observation input to the inverse dynamics model with the difference between future and current observations, discouraging direct encoding of future observations. UniVLA [6] combines DINOv2 [22] features with language conditioning to learn task-centric latent actions, reducing the influence of task-irrelevant dynamics. Other works introduce explicit supervision to constrain latent actions. LAOM [21] uses limited action annotations to train an auxiliary action decoder during pretraining, improving representation quality and robustness to distractors. LAOF [8] further explores optical flow as a substitute for action supervision, enabling stable and improved training of LAOM under extremely limited action labels. VILLA-X [9] introduces robot-state and action prediction constraints, while SkeLAM [18] employs robot skeleton supervision to constrain a kinematic latent component. Studies on limited action supervision consistently show that even incorporating around 10% labeled action data can significantly improve downstream imitation learning performance [21, 31, 2]. We therefore adopt LAOM as our baseline, using the available action labels while avoiding potential errors from additional supervision signals.

2.2 Generative policy learning

Generative policy learning has emerged as a dominant paradigm in embodied intelligence due to its ability to model multimodal action distributions. Diffusion-based policies [10] formulate action generation as a conditional denoising process, enabling diverse and feasible behaviors under identical observations. However, their iterative denoising process results in high inference latency, limiting real time applicability. To address this issue, flow matching [15, 17] based methods have been proposed as alternatives to diffusion models. These approaches directly learn continuous vector fields for state transitions, removing iterative denoising while maintaining high quality generation

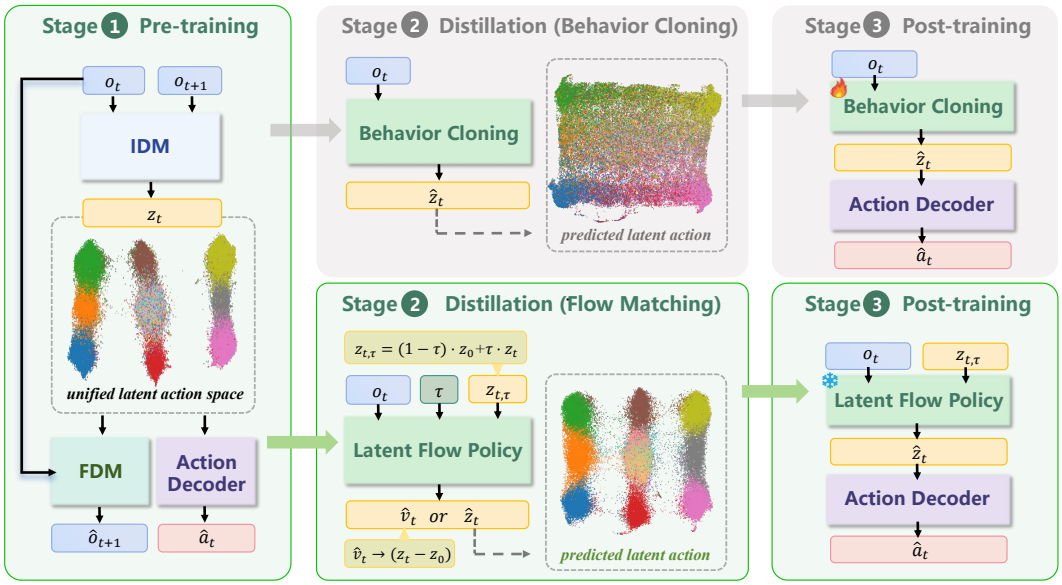


Figure 1: **Overview of LAFP.** Stage 1 learns an IDM-induced latent action space with an auxiliary action decoder. Stage 2 distills the frozen IDM targets using conditional flow matching. Stage 3 freezes the flow policy and trains the Post-AD with decoder aligned interpolation to reduce the mismatch between stochastic latent samples and physical action labels.

and significantly improving inference efficiency. For example, π_0 [4] adopts flow matching as its policy head and ManiFlow [27] combines flow matching with consistency training to achieve high quality actions in 1–2 inference steps. More recently, Just Image Transformer (JiT) [14] revisits generative modeling from an x-prediction perspective, showing that directly predicting clean data improves stability and distributional fidelity in high dimensional generation tasks. Motivated by these advances, our work investigates how different generative objectives, including latent action prediction and vector field prediction, affect downstream task performance under different latent action dimensions.

3 Method

Figure 1 presents Latent Action Flow Policy (LAFP) as a sequential three-stage pipeline. The latent action space learned from observation transitions in Stage 1 provides the targets for policy distillation in Stage 2, which learns to generate latent actions in this space from the current observation alone. In Stage 3, the flow policy remains fixed while a decoder is trained with limited action labels to map its latent outputs to executable actions. In this way, each stage builds on the preceding one while preserving the latent structure transferred through the pipeline.

3.1 Latent Action Pre-training

We follow LAOM [21] and jointly train an inverse dynamics model (IDM), a forward dynamics model (FDM), and an auxiliary action decoder (IDM-AD), as shown in Fig. 1. For an observation transition (o_t, o_{t+1}), where t denotes the environment time step, the IDM produces a latent action z_t, and the FDM reconstructs o_{t+1} from (o_t, z_t). A VQ-VAE codebook [25], updated using exponential moving averages, discretizes the latent space. Let e_t denote the codebook entry nearest to z_t. The representation objectives are

$$\mathcal{L}_{\text{VQ}} = \|z_t - e_t\|_2^2, \qquad \mathcal{L}_{\text{rec}} = \|\hat{o}_{t+1} - o_{t+1}\|_2^2. \tag{1}$$

Approximately 10% of the transitions include a physical action label a_t. The IDM-AD maps z_t to a_t, and its supervised loss $\mathcal{L}_{\text{IDM-AD}}$ is evaluated only on these labeled samples, whereas $\mathcal{L}_{\text{VQ}}$ and $\mathcal{L}_{\text{rec}}$ use the entire minibatch. Following LAOM, we separately set the weight of the supervised

IDM-AD loss to $\lambda_{\mathrm{AD}} = 0.1$. The complete objective is

$$\mathcal{L}_{\mathrm{stage1}} = \mathcal{L}_{\mathrm{VQ}} + \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{AD}}\mathcal{L}_{\text{IDM-AD}}. \tag{2}$$

The IDM-AD provides semantic regularization by encouraging latent actions to remain distinguishable by physical action class. Its action recovery accuracy therefore serves as a measure of the separability of the learned latent space. After latent action pre-training, the IDM, FDM, codebook, and IDM-AD are frozen.

3.2 Flow Matching Policy Distillation

The frozen IDM provides a latent target z_t for each observation transition. A behavior cloning policy trained with a mean squared error (MSE) loss estimates the conditional mean $\mathbb{E}[z_t \mid o_t]$. When $p(z_t \mid o_t)$ is multimodal, this objective can merge distinct action modes into a single prediction. LAFP instead models the conditional latent action distribution with a flow policy π_{flow}.

As shown in Fig. 2, for each training transition, we draw a source latent $z_0 \sim \mathcal{N}(0, I)$ and a flow time $\tau \sim \mathcal{U}[0, 1]$, then construct the linear probability path

$$z_t^\tau = (1 - \tau)z_0 + \tau z_t. \tag{3}$$

Here, z_t^τ denotes the latent state at flow time τ for the environment transition indexed by t. The policy receives (z_t^τ, τ, o_t). We implement π_{flow} as a three-layer MLP, using a lightweight CNN to encode o_t and sinusoidal embeddings to encode τ. We consider two output parameterizations of the flow policy, distinguished by their prediction targets. The x-prediction model directly predicts the target latent z_t,

$$\hat{z}_t = \pi_{\mathrm{flow}}(z_t^\tau, \tau, o_t), \qquad \mathcal{L}_x = \mathbb{E}\|z_t - \hat{z}_t\|_2^2, \tag{4}$$

whereas the v-prediction model predicts the displacement $v_t = z_t - z_0$ with $\mathcal{L}_v = \mathbb{E}\|v_t - \hat{v}_t\|_2^2$. Starting from Gaussian noise, Euler integration uses

$$u_\theta(z_t^\tau, \tau, o_t) = \begin{cases} (\hat{z}_t - z_t^\tau)/(1 - \tau), & \text{x-prediction}, \\ \hat{v}_t, & \text{v-prediction}, \end{cases} \tag{5}$$

followed by $z_t^{\tau+\Delta\tau} = z_t^\tau + \Delta\tau\, u_\theta$. Unless stated otherwise, LAFP uses x-prediction with three integration steps, which provides the best balance between stability and efficiency in our ablations.

3.3 Decoder Aligned Action Mapping

We train an action decoder (Post-AD) g_ϕ to map latent actions generated by the flow policy to physical actions. The main challenge arises from the stochasticity of the flow policy. While a deterministic behavior cloning policy yields a single latent–action pair for each labeled transition, different samples of z_0 can produce different plausible latent actions $\hat{z}_t$ for the same observation o_t, whereas the dataset provides only one recorded action a_t. Pairing every sampled latent with this action can therefore assign the same label to latents with different action semantics. To prevent decoder training from altering the latent distribution learned during policy distillation, we freeze π_{flow} and optimize only g_ϕ using the supervised decoding loss $\mathcal{L}_{\text{Post-AD}}$.

One way to avoid this mismatch is to train g_ϕ directly on the IDM target pairs (z_t, a_t), which assumes that the distilled policy remains aligned with the IDM latent space. Reusing the frozen IDM-AD without further training provides a stricter test of the same assumption. For comparison, Direct-Sample AD starts from Gaussian noise, applies the frozen flow solver, and trains g_ϕ on the resulting pairs $(\hat{z}_t, a_t)$. This exposes the decoder to policy generated latents but does not resolve the sampling induced label mismatch.

To retain policy generated decoder inputs while reducing this mismatch, LAFP uses decoder aligned interpolation. During Post-AD training, the IDM target z_t is available for each labeled transition. We therefore initialize flow inference from an intermediate point between z_0 and z_t, rather than from pure Gaussian noise, and then apply the frozen flow solver. The resulting latent remains policy generated but is constrained toward its transition specific IDM target, providing a more consistent correspondence with a_t. Because only the decoder training inputs are modified, the interpolation neither updates the flow policy nor changes its learned latent distribution. At deployment, the IDM target is unavailable, so inference starts from Gaussian noise and the trained Post-AD decodes the generated latent action.

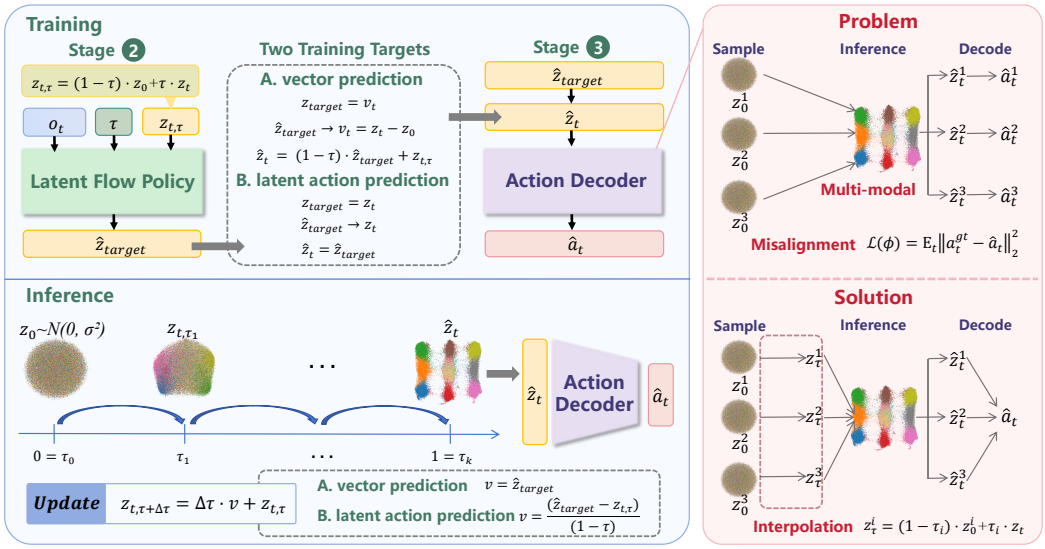


Figure 2: **Flow policy training and decoding.** *Top:* x-prediction and v-prediction parameterize the same linear probability path. *Right:* Direct sampling can pair different plausible latents with the same labeled action during Post-AD training; decoder aligned interpolation keeps the generated latent near its transition specific IDM target. *Bottom:* At deployment, the frozen flow policy transports Gaussian noise to a latent action, which the Post-AD maps to an environment action.

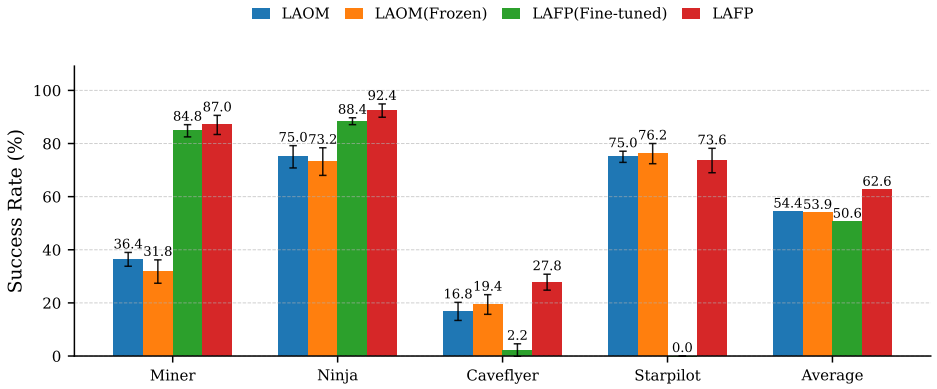


Figure 3: **Success rate comparison between LAOM and LAFP.** Results are averaged over 100 evaluation episodes across 5 random seeds and reported as mean ± standard deviation. Four representative environments are shown, corresponding to different behavioral categories: navigation (*CaveFlyer*), platforming (*Ninja*), collection (*Miner*), and combat (*StarPilot*). Full results on all 16 environments are provided in the Appendix. The rightmost panel reports the average success rate across all 16 environments, consistent with the following figures and tables.

4 Experiments

Our experiments address four questions: **(Q1)** Does the distilled policy remain aligned with the IDM latent action distribution? **(Q2)** Is stronger alignment associated with better downstream imitation performance? **(Q3)** Does decoder-aligned interpolation resolve the sampling-induced mismatch in Stage 3? **(Q4)** How do the prediction target and number of flow steps affect accuracy and efficiency?

4.1 Experimental setup

Benchmark. We evaluate our method on all 16 PROCGEN environments [11]. PROCGEN provides procedurally generated training and testing levels with substantial visual and structural varia-

Table 1: Impact of IDM-AD regularization on pretrained latent action quality and downstream policy performance. IDM label recovery is the IDM-AD accuracy on held-out IDM latent actions, and $\Delta\mathrm{Acc}$ denotes the gain in this accuracy produced by IDM-AD regularization. $\Delta\mathrm{Score}$ denotes the downstream policy gain (AD $-$ NAD). The largest policy improvement is highlighted in bold with a light red background.

Environment	ΔAcc	LAOM	LAOM (Frozen)	LAFP (Fine-tuned)	LAFP
CaveFlyer	13.3	+3.6	+6.2	−7.6	**+10.6**
Miner	7.5	−11.8	−9.2	+6.2	**+14.2**
Ninja	12.3	−4.8	−4.2	+10.4	**+11.4**
StarPilot	5.0	+0.4	−1.0	−0.2	**+4.8**
Average	7.8	+1.2	+6.4	+7.8	**+10.3**

tion, and covers diverse task categories, including navigation, platforming, collection, and combat. This diversity provides a broad test bed for evaluating the effectiveness and generality of our method across different task categories. Moreover, the discrete action spaces provide explicit action classes, making latent action separability, distributional alignment, and decoder transfer directly measurable. Since the environments differ in observations, dynamics, and action semantics, we train a separate policy for each environment. For each environment, we collect approximately 2.6M training frames and 6.5K held-out frames using an expert PPO policy [24] trained for 50M time steps. Latent action pre-training, latent policy distillation, and action decoder post-training run for 50K, 120K, and 5K optimization steps, respectively. The behavior cloning distillation baseline converges after 60K steps; extending its training to 120K yields negligible improvement. Unless stated otherwise, the latent action dimension is 128 and LAFP uses three flow steps for inference.

Baselines and variants. Our primary baseline is LAOM, which uses the same pre-training VQ-VAE latent space [25], including IDM-AD regularization, but distills the latent policy by behavior cloning. Both methods use the same 10% action-labeled data subset in latent action pre-training and action decoder post-training. We evaluate two post-training variants:

- **Fine-tuned**: the distilled latent policy and the Post-AD are updated jointly in action decoder post-training stage, allowing the latent representation to adapt to the downstream action space.
- **Frozen**: the distilled latent policy is fixed and only the Post-AD is trained, preserving the latent structure learned during distillation.

Accordingly, **LAOM** and **LAOM(Frozen)** are behavior cloning baselines, while **LAFP** denotes our complete method with a frozen latent policy and decoder aligned interpolation. We additionally include **LAFP(Fine-tuned)** and **Direct-Sample AD** as comparison variants to isolate the effects of policy freezing and decoder aligned interpolation, respectively.

Metrics. *IDM alignment* compares a frozen policy's latent distribution P with the IDM reference Q on the same held-out observations using squared maximum mean discrepancy (MMD),

$$\mathrm{MMD}^2(P, Q) = \mathbb{E}k(x, x') + \mathbb{E}k(y, y') - 2\mathbb{E}k(x, y), \tag{6}$$

where $x, x' \sim P$, $y, y' \sim Q$, and k is the Gaussian RBF kernel $k(x, y) = \exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right)$. Lower MMD means closer alignment to the IDM distribution. For downstream evaluation, *success rate* is the fraction of episodes whose cumulative reward reaches the environment specific maximum defined by Cobbe et al. [11]. Each method is evaluated over five random seeds, with 100 episodes run for each seed; we report the mean and standard deviation across seeds. Full per-environment MMD and success rate results appear in the Appendix.

4.2 Latent Action Structure and Downstream Performance

Latent action structure after policy distillation (Q1). We first examine whether the distilled policy retains the latent action structure learned by the IDM. The UMAP visualizations in Fig. 4 show that LAFP more closely preserves the IDM cluster structure, whereas LAOM(Frozen) shows

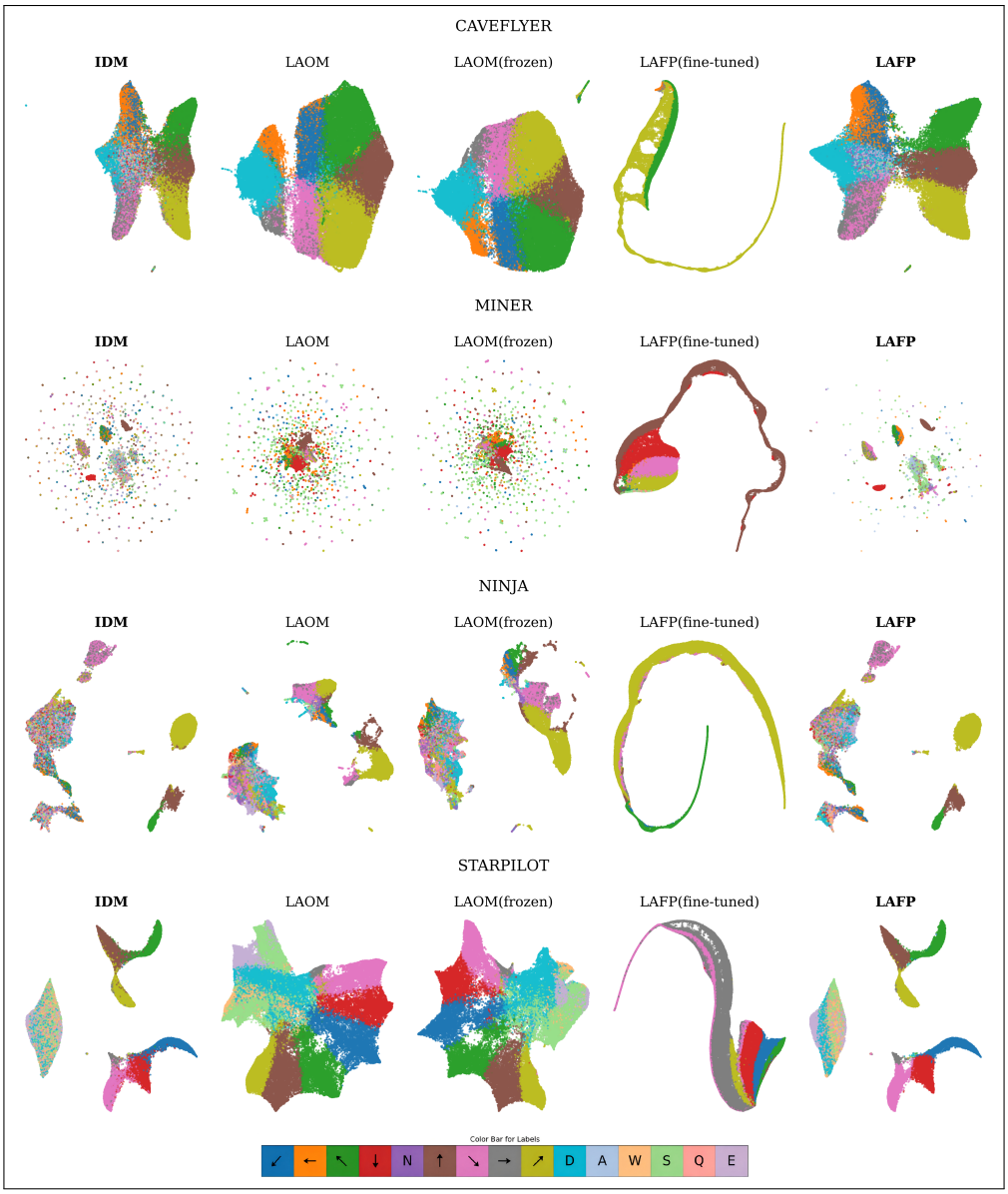


Figure 4: **UMAP projections of latent action spaces for the IDM and downstream policies across four representative environments.** Each row corresponds to one environment, while columns are ordered as the IDM, LAOM, LAOM(Frozen), LAFP(Fine-tuned) and LAFP latent space for reference. The LAFP and IDM columns are highlighted to facilitate direct comparison. Colors denote discrete action classes: the IDM latent space is colored using ground-truth action labels, while downstream policies are colored using decoded action labels.

larger deviations from the IDM reference across the four representative environments. Since UMAP provides only a qualitative two-dimensional projection, we additionally quantify distributional alignment in the full latent space using squared MMD. As shown in Table 2, LAFP obtains substantially lower MMD than LAOM(Frozen) in all four environments. The same trend holds across all 16 environments, with average MMD values of 0.00060 and 0.03846, respectively. To examine whether this alignment also preserves the action semantics encoded by the IDM, we replace each frozen policy's Post-AD with the pretrained IDM-AD without retraining the decoder. This replacement leaves LAFP's average success rate unchanged at 62.6%, but reduces LAOM(Frozen) from 53.9% to 44.6% (Table 3). Since the IDM-AD is trained only on IDM latents, its transfer performance indicates that LAFP remains more compatible with the action semantics of the IDM-induced latent

Table 2: Squared MMD between the distilled-policy latent distribution and the IDM reference in four representative PROCGEN environments. Lower values indicate better structure preservation. Results are averaged over five random seeds; complete results for all 16 environments are reported in Table 5.

Environment	LAFP	LAOM(Frozen)
CaveFlyer	**0.00045**	0.04481
Ninja	**0.00047**	0.01473
Miner	**0.00070**	0.08450
StarPilot	**0.00068**	0.00679

Table 3: Impact of decoder training and transfer on LAOM(Frozen) and LAFP. IDM-AD denotes the pretrained decoder reused without further optimization. Direct-Sample AD trains the Post-AD on latents generated from Gaussian noise using three flow steps, without decoder-aligned interpolation. The remaining columns use a Post-AD trained after policy distillation.

Environment	LAOM(Frozen) IDM-AD	LAOM(Frozen)	LAFP Direct-Sample AD	LAFP IDM-AD	LAFP
Miner	0.2±3.9	31.8±5.2	86.2±2.6	81.4±0.4	**87.0±2.5**
Ninja	63.8±2.1	73.2±3.7	92.4±1.3	**93.0±6.7**	92.4±3.0
CaveFlyer	19.4±5.3	19.4±4.4	18.4±4.4	**33.0±7.2**	27.8±3.6
StarPilot	70.6±2.7	76.2±3.8	73.8±4.5	**76.6±0.9**	73.6±4.6
Average	44.6	53.9	61.3	**62.6**	**62.6**

space. The combined evidence from UMAP, MMD, and decoder transfer shows that flow matching policy distillation more faithfully preserves the pretrained latent action structure.

Latent space alignment and downstream performance. We next investigate how alignment with the IDM latent distribution relates to downstream performance. Holding the environment and pretrained IDM latent space fixed, LAFP achieves lower MMD than LAOM(Frozen) in all 16 environments and an equal or higher success rate in 13 of them. Its average success rate is 62.6%, compared with 53.9% for LAOM(Frozen) and 54.4% for the fine-tuned LAOM baseline (Figure 3). The matched comparison with LAOM(Frozen) controls for policy updates during action decoder post-training and shows that LAFP combines closer alignment with the IDM distribution with higher downstream success. The fine-tuned variants are analyzed separately below since their latent spaces are further modified during post-training.

Fine-tuned versus frozen latent policies. The effect of fine-tuning differs between the two policy families. For LAFP, jointly updating the flow matching policy and Post-AD during action decoder post-training reduces the average success rate from 62.6% to 50.6%. Performance declines moderately in several environments and falls to zero in Dodgeball and StarPilot. Both tasks are sensitive to individual action errors and offer few alternative valid actions, leaving limited room for recovery once fine-tuning perturbs the pretrained latent geometry. Because flow matching learns a transport field over this geometry, such perturbations can distort the field and degrade performance. In contrast, fine-tuning changes LAOM only slightly, from 53.9% to 54.4%, because its pointwise mapping can adapt directly to the labeled action objective. This comparison shows that freezing the flow policy during action decoder post-training is important for retaining the structure learned during policy distillation.

4.3 Ablation Studies

IDM-AD regularization. Table 1 evaluates the effect of IDM-AD regularization during latent action pretraining. It improves IDM label recovery accuracy in all four representative environments, with an average $\Delta\mathrm{Acc}$ of $+7.8$ points across all 16 environments. For LAFP, this improvement is accompanied by an average downstream gain of 10.3 points, compared with 1.2 points for LAOM. Since LAFP distills the latent action distribution, clearer separation among action modes provides a better structured target for the flow matching policy. These results provide complementary evidence that latent action quality is important for downstream flow matching policy learning.

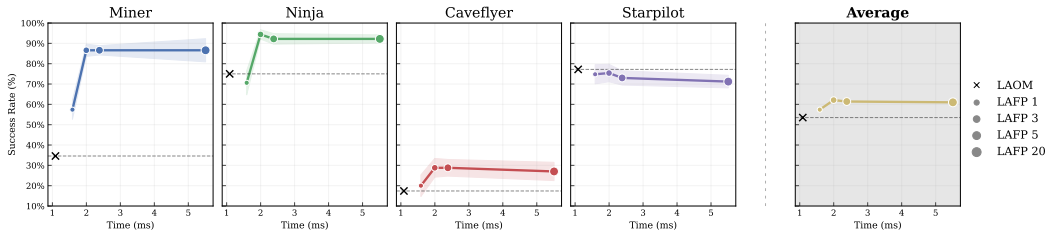


Figure 5: Performance and per-action inference time of flow matching under different inference step budgets, compared with behavior cloning. LAFP-k denotes a flow matching policy using k inference steps.

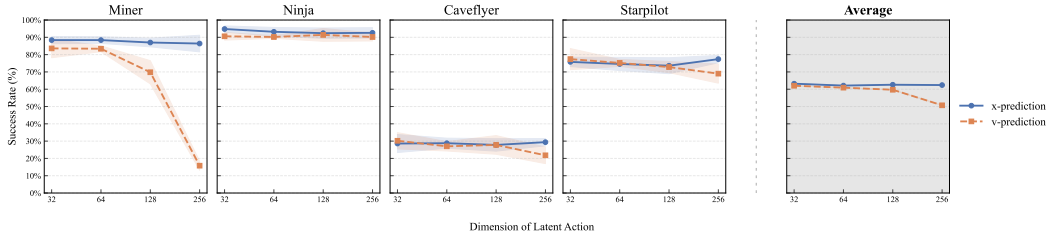


Figure 6: Comparison between latent action prediction (x-prediction) and vector field prediction (v-prediction) under different latent action dimensions.

Decoder aligned interpolation. To isolate the effect of interpolation, Direct-Sample AD uses the same frozen flow policy but trains the Post-AD on latents sampled directly from Gaussian noise. It achieves an average success rate of 61.3%, compared with 62.6% for LAFP. The difference is larger in CaveFlyer, where success increases from 18.4% to 27.8% with interpolation (Table 3). During decoder training, each generated latent is interpolated toward the transition specific IDM latent, maintaining consistency between the decoder input and its ground-truth action label without updating the frozen flow policy. The 9.4-point improvement in CaveFlyer, despite a 1.3-point average gain, suggests that interpolation is particularly beneficial in tasks with more multimodal action distributions, where stochastic latent samples are more likely to differ from the single action label available for decoder training.

Inference steps budget. Figure 5 compares performance and latency under different flow steps budgets. Performance improves with the first few steps and then saturates, while inference time increases monotonically. Three steps provide the best trade-off and are therefore used as the default setting. Additional steps initially improve the numerical integration of the learned flow, but provide little benefit once the trajectory is sufficiently resolved.

Prediction target and latent dimensionality. Figure 6 compares latent action prediction (x-prediction) with vector field prediction (v-prediction) across latent dimensions. X-prediction remains stable across all dimensions and performs better at dimensions 32–128, whereas v-prediction becomes less stable as dimensionality increases and shows the largest degradation at dimension 256. This trend suggests that direct supervision of the target latent is more stable as dimensionality grows, whereas errors in the predicted displacement can accumulate during integration.

5 Discussion

Our results indicate that preserving the IDM-induced latent structure during policy distillation and action decoder post-training is closely associated with downstream performance. LAFP achieves lower MMD to the IDM reference in every environment, allows the IDM-AD to be reused without an average performance loss, and performs best when the flow matching policy remains frozen during Post-AD training. Together, these results show that downstream decoding benefits when the distilled latents retain the distributional structure and action semantics of the pretrained IDM latent space. Flow matching supports this preservation by modeling the conditional latent distribu-

tion, while decoder aligned interpolation reduces the mismatch between stochastic latent samples and their unique action labels for each transition during Post-AD training. With three inference steps, LAFP provides a favorable balance between performance and efficiency, adding less than $1\times$ inference overhead relative to behavior cloning.

6 Limitations and future work

This study focuses on offline imitation learning in discrete PROCGEN environments, with a separate model trained for each game. Its effectiveness in continuous control, multi-task, and real world settings therefore remains to be evaluated. Moreover, jointly fine-tuning the flow policy and Post-AD is unstable in some tasks, motivating optimization strategies that adapt both components without disrupting the learned latent geometry. LAFP requires multiple flow inference steps and therefore has a higher computational cost than one-step policies, although the default three-step configuration introduces only a limited increase in latency. Future work will address these limitations through one-step flow distillation, structure preserving joint optimization, shared latent policies across tasks, and evaluation in continuous control and real world settings.

Acknowledgment

We thank Shanghai Institute for Mathematics and Interdisciplinary Sciences (SIMIS) for their financial support. This work was funded by SIMIS under Grant No. SIMIS-ID2025-RB. The authors are grateful for the resources and facilities provided by SIMIS, which were essential for the completion of this work.

References

[1] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine intelligence 15*, pages 103–129, 1995.

[2] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *CVPR*, 2026.

[3] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

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

[5] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *IROS*, 2025.

[6] 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 *RSS*, 2025.

[7] Xizhou Bu, Qingda Hu, Lei Zhou, Lingfeng Zhang, Yingbo Tang, Zihao Liu, Xinyi Tao, Zhiqiang Ma, Qingqiu Huang, Chufeng Tang, et al. What matters for latent actions in robot learning. *arXiv preprint arXiv:2608.19613*, 2026.

[8] Xizhou Bu, Jiexi Lyu, Fulei Sun, Ruichen Yang, Zhiqiang Ma, and Wei Li. LAOF: Robust latent action learning with optical flow constraints. In *CVPR*, 2026.

[9] Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, et al. VILLA-X: enhancing latent action modeling in vision-language-action models. In *ICLR*, 2026.

[10] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[11] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, 2020.

[12] Youngjoon Jeong, Junha Chun, and Taesup Kim. Learning to act robustly with view-invariant latent actions. *arXiv preprint arXiv:2601.02994*, 2026.

[13] Albina Klepach, Alexander Nikulin, Ilya Zisman, Denis Tarasov, Alexander Derevyagin, Andrei Polubarov, Nikita Lyubaykin, and Vladislav Kurenkov. Object-centric latent action learning. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025.

[14] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *CVPR*, 2026.

[15] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[16] Mingyu Liu, Jiuhe Shu, Hui Chen, Zeju Li, Canyu Zhao, Jiange Yang, Shenyuan Gao, Hao Chen, and Chunhua Shen. StaMo: Unsupervised learning of generalizable robot motion from compact state representation. In *CVPR*, 2026.

[17] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[18] Qi Lv, Jiayi Lin, Yuxin Jiang, Jianzong Wang, Xiang Deng, and Mike Shou Zheng. Skeleton-grounded latent action model for robot world models. *arXiv preprint arXiv:2610.31474*, 2026.

[19] Robert McCarthy, Daniel CH Tan, Dominik Schmidt, Fernando Acero, Nathan Herr, Yilun Du, Thomas G Thuruthel, and Zhibin Li. Towards generalist robot learning from internet video: A survey. *Journal of Artificial Intelligence Research*, 83, 2025.

[20] Dujun Nie, Fengjiao Chen, Qi Lv, Jun Kuang, Xiaoyu Li, and Xuezhi Cao. LARY: A latent action representation yielding benchmark. In *ECCV*, 2026.

[21] Alexander Nikulin, Ilya Zisman, Denis Tarasov, Nikita Lyubaykin, Andrei Polubarov, Igor Kiselev, and Vladislav Kurenkov. Latent action learning requires supervision in the presence of distractors. In *ICML*, 2025.

[22] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[23] Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *ICLR*, 2024.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[25] Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

[26] Chengen Xie, Chonghao Sima, Tianyu Li, Bin Sun, Junjie Wu, Zhihui Hao, and Hongyang Li. FLARE: Learning future-aware latent representations from vision-language models for autonomous driving. *arXiv preprint arXiv:2601.05611*, 2026.

[27] Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, et al. ManiFlow: A general robot manipulation policy via consistency flow training. *arXiv preprint arXiv:2509.01819*, 2025.

[28] Jiange Yang, Yansong Shi, Haoyi Zhu, Mingyu Liu, Kaijing Ma, Yating Wang, Gangshan Wu, Tong He, and Limin Wang. CoMo: Learning continuous latent motion from internet videos for scalable robot learning. In *CVPR*, 2026.

[29] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *ICLR*, 2025.

[30] Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, Jie Xu, et al. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint arXiv:2604.02029*, 2026.

[31] Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? In *NeurIPS*, 2025.

Appendix

Overview

This appendix provides supplementary details and experimental results omitted from the main paper due to page constraints. It is organized as follows:

- Sec. A reports success rates and MMD alignment scores across all 16 PROCGEN environments. Extended ablations cover IDM-AD regularization, Post-AD variants, the time–performance trade-off, and the flow prediction target.
- Sec. B provides additional UMAP visualizations of latent action spaces for the remaining environments not shown in the main paper.
- Sec. C details the model architecture, normalization procedure, and computational cost of all methods.

A Full results on all PROCGEN environments

In the main paper, we report results on four representative environments. To provide a comprehensive evaluation, we extend the comparison to all 16 PROCGEN environments.

Table 4: Performance comparison across all methods and all 16 PROCGEN environments (mean $\pm$ std, in %). LAFP achieves the highest average performance, with particularly pronounced gains in environments requiring diverse or multimodal behaviors.

Environment	**Method**			
	LAOM	**LAOM (Frozen)**	**LAFP (Fine-tuned)**	**LAFP**
bigfish	89.4 ± 1.8	88.0 ± 3.9	82.4 ± 1.8	88.8 ± 2.9
bossfight	45.8 ± 2.8	50.4 ± 4.2	34.2 ± 4.3	55.0 ± 7.1
caveflyer	16.8 ± 2.6	19.4 ± 4.4	2.2 ± 2.3	27.8 ± 3.6
chaser	19.8 ± 3.4	8.8 ± 2.8	27.0 ± 1.4	29.6 ± 7.4
climber	17.6 ± 4.2	20.4 ± 4.6	24.0 ± 4.2	29.0 ± 5.9
coinrun	93.4 ± 1.9	94.0 ± 2.7	94.4 ± 1.8	96.4 ± 1.1
dodgeball	35.0 ± 4.7	29.6 ± 4.3	0.0 ± 0.0	28.2 ± 6.2
fruitbot	39.2 ± 3.7	42.6 ± 6.0	33.0 ± 5.1	35.6 ± 3.2
heist	83.4 ± 4.0	83.2 ± 1.5	93.4 ± 0.9	97.2 ± 1.1
jumper	66.4 ± 3.2	69.6 ± 3.0	81.6 ± 4.5	79.4 ± 7.2
leaper	70.4 ± 5.5	71.2 ± 2.6	62.2 ± 1.6	73.0 ± 2.2
maze	98.4 ± 1.3	98.0 ± 1.7	93.0 ± 2.0	99.8 ± 0.4
miner	36.4 ± 4.2	31.8 ± 5.2	84.8 ± 1.3	87.0 ± 2.5
ninja	75.0 ± 3.4	73.2 ± 3.7	88.4 ± 2.4	92.4 ± 3.0
plunder	8.6 ± 2.9	5.8 ± 1.6	9.4 ± 4.6	8.0 ± 2.1
starpilot	75.0 ± 2.1	76.2 ± 3.8	0.0 ± 0.0	73.6 ± 4.6
Average	54.4	53.9	50.6	62.6

A.1 Quantitative Latent Space Alignment

Table 5 reports squared MMD between each frozen policy's latent action distribution and the pretrained IDM reference on the same held-out observations. LAFP is closer to the IDM than LAOM(Frozen) in every environment, with an average of 0.00060 versus 0.03846. This full dimensional measure complements the UMAP visualizations reported in the main paper.

A.2 Overall Performance

Table 4 summarizes the performance of all methods. Overall, the results consistently corroborate the conclusions drawn in the main paper. LAFP achieves superior performance over LAOM in the majority of environments, with particularly notable gains in tasks that require diverse or multimodal

Table 5: Squared MMD between each distilled policy latent distribution and the pretrained IDM reference across all 16 PROCGEN environments. Lower values indicate better structure preservation. Values are averaged over five random seeds.

Environment	LAFP	LAOM(Frozen)	Environment	LAFP	LAOM(Frozen)
BigFish	0.00055	0.03262	Heist	0.00021	0.05584
BossFight	0.00048	0.01464	Jumper	0.00061	0.06160
CaveFlyer	0.00045	0.04481	Leaper	0.00035	0.03780
Chaser	0.00051	0.00692	Maze	0.00025	0.00375
Climber	0.00092	0.01648	Miner	0.00070	0.08450
CoinRun	0.00110	0.00530	Ninja	0.00047	0.01473
Dodgeball	0.00021	0.01428	Plunder	0.00099	0.16817
FruitBot	0.00130	0.04756	StarPilot	0.00068	0.00679
	Average	LAFP: **0.00060**	LAOM(Frozen): **0.03846**		

Table 6: Ablation on the Stage-1 IDM-AD constraint. AD denotes models trained with the constraint and NAD denotes models trained without it.

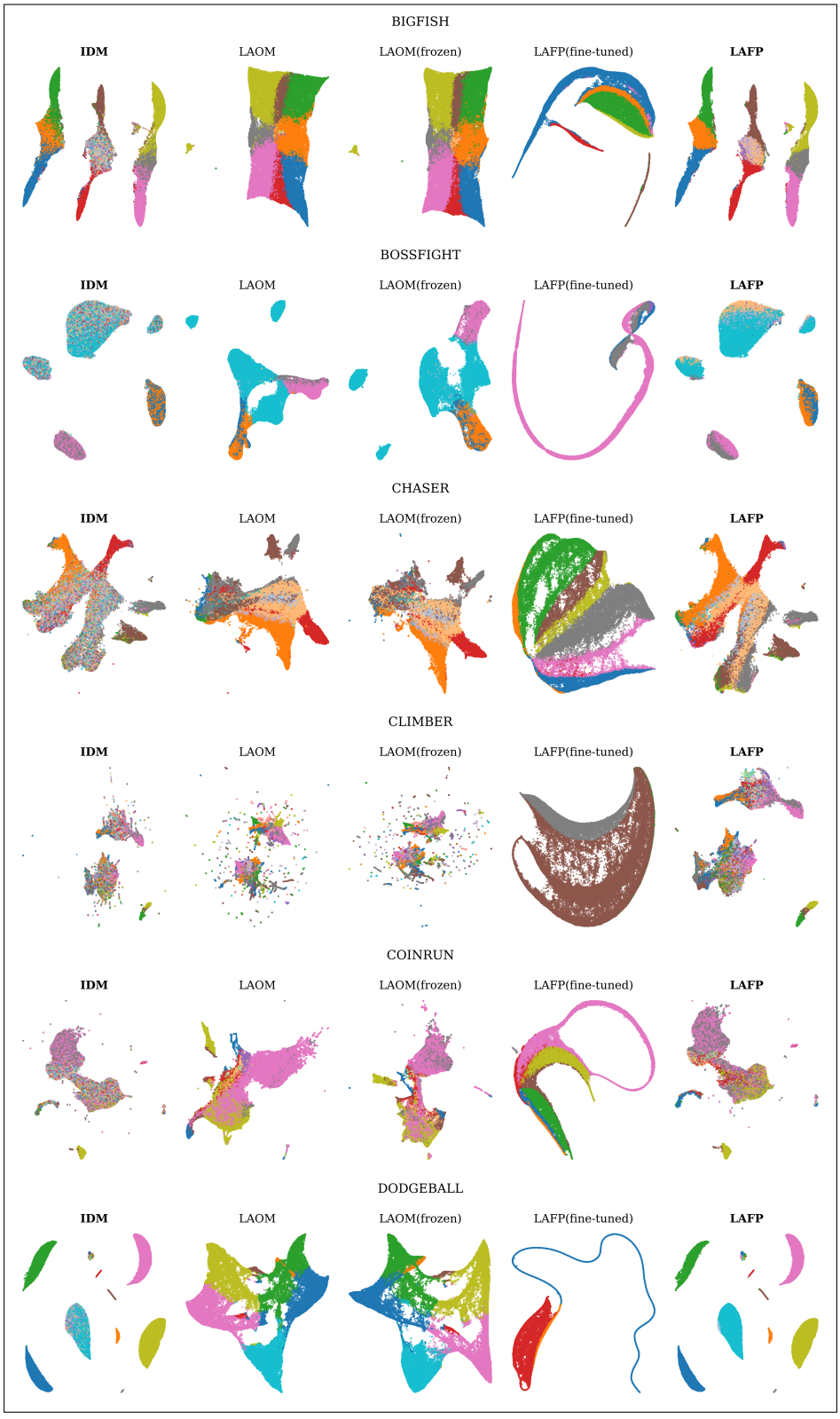
Environment	**Method**							
	LAOM		**LAOM (Frozen)**		**LAFP (Fine-tuned)**		**LAFP**	
	AD	NAD	AD	NAD	AD	NAD	AD	NAD
bigfish	89.4 ± 1.8	85.2 ± 6.1	88.0 ± 3.9	88.0 ± 3.0	82.4 ± 1.8	65.6 ± 5.5	88.8 ± 2.9	79.6 ± 6.1
bossfight	45.8 ± 2.8	13.8 ± 1.3	50.4 ± 4.2	3.0 ± 0.7	34.2 ± 4.3	0.8 ± 0.8	55.0 ± 7.1	0.8 ± 1.3
caveflyer	16.8 ± 2.6	13.2 ± 4.1	19.4 ± 4.4	13.2 ± 4.0	2.2 ± 2.3	9.8 ± 3.6	27.8 ± 3.6	17.2 ± 4.1
chaser	19.8 ± 3.4	25.8 ± 3.5	8.8 ± 2.8	4.2 ± 1.6	27.0 ± 1.4	0.8 ± 0.8	29.6 ± 7.4	28.6 ± 3.5
climber	17.6 ± 4.2	22.8 ± 3.3	20.4 ± 4.6	20.4 ± 4.6	24.0 ± 4.2	24.6 ± 3.2	29.0 ± 5.9	21.2 ± 3.3
coinrun	93.4 ± 1.9	88.0 ± 2.6	94.0 ± 2.7	75.0 ± 3.4	94.4 ± 1.8	48.2 ± 4.7	96.4 ± 1.1	87.8 ± 2.6
dodgeball	35.0 ± 4.7	31.6 ± 3.6	29.6 ± 4.3	37.0 ± 1.6	0.0 ± 0.0	28.4 ± 5.8	28.2 ± 6.2	29.6 ± 3.6
fruitbot	39.2 ± 3.7	38.4 ± 4.7	42.6 ± 6.0	34.0 ± 4.3	33.0 ± 5.1	18.8 ± 5.5	35.6 ± 3.2	31.2 ± 4.7
heist	83.4 ± 4.0	83.0 ± 4.9	83.2 ± 1.5	50.0 ± 7.8	93.4 ± 0.9	88.6 ± 3.8	97.2 ± 1.1	69.6 ± 4.9
jumper	66.4 ± 3.2	71.4 ± 3.0	69.6 ± 3.0	67.6 ± 5.2	81.6 ± 4.5	73.8 ± 3.1	79.4 ± 7.2	77.6 ± 3.0
leaper	70.4 ± 5.5	74.8 ± 3.5	71.2 ± 2.6	72.2 ± 3.3	62.2 ± 1.6	71.8 ± 4.1	73.0 ± 2.2	72.0 ± 3.5
maze	98.4 ± 1.3	98.2 ± 1.6	98.0 ± 1.7	97.8 ± 1.3	93.0 ± 2.0	97.8 ± 0.8	99.8 ± 0.4	98.0 ± 1.6
miner	36.4 ± 4.2	48.2 ± 4.5	31.8 ± 5.2	41.0 ± 6.3	84.8 ± 1.3	78.6 ± 4.0	87.0 ± 2.5	72.8 ± 4.5
ninja	75.0 ± 3.4	79.8 ± 1.7	73.2 ± 3.7	77.4 ± 3.6	88.4 ± 2.4	78.0 ± 3.6	92.4 ± 3.0	81.0 ± 1.7
plunder	8.6 ± 2.9	3.0 ± 0.5	5.8 ± 1.6	1.6 ± 1.5	9.4 ± 4.6	0.0 ± 0.0	8.0 ± 2.1	0.4 ± 0.5
starpilot	75.0 ± 2.1	74.6 ± 3.6	76.2 ± 3.8	77.2 ± 3.8	0.0 ± 0.0	0.2 ± 0.4	73.6 ± 4.6	68.8 ± 3.6
Average	54.4	53.2	53.9	47.5	50.6	42.9	62.6	52.3

behaviors, such as *chaser*, *jumper* and *miner*. In contrast, in relatively simple environments such as *coinrun* and *maze*, both methods perform comparably, suggesting that the advantage of LAFP mainly arises in scenarios where preserving action diversity is critical.

Across all environments, LAFP benefits from freezing the distilled latent policy during action decoder post-training. LAFP(Fine-tuned) updates the flow policy jointly with the Post-AD and decreases from 62.6% to 50.6% on average, including 0% success in Dodgeball and StarPilot. Both tasks are sensitive to individual action errors and offer few alternative valid actions, leaving limited room for recovery once fine-tuning perturbs the pretrained latent geometry. By contrast, fine-tuning changes LAOM only slightly because its pointwise mapping can adapt directly to the labeled action objective.

A.3 Additional Ablation Results

We report all ablations over five random seeds and 100 episodes per seed; standard deviations are computed across seeds. Table 6 shows that the Stage-1 IDM-AD constraint is particularly important for LAFP. Table 7 evaluates whether the frozen IDM-AD can be transferred after Stage 2, providing a functional test of latent space alignment. Table 8 shows that three to five flow steps provide the best performance–efficiency trade-off. Table 9 shows that x-prediction remains robust as the latent dimension grows, whereas v-prediction degrades in the highest dimensional setting.


BIGFISH
IDM
LAOM
LAOM(frozen)
LAFP(fine-tuned)
LAFP
BOSSFIGHT
IDM
LAOM
LAOM(frozen)
LAFP(fine-tuned)
LAFP
CHASER
IDM
LAOM
LAOM(frozen)
LAFP(fine-tuned)
LAFP
CLIMBER
IDM
LAOM
LAOM(frozen)
LAFP(fine-tuned)
LAFP
COINRUN
IDM
LAOM
LAOM(frozen)
LAFP(fine-tuned)
LAFP
DODGEBALL
IDM
LAOM
LAOM(frozen)
LAFP(fine-tuned)
LAFP

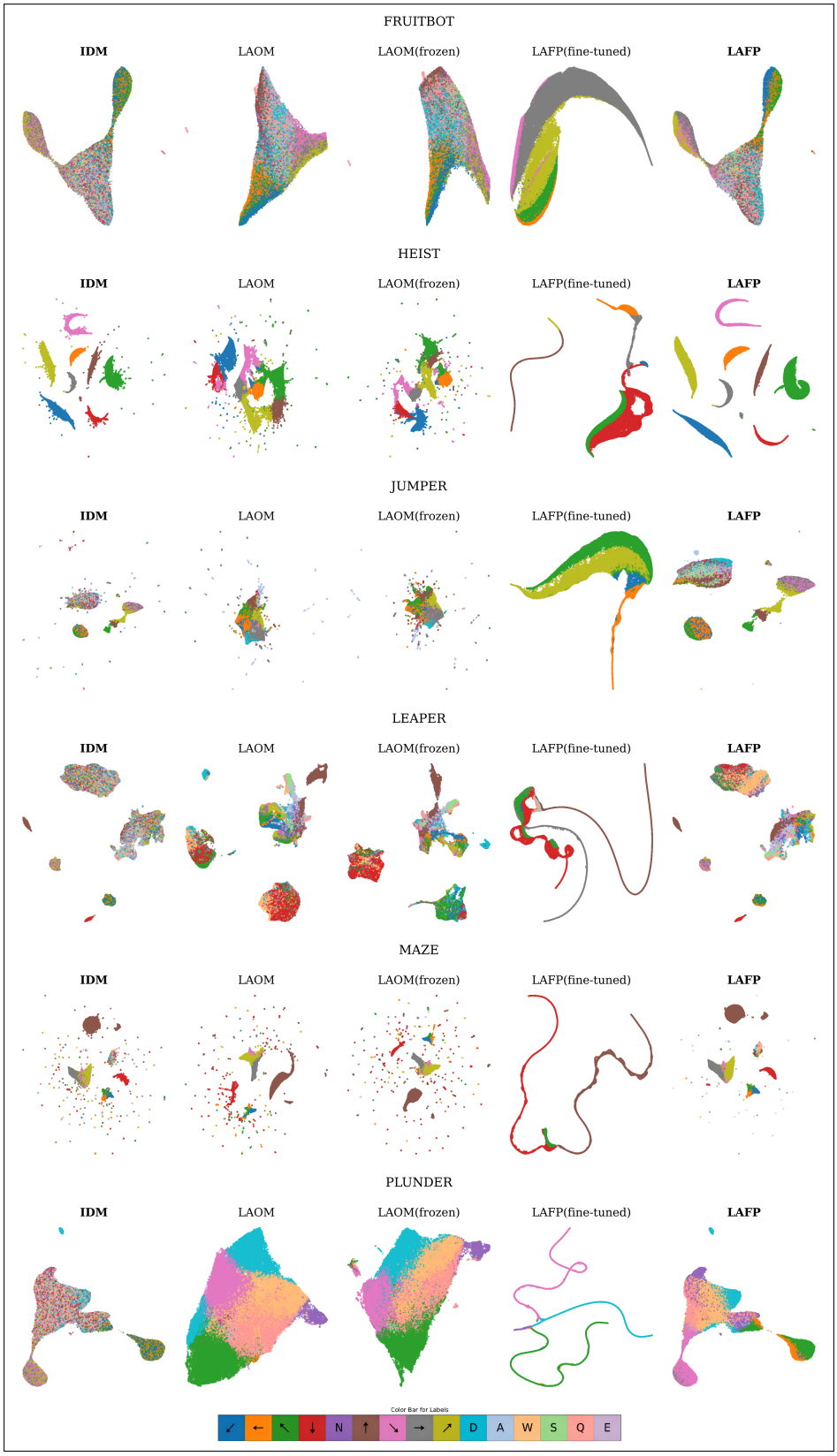


Figure 7: UMAP projections of latent action spaces for the IDM and downstream policies across the remaining 12 PROCGEN environments.

Table 7: Comparison of decoder designs. Reusing the pretrained IDM-AD benefits LAFP but often degrades LAOM, highlighting the importance of preserving latent structure for flow-based policies.

Environment	Method			
	LAOM (Frozen)		LAFP	
	Post-AD	IDM-AD	Post-AD	IDM-AD
bigfish	88.0 ± 3.9	89.4 ± 3.4	88.8 ± 2.9	87.6 ± 3.2
bossfight	50.4 ± 4.2	47.0 ± 6.1	55.0 ± 7.1	46.0 ± 3.8
caveflyer	19.4 ± 4.4	19.4 ± 5.3	27.8 ± 3.6	33.0 ± 7.2
chaser	8.8 ± 2.8	5.6 ± 4.8	29.6 ± 7.4	25.6 ± 1.8
climber	20.4 ± 4.6	14.8 ± 3.2	29.0 ± 5.9	29.6 ± 0.8
coinrun	94.0 ± 2.7	97.8 ± 2.3	96.4 ± 1.1	97.8 ± 1.1
dodgeball	29.6 ± 4.3	20.2 ± 2.6	28.2 ± 6.2	33.4 ± 2.9
fruitbot	42.6 ± 6.0	38.8 ± 7.3	35.6 ± 3.2	36.6 ± 5.2
heist	83.2 ± 1.5	12.2 ± 1.5	97.2 ± 1.1	96.8 ± 3.0
jumper	69.6 ± 3.0	62.0 ± 6.3	79.4 ± 7.2	83.4 ± 5.8
leaper	71.2 ± 2.6	70.2 ± 3.7	73.0 ± 2.2	75.0 ± 3.7
maze	98.0 ± 1.7	96.8 ± 0.8	99.8 ± 0.4	99.2 ± 1.9
miner	31.8 ± 5.2	0.2 ± 3.9	87.0 ± 2.5	81.4 ± 0.4
ninja	73.2 ± 3.7	63.8 ± 2.1	92.4 ± 3.0	93.0 ± 6.7
plunder	5.8 ± 1.6	4.4 ± 1.6	8.0 ± 2.1	6.2 ± 1.5
starpilot	76.2 ± 3.8	70.6 ± 2.7	73.6 ± 4.6	76.6 ± 0.9
Average	53.9	44.6	62.6	62.6

Table 8: Effect of the number of flow steps. A small number of steps (3–5) achieves the best balance between performance and efficiency, while more steps yield diminishing returns.

Environment	Model				
	LAFP(1-step)	**LAFP(3-steps)**	**LAFP(5-steps)**	**LAFP(20-steps)**	**LAOM**
bigfish	87.6 ± 4.5	88.8 ± 2.9	84.2 ± 4.8	84.8 ± 4.0	86.2 ± 1.9
bossfight	48.6 ± 2.4	55.0 ± 7.1	48.2 ± 3.3	51.4 ± 6.3	46.6 ± 2.1
caveflyer	20.0 ± 5.3	27.8 ± 3.6	28.8 ± 4.1	27.0 ± 4.5	17.4 ± 4.3
chaser	29.0 ± 1.9	29.6 ± 7.4	23.2 ± 5.8	15.2 ± 3.8	17.0 ± 2.1
climber	21.6 ± 3.8	29.0 ± 5.9	27.8 ± 2.0	28.8 ± 4.0	15.4 ± 2.2
coinrun	97.4 ± 1.8	96.4 ± 1.1	95.2 ± 2.5	96.6 ± 2.5	92.6 ± 3.2
dodgeball	32.4 ± 7.1	28.2 ± 6.2	26.8 ± 2.6	31.8 ± 2.9	31.2 ± 1.9
fruitbot	41.8 ± 7.0	35.6 ± 3.2	35.0 ± 5.3	33.6 ± 3.9	40.8 ± 6.4
heist	88.6 ± 2.5	97.2 ± 1.1	97.6 ± 0.9	95.2 ± 1.6	81.4 ± 2.7
jumper	71.4 ± 3.8	79.4 ± 7.2	84.8 ± 2.9	82.4 ± 4.3	65.4 ± 3.2
leaper	71.0 ± 4.0	73.0 ± 2.2	72.6 ± 2.4	73.4 ± 6.2	68.0 ± 4.3
maze	98.4 ± 0.9	99.8 ± 0.4	99.8 ± 0.4	99.6 ± 0.5	98.0 ± 0.7
miner	57.4 ± 4.7	87.0 ± 2.5	86.6 ± 2.3	86.6 ± 5.7	34.6 ± 5.4
ninja	70.6 ± 5.9	92.4 ± 3.0	92.2 ± 2.6	92.2 ± 2.0	75.0 ± 2.1
plunder	7.6 ± 2.1	8.0 ± 2.1	7.0 ± 2.4	5.6 ± 3.0	9.0 ± 1.6
starpilot	74.8 ± 4.7	73.6 ± 4.6	73.0 ± 3.5	71.2 ± 3.1	77.2 ± 4.2
Average	57.4	62.6	61.4	61.0	53.5
Inference time per action (ms)	1.59	2.00	2.38	5.51	1.09

B Additional UMAP Visualizations

Figure 7 provides UMAP visualizations for the remaining environments. The IDM forms clusters associated with different action modes. LAFP most closely retains this organization, whereas LAOM and LAFP(Fine-tuned) tend to distort or contract multiple modes. UMAP is only qualitative; the full dimensional MMD results in Table 5 and the IDM-AD transfer test provide the corresponding quantitative evidence.

Table 9: Effect of latent dimension and prediction target. Direct latent prediction remains robust across dimensions, while vector field prediction degrades significantly in high dimensional settings.

Environment	Dimension & Predicted Target							
	32		64		128		256	
	latent	vector	latent	vector	latent	vector	latent	vector
bigfish	85.4 ± 5.3	88.4 ± 2.5	87.4 ± 3.2	87.4 ± 4.0	88.8 ± 2.9	86.4 ± 4.1	86.0 ± 2.9	38.4 ± 3.6
bossfight	55.0 ± 8.1	53.4 ± 4.3	48.4 ± 4.0	52.0 ± 3.2	55.0 ± 7.1	49.4 ± 4.5	46.6 ± 3.3	49.6 ± 4.4
caveflyer	28.6 ± 5.2	30.2 ± 4.7	28.8 ± 2.9	27.0 ± 2.7	27.8 ± 3.6	28.8 ± 5.4	29.4 ± 2.1	21.8 ± 4.8
chaser	35.2 ± 4.7	31.6 ± 5.2	27.0 ± 3.8	26.2 ± 3.7	29.6 ± 7.4	28.8 ± 3.1	26.8 ± 3.8	0.0 ± 0.0
climber	25.6 ± 5.3	23.0 ± 3.1	26.4 ± 6.0	23.6 ± 6.5	29.0 ± 5.9	25.0 ± 7.4	27.4 ± 1.1	28.4 ± 5.0
coinrun	97.4 ± 1.1	94.2 ± 2.8	95.0 ± 1.4	93.0 ± 2.2	96.4 ± 1.1	97.0 ± 1.2	96.8 ± 2.8	90.8 ± 4.1
dodgeball	29.6 ± 2.7	32.2 ± 6.5	30.8 ± 5.4	28.2 ± 1.9	28.2 ± 6.2	28.0 ± 5.6	29.0 ± 5.8	29.0 ± 5.8
fruitbot	37.0 ± 4.2	32.8 ± 4.8	37.2 ± 7.4	31.6 ± 3.4	35.6 ± 3.2	35.2 ± 4.1	39.8 ± 7.3	32.2 ± 2.2
heist	98.4 ± 1.5	98.8 ± 0.8	96.6 ± 1.5	96.4 ± 1.5	97.2 ± 1.1	94.2 ± 1.9	97.2 ± 1.9	95.4 ± 1.9
jumper	82.2 ± 1.3	81.2 ± 3.7	84.6 ± 0.9	82.4 ± 3.8	79.4 ± 7.2	79.2 ± 2.9	83.6 ± 5.4	78.4 ± 2.9
leaper	71.6 ± 3.9	72.8 ± 2.9	72.4 ± 3.5	73.8 ± 5.5	73.0 ± 2.2	71.4 ± 2.7	73.6 ± 4.4	71.6 ± 4.8
maze	99.4 ± 0.9	99.6 ± 0.5	98.4 ± 1.9	100.0 ± 0.0	99.8 ± 0.4	99.6 ± 0.5	99.6 ± 0.5	100.0 ± 0.0
miner	88.4 ± 2.1	83.6 ± 5.4	84.4 ± 1.9	83.4 ± 1.7	87.0 ± 2.5	77.0 ± 6.7	86.4 ± 4.8	15.8 ± 2.6
ninja	94.8 ± 1.8	90.6 ± 2.2	93.2 ± 2.5	90.2 ± 0.8	92.4 ± 3.0	92.8 ± 3.7	92.6 ± 3.0	90.2 ± 2.3
plunder	7.0 ± 2.0	1.8 ± 1.9	9.0 ± 4.2	3.6 ± 1.7	8.0 ± 2.1	2.2 ± 1.1	6.8 ± 2.4	0.2 ± 0.4
starpilot	75.8 ± 2.7	77.4 ± 6.1	74.6 ± 3.8	75.2 ± 2.0	73.6 ± 4.6	73.6 ± 3.2	77.4 ± 2.1	69.0 ± 5.5
Average	63.2	62.0	62.1	60.9	62.6	60.5	62.4	50.7

C Implementation Details

Models. All methods share architectures and optimization settings across environments, without per-environment tuning. Our implementation builds on the official LAPO codebase. The IDM uses a lightweight CNN encoder, the FDM uses a U-Net, and the flow policy is a three-layer MLP conditioned on CNN observation features and a sinusoidal embedding of flow time τ. Both IDM-AD and Post-AD are three-layer MLPs.

Normalization. After pre-training, we compute per-dimension statistics of the latent action space for each environment. Specifically, the center (midrange) is defined as the average of the maximum and minimum values along each dimension, and the standard deviation is computed accordingly. These statistics are used to normalize latent actions during both training and inference in subsequent stages, which improves numerical stability and ensures consistent scaling across environments.

Computational cost. All experiments are conducted on a single NVIDIA H20 GPU per environment under the same computational budget. In terms of computational cost, pre-training takes approximately 50 minutes per environment. Distillation requires around one hour for LAFP and 40-50 minutes for LAOM (behavior cloning), suggesting that the additional cost of flow matching is relatively minor. The post-training stage is lightweight, requiring only a few minutes. Overall, the total training time per environment remains within a practical range.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction accurately reflect the contributions and scope of this work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Sec. 6

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This work does not include formal theoretical results such as theorems or proofs; the contributions are primarily empirical and methodological.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide sufficient details to reproduce the main experimental results. The model architecture, training objectives, and optimization procedures are described in Sec. 3 and Appendix. C, and dataset preprocessing and evaluation protocols are detailed in Sec. 4.1. Additional implementation details are provided in the appendix to ensure reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide open access to the code and relevant resources in the supplemental material (anonymized). The repository includes training scripts and evaluation pipelines.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all necessary experimental details, including dataset splits, model configurations, training hyperparameters, and optimization settings in Sec.4.1 and the Appendix. C. These details are sufficient to understand and reproduce the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report statistical variability using (e.g., mean and standard deviation over 5 seeds), as shown in Sec. 4.1. The error bars reflect variation due to random initialization and data sampling.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details of the compute resources used in Appendix.C, including hardware (NVIDIA H20 GPU), training time, and approximate computational cost. These details allow estimation of the resources required to reproduce our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research complies with the NeurIPS Code of Ethics. We ensure responsible use of data and models, and no ethical concerns arise from the methodology or experiments conducted in this work.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work focuses on foundational methodological improvements and does not have immediate or direct societal impact. Potential broader impacts are therefore not explicitly discussed.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The proposed method does not introduce assets or models with a high risk of misuse and no specific safeguards are required.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: All datasets (PROCGEN) and codebases (LAPO) used in this work are properly cited in Sec. 4.1 and Appendix. C. But We didn't find their licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: This work does not introduce new datasets or assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve human subjects or crowdsourcing experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve human subjects and IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as a component of the proposed method and no declaration is required.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.