Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

Learning Native Continuation for Action Chunking Flow Policies

Yufeng Liu[1,2], Hang Yu[2,4], Juntu Zhao[1,2], Bocheng Li[2,5], Di Zhang[2,4], Mingzhu Li[2], Wenxuan Wu[2], Yingdong Hu[3], Junyuan Xie[2], Junliang Guo[2‡], Dequan Wang[1†], Yang Gao[2,3†]
[1]Shanghai Jiao Tong University [2]Spirit AI [3]Tsinghua University
[4]Tongji University [5]University of Science and Technology of China
Project page: lyfeng001.github.io/Legato/

***Abstract*—Action chunking enables Vision Language Action (VLA) models to run in real time, but naive chunked execution often exhibits discontinuities at chunk boundaries. Real-Time Chunking (RTC) alleviates this issue but is external to the policy, leading to spurious multimodal switching and trajectories that are not intrinsically smooth. We propose Legato, a training-time continuation method for action-chunked flow-based VLA policies. Specifically, Legato initializes denoising from a schedule-shaped mixture of known actions and noise, exposing the model to partial action information. Moreover, Legato reshapes the learned flow dynamics to ensure that the denoising process remains consistent between training and inference under per-step guidance. Legato further uses randomized schedule condition during training to support varying inference delays and achieve controllable smoothness. Empirically, Legato produces smoother trajectories and reduces spurious multimodal switching during execution, leading to less hesitation and shorter task completion time. Extensive real-world experiments show that Legato consistently outperforms RTC across five manipulation tasks, achieving approximately 10% improvements in both trajectory smoothness and task completion time.**

I. Introduction

Action chunking [20] has become a widely adopted strategy for deploying large Vision Language Action (VLA) models in real-world robotic systems [7, 19, 37–43, 46]. By predicting sequences of action vectors, chunking amortizes inference cost and enables high-frequency control. However, naive chunked execution introduces a fundamental drawback: due to inference delay and the intrinsic multimodality of flow-based policies [24], transitions between consecutive chunks are often not smooth, leading to visible discontinuities during execution.

Real-Time Chunking (RTC) [8] was proposed to mitigate this issue by applying inference-time inpainting [29, 32] that partially constrains newly generated action chunks to previously generated actions in their overlapping regions. While RTC improves continuity compared to naive chunked execution, its continuation mechanism is applied only at inference time and is not learned as part of the policy. As a result, the policy is prone to spurious multimodal switching across chunk boundaries and producing trajectories that are not intrinsically smooth. Spurious multimodal switching often manifests as hesitation and abrupt direction changes, resulting in prolonged task completion time, as shown in fig. 1.

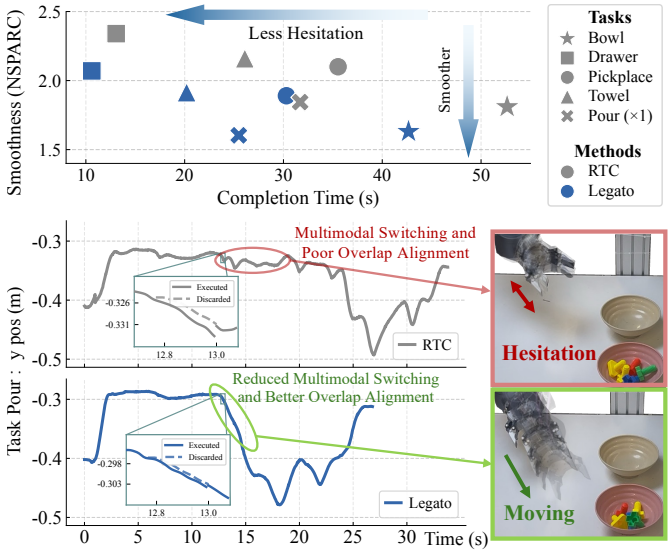

Fig. 1. Legato reduces task completion time while improving trajectory smoothness compared to RTC [8]. Across five real-world manipulation tasks, Legato consistently achieves shorter execution time and lower NSPARC [2] (indicating smoother trajectories, discussed in section IV-A2) than RTC. The bottom plot shows an example execution trace on the *pour* task, as defined in section IV-A1, where Legato produces smoother action trajectories with fewer hesitation-induced slowdowns than RTC.

In this work, we argue that stable chunked execution requires chunk continuation to be a native, learned property of the policy. Achieving this entails two requirements: **(i) per-step guidance**, where guidance is applied repeatedly across denoising steps, and **(ii) training-inference consistency**. We propose Legato, a training-time continuation mechanism for action-chunked flow-based VLA policies. Rather than learning the canonical flow-matching velocity field [6, 24] and relying on inference-time correction, Legato internalizes chunk-to-chunk continuation into the learned denoising dynamics.

To satisfy requirement *per-step guidance*, we first define a guidance schedule that specifies how strongly each timestep should adhere to the guidance actions. Unlike Training-time RTC [9], which enforces continuation via a hard clamp on the prefix, Legato uses a smooth schedule: it anchors the beginning of the chunk to known actions and gradually ramps

This work was done during the internship at Spirit AI.
† Corresponding authors. ‡ Project leader.

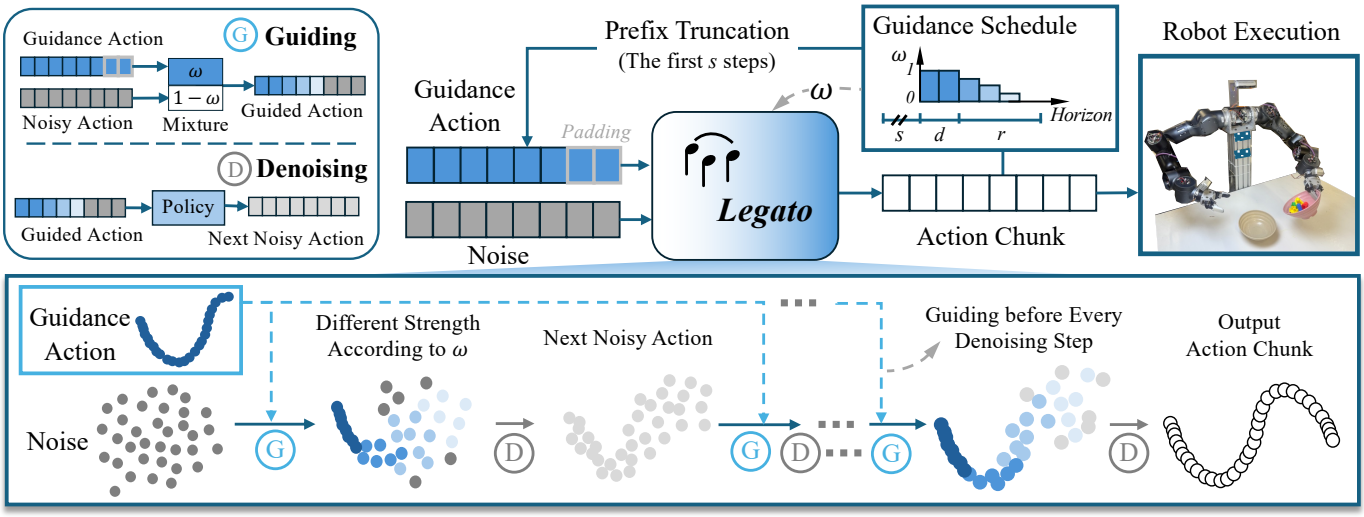

Fig. 2. Overview of Legato with schedule-shaped continuation dynamics. The schedule parameters are defined as follows: s is the executed length per cycle, d sets the fully guided prefix (inference delay), and r controls the ramp-down length of the guidance schedule over the remaining horizon. Given $\boldsymbol{\omega}$, Legato initializes actions via an action–noise mixture and learns a reshaped velocity field so that the native schedule effect is realized during multi-step denoising.

the guidance strength down to zero. During training, the known actions are the ground-truth of the same chunk [8]. During inference, the known actions correspond to the overlapping prefix of the previously generated chunk. This schedule-shaped design provides fine-grained control over the continuity strength between adjacent chunks.

With the schedule-shaped guidance, we enforce requirement *training-inference consistency* under per-step guidance. At inference time, action generation proceeds through multiple denoising steps, and empirically proved by section III-B, effective continuation requires per-step guidance before every denoising step.

Training-time RTC [9] achieves this by hard-fixing the executed prefix and learning to denoise only the remaining horizon. In contrast, Legato trains the policy to generate the entire chunk under per-step, schedule-shaped guidance by reshaping the velocity field. This yields strict training-inference consistency, as shown in fig. 2.

To make the above dynamics usable in real-world deployments, we account for variations in inference latency and desired continuation strength. In real-world deployment, inference latency can vary across hardware and runtime optimizations [8]. Under a fixed guidance schedule, such variations lead to mismatched overlap regions and require retraining to maintain consistent behavior. At the same time, we may want to adjust the schedule (i.e., ramp length) to control how strongly continuation is enforced. To handle both factors, we randomize the schedule parameters during training and condition the policy on the resulting schedule, so the same model can adapt to different latencies and ramp lengths.

We evaluate Legato extensively in real-world environments to assess the necessity of learning action continuation as part of the policy dynamics. We consider five diverse robotic manipulation tasks. Across all settings, Legato consistently produces smoother trajectories and achieves significantly shorter task completion time by suppressing spurious multimodal switching compared to RTC, as shown in fig. 1. Additional ablation studies further validate the robustness of Legato across different guidance schedules, VLA models, and conditioning strategies, demonstrating that its learned continuation behavior generalizes well under varying inference conditions.

Our work offers three main contributions:

- We propose Legato, a training-time continuation framework that enables per-step, schedule-shaped guidance while maintaining strict training-inference consistency by reshaping the flow dynamics of action-chunked policies.
- We introduce randomized schedule conditioning to support varying inference delays and to provide flexible control over trajectory smoothness.
- Extensive real-robot experiments across five manipulation tasks show that Legato consistently outperforms RTC and training-time RTC, producing smoother trajectories and shorter task completion time.

II. Related Works

A. VLA and Action Chunking Methods

Recent Vision Language Action (VLA) models couple large vision–language representations with learned heads to enable end-to-end visuomotor policies [5–7, 11, 22, 25, 33–35, 44, 45]. Most VLA systems generate actions in chunks, predicting a sequence of future controls per inference step [30, 35, 42]. Action chunking has been successfully combined with a variety of generative policy formulations, including diffusion-based [3, 13, 14, 21, 27, 36, 38], flow-based [6, 7, 10, 23], and discrete action representations [4, 28]. However, chunked execution trades off responsiveness for smoothness [8], and inference latency further increases discontinuities between successive chunks, motivating methods to improve continuation.

B. Trajectory Continuation in Learned Policies

Building on action chunking, a common approach to improve responsiveness is asynchronous execution, where action generation overlaps with execution [31, 43]; however, without explicit continuation constraints, independently generated chunks can exhibit abrupt multimodal switches at their boundaries. Bidirectional decoding (BID) [26] uses rejection sampling to keep continuity across chunks. Real-time chunking (RTC) [8] addresses continuation under asynchronous inference by conditioning new action chunks on previously issued actions that are guaranteed to execute. While RTC effectively mitigates boundary artifacts caused by inference latency, it is an inference-time mechanisms, leaving open the question of how to induce robust trajectory continuation without additional test-time intervention.

C. Conditioning in Diffusion- and Flow-Based Policies

Recent diffusion- [16] and flow-based [24] policies explore conditioning mechanisms to improve temporal coherence and execution efficiency. Diffusion Forcing [12] and Fast Policy Synthesis with Variable Noise Diffusion Models [17] adopt a timestep-level diffusion formulation, generating a single action per inference step and improving reactivity through noise modulation, but without explicitly modeling continuation across action chunks. Rolling Diffusion Policy [18] similarly operates at the timestep level, incrementally refining future actions via rolling denoising to enhance temporal awareness. In contrast, SAIL [1] performs chunk-level conditioning by leveraging overlapping actions between consecutive chunks using classifier-free guidance [15], which mitigates discontinuities under fast execution but provides only soft alignment and limited control over continuation strength. Concurrent with our work, training-time RTC [9] introduces continuation during training by conditioning on a hard action prefix that simulates inference delay. While this exposes the policy to prefix-based continuation, the conditioning remains an external constraint and does not account for the effective denoising dynamics induced by repeated, schedule-shaped guidance at inference time, leaving continuation outside the learned policy dynamics.

III. Methodology

A. Preliminaries

We consider Vision Language Action (VLA) policies that generate action sequences in fixed-length chunks using flow-based generative models. Let $\mathbf{A} \in \mathbb{R}^{H \times D_a}$ denote a ground-truth action chunk of horizon H, where D_a is the action dimension, and let $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ denote Gaussian noise of the same shape. Flow matching (FM) [24] constructs a continuous-time interpolation between noise and action, and trains a neural velocity field to transport samples along this path.

1) ***Flow matching****:* Given a time variable $t \in [0, 1]$, standard flow matching defines the interpolation

$$\mathbf{X}_t = (1 - t)\,\boldsymbol{\epsilon} + t\,\mathbf{A}, \tag{1}$$

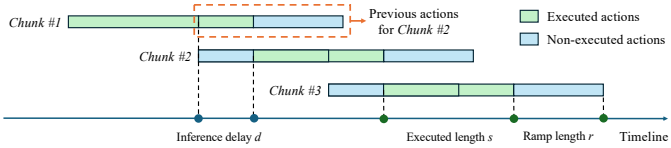

Fig. 3. Continuous chunk execution and guidance construction: d (full guidance), r (ramp-down), s (executed length), $r{+}s{+}d{=}H$.

and supervises the model to predict the corresponding velocity field

$$\mathbf{u}^{\mathrm{FM}}(\mathbf{X}_t, t) = \mathbf{A} - \boldsymbol{\epsilon}. \tag{2}$$

At inference time, action generation begins from an initial noise sample $\boldsymbol{\epsilon}$ and progressively transforms it into an action chunk by integrating the learned velocity field from $t = 0$ to $t = 1$.

2) ***Real-Time Chunking****:* Real-Time Chunking (RTC) [8] enforces continuity between successive action chunks through a test-time guidance mechanism inspired by inpainting, which encourages partial agreement with previously generated actions.

Beyond continuity, RTC also introduces an asynchronous execution scheme that overlaps inference and action execution to mitigate model latency. For an action chunk of horizon H, the first d timesteps correspond to inference latency, during which the robot continues executing the previous chunk. The next s timesteps correspond to the portion of the current chunk that will be executed before the next inference completes. Once $(s - d)$ timesteps of this portion have been executed, inference for the next chunk is triggered while execution continues, enabling overlapped computation and control.

RTC further employs a structured guidance schedule over the chunk horizon. The initial d timesteps receive full guidance to strictly enforce continuity with past actions, followed by a ramp-down phase. Let r denote the length of this ramp; the schedule commonly satisfies

$$r + s + d = H. \tag{3}$$

This design enforces strong adherence to previously executed actions near the chunk boundary while gradually relaxing constraints toward the end of the horizon.

Our method draws inspiration from RTC in both its use of previously executed actions and its structured guidance scheduling, as shown in fig. 3, green regions indicate actual execution.

B. Why per-step guidance matters?

We aim to learn a policy that remains strictly consistent between training and inference. To satisfy this requirement, guidance must be incorporated during training. Under the standard FM formulation, training optimizes the velocity field that transports samples from an initial noise to the ground-truth action. The only inference strategy that remains strictly consistent with training is to apply guidance only once at initialization and then perform multi-step denoising.

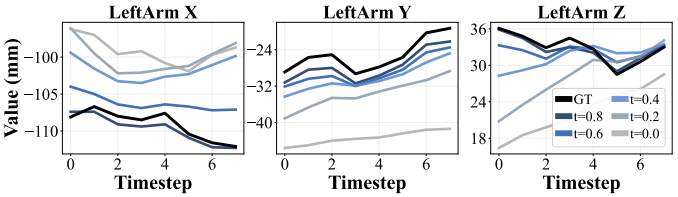

Fig. 4. One-shot prefix guidance cannot preserve prefix constraints during denoising. Trajectories show three dimensions of the overlap (prefix) actions across denoising steps; colors indicate diffusion times t (from 1 to 0), and GT denotes the ground-truth prefix. Although clamped at initialization, the overlap actions drift from the reference as denoising proceeds, motivating the need for per-step guidance. Evaluated on the *pour* task, as defined in section IV-A1.

To verify whether the one-shot guidance is enough for continuous guidance, we trained a flow policy where the standard noise initialization was replaced with a prefix-guided variant $\boldsymbol{\epsilon}'$. Let $\mathbf{m} \in \{0,1\}^H$ denote a horizon-wise mask for the overlap region, and let $\mathbf{A}_{\text{ref}}$ be the ground-truth reference actions on this overlap. We construct the guided noise:

$$\boldsymbol{\epsilon}' = (\mathbf{1} - \mathbf{m}) \odot \boldsymbol{\epsilon} + \mathbf{m} \odot \mathbf{A}_{\text{ref}}, \ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4}$$

where $\odot$ denotes element-wise multiplication. Training follows standard flow matching, but with $\boldsymbol{\epsilon}'$ as the start point:

$$\mathbf{X}_t = (1-t)\,\boldsymbol{\epsilon}' + t\,\mathbf{A}, \ \mathbf{u}^{\text{FM}}(\mathbf{X}_t, t) = \mathbf{A} - \boldsymbol{\epsilon}'. \tag{5}$$

During inference, we apply the same prefix clamp only at initialization, $\mathbf{X}_0 = \boldsymbol{\epsilon}'$, and then perform standard multi-step denoising without any intermediate guidance.

However, empirical results reveal that one-shot guidance is insufficient for continuous guidance. As the denoising process iterates, the overlap region in the generated chunk progressively deviates from the reference actions as shown in fig. 4 (details are shown in the appendix). The prefix part moves away from the desired constraint without repeated guidance.

This study leads to a crucial conclusion: **effective continuation requires per-step guidance.** However, simply applying per-step guidance on the standard FM remains inconsistent with the training objective. This dilemma motivates Legato: we reshape the flow dynamics during training so that the model can support per-step, schedule-shaped guidance while remaining fully consistent with the training objective.

C. Native Continuation for Action Chunk Generation

We aim to make action continuation a *native property* of the learned policy. This entails two requirements: **(i) per-step guidance** as discussed in section III-B, where guidance is applied repeatedly across denoising steps, and **(ii) training-inference consistency**.

Accordingly, we first construct a schedule-shaped training path, then derive the induced guided dynamics, and finally reshape the velocity field to eliminate the train-test mismatch.

*1) **Action-noise mixture**:* To incorporate guidance into training, we introduce a horizon-wise continuation vector $\boldsymbol{\omega} \in [0,1]^H$, which encodes the guidance schedule over the chunk horizon, i.e., full guidance near the chunk beginning and a gradual ramp-down toward the end of the horizon.

Algorithm 1 Legato: Training and Inference

Require: policy f_θ; observation o; horizon H; denoising steps N; schedule params (d, r); executed length s

Construct schedule $\boldsymbol{\omega} \in [0,1]^H$ from (d, r)
$\Delta t \leftarrow 1/N, \quad \boldsymbol{\kappa} \leftarrow \boldsymbol{\omega}/\Delta t$
if training **then**
  Sample $t \sim \mathcal{U}(0,1)$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\boldsymbol{\epsilon}_{\text{eff}} \leftarrow \boldsymbol{\omega} \odot \mathbf{A} + (\mathbf{1} - \boldsymbol{\omega}) \odot \boldsymbol{\epsilon}$
  $\mathbf{Y}_t \leftarrow (1-t)\boldsymbol{\epsilon}_{\text{eff}} + t\mathbf{A}$
  $\mathbf{v}_{\text{target}} \leftarrow (1 - \boldsymbol{\kappa} \odot (1-t)) \odot (\mathbf{A} - \boldsymbol{\epsilon})$
  Update θ by $\|f_\theta(\mathbf{Y}_t, o, t, \boldsymbol{\omega}) - \mathbf{v}_{\text{target}}\|_2^2$
else ▷ *Inference*
  Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  if no previous chunk **then**
    $\mathbf{A}_{\text{prev}} \leftarrow \mathbf{0}$
  end if
  $\mathbf{A}_{\text{ref}} \leftarrow \textsc{PadLast}(\mathbf{A}_{\text{prev}}[s{:}H])$
    ▷ *Truncate and pad with the last value to length H*
  $\mathbf{X}_0 \leftarrow \boldsymbol{\epsilon}$
  for $k = 0$ **to** $N-1$ **do**
    $\mathbf{Y}_k \leftarrow (\mathbf{1} - \boldsymbol{\omega}) \odot \mathbf{X}_k + \boldsymbol{\omega} \odot \mathbf{A}_{\text{ref}}$ ▷ *Guiding*
    $\mathbf{X}_{k+1} \leftarrow \mathbf{Y}_k + \Delta t\, f_\theta(\mathbf{Y}_k, o, t_k, \boldsymbol{\omega})$ ▷ *Denoising*
  end for
  return $\hat{\mathbf{A}} \leftarrow \mathbf{X}_N$
end if

Using $\boldsymbol{\omega}$, we define an action-noise mixture

$$\boldsymbol{\epsilon}_{\text{eff}} = (\mathbf{1} - \boldsymbol{\omega}) \odot \boldsymbol{\epsilon} + \boldsymbol{\omega} \odot \mathbf{A}, \tag{6}$$

where $\odot$ denotes element-wise multiplication and $\boldsymbol{\epsilon}_{\text{eff}}$ represents the *effective noise initialization* induced by continuation guidance, interpolating between the action chunk and pure noise in a horizon-wise manner.

Based on this mixture, we construct the interpolation path

$$\mathbf{Y}_t = (1-t)\,\boldsymbol{\epsilon}_{\text{eff}} + t\,\mathbf{A}, \tag{7}$$

which reduces to the standard flow-matching path when $\boldsymbol{\omega} = \mathbf{0}$ and collapses to the action chunk for all t when $\boldsymbol{\omega} = \mathbf{1}$.

The corresponding flow-matching velocity is

$$\mathbf{u}^{\text{FM}}(\mathbf{Y}_t, t) = \mathbf{A} - \boldsymbol{\epsilon}_{\text{eff}} = (\mathbf{1} - \boldsymbol{\omega}) \odot (\mathbf{A} - \boldsymbol{\epsilon}), \tag{8}$$

reflecting a horizon-wise modulation of the FM velocity.

Multimodal persistence and smoothness: Eq. (8) reveals a *schedule-shaped* velocity: the target transport magnitude is modulated by $\boldsymbol{\omega}$. In timesteps with large $\boldsymbol{\omega}_i$ (strong continuation), the effective u_i^{FM} is suppressed as $u_i^{\text{FM}} \propto (1 - \boldsymbol{\omega}_i)$, making the overlap and the ramp region intrinsically less mutable than the none-guidance region during denoising. This discourages frequent switching among competing action modes in highly multimodal tasks.

Moreover, since $\boldsymbol{\omega}$ decreases from 1 along the horizon, the effect of continuation is gradually relaxed through a ramp, yielding a smooth transition from strict guidance to free generation. As a result, Legato reduces chunk-boundary

discontinuities and improves trajectory smoothness.

*2) **Effective dynamics of repeated continuation guidance**:* The velocity construction above specifies the schedule-shaped guidance. At inference time, continuation requires per-step guidance to keep effective. We therefore derive the exact dynamics induced by the per-step guidance.

At each denoising step k, the current noisy action is first guided toward the reference action according to the guidance schedule $\boldsymbol{\omega}$:

$$\mathbf{Y}_k = (\mathbf{1} - \boldsymbol{\omega}) \odot \mathbf{X}_k + \boldsymbol{\omega} \odot \mathbf{A}, \tag{9}$$

where $\mathbf{A}$ denotes the reference action and $\mathbf{X}_k$ is the current noisy action before guidance.

We then perform one denoising update:

$$\mathbf{X}_{k+1} = \mathbf{Y}_k + \Delta t \, f_\theta(\mathbf{Y}_k, t_k), \tag{10}$$

after which the same guidance in eq. (9) is applied again at the next step, as shown in fig. 2.

Eliminating $\mathbf{X}_k$ yields the exact recurrence

$$\mathbf{Y}_{k+1} = \boldsymbol{\omega} \odot \mathbf{A} + (\mathbf{1} - \boldsymbol{\omega}) \odot \mathbf{Y}_k + (\mathbf{1} - \boldsymbol{\omega}) \odot \Delta t \, f_\theta(\mathbf{Y}_k, t_k). \tag{11}$$

Taking the continuous-time limit, this recurrence corresponds to the ordinary differential equation

$$\dot{\mathbf{Y}}(t) = (\mathbf{1} - \boldsymbol{\omega}) \odot f_\theta(\mathbf{Y}(t), t) - \boldsymbol{\kappa} \odot (\mathbf{Y}(t) - \mathbf{A}), \;\; \boldsymbol{\kappa} = \boldsymbol{\omega} / \Delta t. \tag{12}$$

Importantly, eq. (12) is not an approximation: it is the exact continuous-time system whose Euler discretization reproduces repeated continuation guidance.

*3) **Training-inference consistency**:* Having characterized the dynamics induced by per-step guidance, we now turn to the second requirement: training-inference consistency. Standard flow matching supervises the velocity field $\mathbf{u}^{\mathrm{FM}}$, whereas inference with repeated continuation guidance follows the dynamics in eq. (12). To eliminate this mismatch, we require the executed velocity field to coincide with the flow-matching target:

$$(\mathbf{1} - \boldsymbol{\omega}) \odot f_\theta(\mathbf{Y}, t) - \boldsymbol{\kappa} \odot (\mathbf{Y} - \mathbf{A}) = \mathbf{u}^{\mathrm{FM}}(\mathbf{Y}, t). \tag{13}$$

Solving eq. (13) for f_θ yields the *Legato velocity field*

$$f_\theta(\mathbf{Y}, t) = (\mathbf{1} - \boldsymbol{\omega})^{-1} \odot \left[\mathbf{u}^{\mathrm{FM}}(\mathbf{Y}, t) + \boldsymbol{\kappa} \odot (\mathbf{Y} - \mathbf{A}) \right], \tag{14}$$

where the inverse is taken element-wise.

Substituting eq. (7) and eq. (8) into eq. (14), we obtain a closed-form target velocity

$$\mathbf{v}_{\text{target}}(t, \mathbf{A}, \boldsymbol{\epsilon}, \boldsymbol{\omega}) = \big(1 - \boldsymbol{\kappa} \odot (1 - t)\big) \odot (\mathbf{A} - \boldsymbol{\epsilon}), \tag{15}$$

The network is trained by regressing $f_\theta(\mathbf{Y}_t, o, t, \boldsymbol{\omega})$ to $\mathbf{v}_{\text{target}}$. Thus, Legato preserves the geometric direction of standard flow matching while reshaping the velocity magnitude to internalize continuation dynamics.

Inference: At inference time, we use the previously generated (but haven't been executed) chunk as the reference for continuation. We construct a reference action chunk $\mathbf{A}_{\text{ref}}$ from the previous prediction using the alignment procedure as shown in algorithm 1. We then instantiate the guidance term in eq. (12) by setting $\mathbf{A} \leftarrow \mathbf{A}_{\text{ref}}$.

Given a schedule $\boldsymbol{\omega}$, we initialize

$$\mathbf{Y}_0 = \boldsymbol{\omega} \odot \mathbf{A}_{\text{ref}} + (\mathbf{1} - \boldsymbol{\omega}) \odot \boldsymbol{\epsilon}, \;\; \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{16}$$

We integrate eq. (12) forward in time from $t = 0$ to $t = 1$ using the learned velocity field in eq. (14), with the same discretization (number of denoising steps N) as used during training. This enable the strict training-inference alignment.

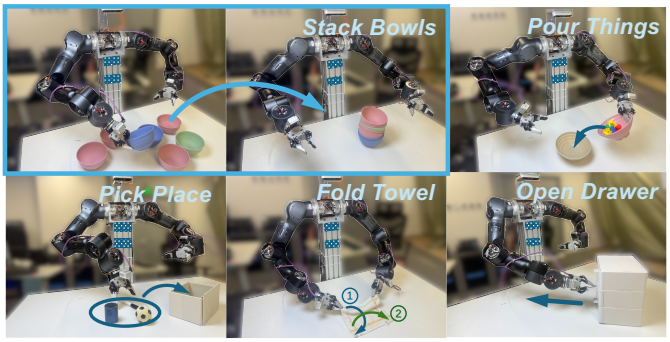

Fig. 5. Real-world evaluation tasks on a dual-arm robot. We consider five manipulation tasks (stack bowls, pour things, pick and place, fold towel and open drawer) covering diverse motion patterns and multimodal choices such as alternative grasp goals and left/right arm selection.

D. Schedule Randomization and Conditioning

In our framework, the continuation schedule over an action chunk of horizon H is fully specified by two scalar parameters: the inference delay d and the ramp length r. Given (d, r), the guidance schedule $\boldsymbol{\omega} \in [0, 1]^H$ is uniquely determined, consisting of a full-guidance prefix of length d followed by a ramp part of length r.

In real-world deployment, effective inference delay varies across hardware platforms, model sizes, and inference optimizations. To account for this variability while enabling flexible control over continuation smoothness, we randomize (d, r) during training, thereby exposing the policy to a diverse family of guidance schedules.

When training with randomized schedules, the policy must be informed of the schedule at inference time. We therefore explicitly condition the action decoder on the schedule. Concretely, if the noisy action $\mathbf{Y}_t \in \mathbb{R}^{H \times D_a}$, where D_a denotes the action dimension, we append the guidance schedule along the feature dimension, resulting in an noisy action of shape $(H, D_a + 1)$. At inference time, adapting to a new continuation regime only requires changing the guidance schedule $\boldsymbol{\omega}$, without retraining the model. Empirically, this schedule conditioning substantially improves robustness across hardware platforms and inference budgets.

IV. Experiments

A. Experimental Setups

*1) **Tasks and Environments**:* We evaluate our method on five real-world manipulation tasks: (i) stack the bowls, (ii) pour

TABLE I

MAIN REAL-WORLD RESULTS COMPARING RTC AND LEGATO ACROSS FIVE TASKS. WE REPORT TASK SCORE (↑), COMPLETION TIME IN SECONDS (↓), AND SMOOTHNESS METRICS (↓): NLDLJ (NEGATIVE LOG DIMENSIONLESS JERK [1, 2]), NSPARC (NEGATIVE LINEAR AND ANGULAR SPECTRAL ARC LENGTH [1, 2]), AND OVERLAP RMSE (ROOT MEAN SQUARED ERROR, $\times 10^3$). VALUES ARE REPORTED AS MEAN ± STANDARD ERROR.

Task	Score ↑		Completion Time (s) ↓		Smoothness ↓					
					NLDLJ ↓		NSPARC ↓		Overlap RMSE ($\times 10^3$) ↓	
	RTC	Legato	RTC	Legato	RTC	Legato	RTC	Legato	RTC	Legato
Bowls	8.68 ± 0.35	**9.08 ± 0.33**	52.88 ± 3.54	**42.66 ± 2.68**	36.00 ± 0.34	**35.86 ± 0.38**	1.82 ± 0.04	**1.63 ± 0.02**	6.83 ± 0.50	**4.58 ± 0.17**
Pour	9.34 ± 0.18	**9.72 ± 0.13**	95.07 ± 2.86	**75.73 ± 1.51**	39.82 ± 0.15	**39.50 ± 0.13**	2.85 ± 0.24	**1.65 ± 0.08**	7.64 ± 0.70	**5.14 ± 0.17**
PickPlace	9.47 ± 0.15	**9.53 ± 0.12**	35.53 ± 1.24	**30.37 ± 0.65**	34.42 ± 0.18	**34.34 ± 0.14**	2.10 ± 0.08	**1.89 ± 0.05**	10.17 ± 0.66	**5.98 ± 0.40**
Drawer	9.20 ± 0.16	**9.50 ± 0.13**	25.97 ± 0.74	**21.80 ± 0.72**	32.73 ± 0.13	**28.55 ± 0.26**	2.24 ± 0.05	**1.99 ± 0.08**	12.11 ± 0.66	**11.74 ± 0.55**
Towel	7.33 ± 0.62	**8.17 ± 0.56**	25.93 ± 0.98	**20.00 ± 0.78**	32.79 ± 0.20	**32.43 ± 0.24**	2.17 ± 0.07	**1.97 ± 0.05**	11.28 ± 0.55	**6.22 ± 0.66**

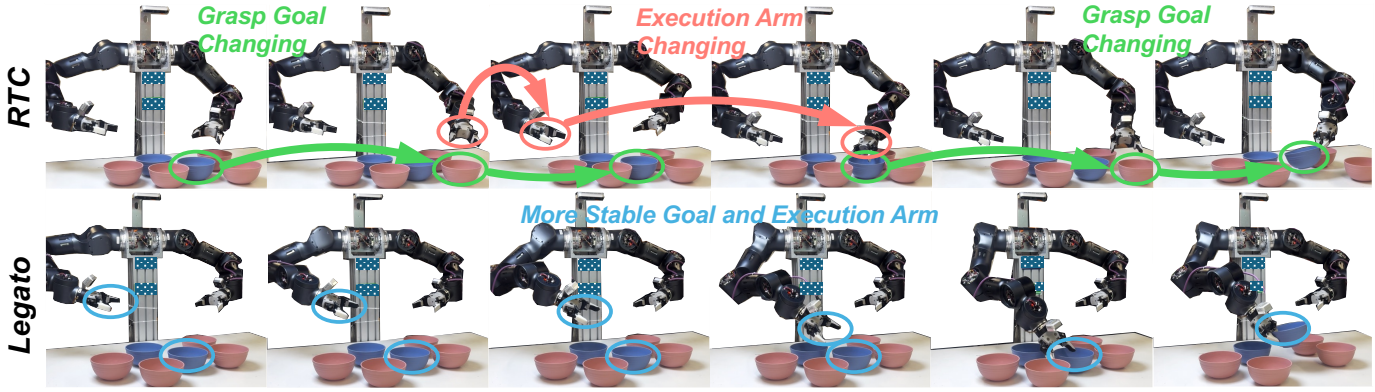

Fig. 6. Legato suppresses spurious multimodal switching across chunk boundaries. In a representative bowl-stacking rollout, RTC alternates (arrow) between competing grasp goals (green circle) and execution arms (red circle) over successive chunks, producing visibly hesitant corrections. Legato preserves a consistent grasp goal and arm choice (blue circle), leading to steadier progress.

things into the bowl, (iii) put all the items into the box, (iv) fold the towel, and (v) open the drawer, as shown in fig. 5. These tasks jointly test different action patterns (e.g., rotation- or translation-dominant motions) and multimodal action selection (e.g., multiple valid grasp goals or the choice of different arms for execution). All tasks are evaluated with a fixed time cutoff of 120 s. Details are provided in the appendix.

*2) **Evaluation Metrics**:* The following evaluation metrics are used to assess real-world experimental performance.

Task completion score. Each rollout is assigned a task-specific completion score based on task progress and failure cases (e.g., partial success, object drops, or incorrect actions). Higher scores indicate better task completion.

Task completion time. We measure the total time required to complete each task. This metric reflects the execution efficiency of the policy, capturing delays caused by hesitation or spurious action switching during real-world execution.

Trajectory smoothness metrics. Following prior work on action smoothness, we evaluate smoothness on the model output commands rather than robot executed states. This decouples model behavior from low-level controller performance. Specifically, we report three smoothness-related metrics that capture complementary aspects of trajectory quality [1, 2]:

- **Negative SPARC (NSPARC)**, where SPARC [1, 2] (Linear and Angular Spectral Arc Length) measures the smoothness of the velocity profile in the frequency domain over the *entire trajectory*. Lower values of NSPARC indicate smoother global speed modulation with reduced high-frequency fluctuations.
- **Negative LDLJ (NLDLJ)**, where LDLJ [1, 2] (Log Dimensionless Jerk) quantifies high-order geometric smoothness by integrating squared jerk over the *entire trajectory*. Lower NLDLJ correspond to reduced overall jerk energy and smoother motion at a global level.
- **Chunk-overlap RMSE**, computed over the overlapping delay segment between consecutive action chunks, which evaluates *local trajectory continuity* at chunk connections rather than global smoothness.

Except for the task completion score, lower values indicate better performance for all metrics. Details of all metrics are provided in the appendix.

*3) **Models and Training Protocol**:* We compare the RTC baseline and our proposed Legato method under a strictly controlled setting. Both methods are initialized from the same $\pi_{0.5}$ pretrained checkpoint, trained on identical task datasets, and optimized using the same training hyperparameters and number of training steps.

B. Main Results

In this section, we report real-world evaluation results of Legato and RTC across five manipulation tasks executed on physical robotic platforms. As summarized in table I, Legato consistently outperforms RTC across all evaluated tasks.

TABLE II
COMPARISON OF TRAINING-TIME RTC AND LEGATO. THE GUIDANCE CONFIGURATION OF LEGATO IS d=8, s=30, r=22. VALUES ARE REPORTED AS MEAN ± STANDARD ERROR.

Metric	Training-time RTC	Legato
Score ↑	9.46 ± 0.16	**9.72 ± 0.13**
Completion Time (s) ↓	81.73 ± 1.12	**75.73 ± 1.51**
NSPARC ↓	2.46 ± 0.14	**1.65 ± 0.08**
NLDLJ ↓	39.95 ± 0.13	**39.50 ± 0.13**

*1) **Task efficiency**:* Legato consistently achieves shorter task completion time than RTC across all tasks. As analyzed in section III-C, the schedule-shaped velocity reweighting increases the difficulty of switching between competing action modes, effectively suppressing frequent multimodal oscillations during execution.

Empirically, this leads to more decisive action generation with reduced hesitation before execution, thereby shortening overall task duration. The effect is particularly pronounced in the bowl-stacking task, where multiple visually similar bowls induce a large number of plausible action modes, as shown in fig. 6. In such settings, RTC often alternates between competing strategies, while Legato maintains consistent mode selection and completes the task more efficiently.

*2) **Trajectory smoothness**:* Legato also demonstrates clear advantages in trajectory smoothness compared to RTC. With the exception of NLDLJ, all smoothness-related metrics show statistically significant improvements in favor of Legato.

Specifically, Legato consistently achieves lower NSPARC values across all tasks. This result indicates that Legato produces commands with reduced high-frequency velocity fluctuations and more regular speed modulation. Such improvements correspond to smoother and more visually coherent motions observed during real-world execution, as shown in the trajectories in fig. 1.

In addition, Legato substantially reduces the chunk-overlap RMSE across tasks. The observed improvements indicate that Legato generates more coherent chunk-to-chunk transitions, leading to improved continuity at action boundaries and smoother chunk-to-chunk stitching.

In contrast, improvements in NLDLJ do not consistently reach statistical significance across all tasks. NLDLJ measures high-order geometric smoothness by integrating squared jerk over the entire trajectory and is therefore dominated by motion segments outside the chunk overlap regions. Importantly, NLDLJ does not degrade under Legato compared to RTC, indicating that while trajectory continuity is improved at chunk boundaries, the remaining portions of the trajectory do not exhibit degraded smoothness.

*3) **Task success**:* Finally, Legato exceeds the task completion scores achieved by RTC. This confirms that the observed improvements in execution efficiency and trajectory smoothness do not come at the cost of task success, but instead translate into more reliable and effective real-world manipulation performance.

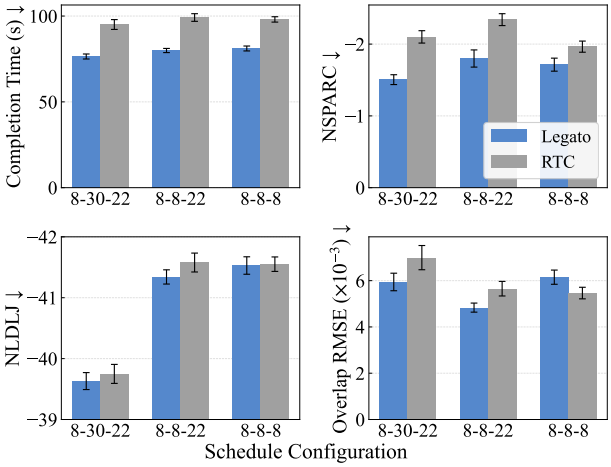

Fig. 7. Schedule ablation reveals a controllable trade-off between local overlap consistency and smoothness. Across schedule configurations (d, s, r), Legato outperforms RTC on completion time and smoothness. Decreasing stride strengthens overlap coupling but can increase high-frequency content; shortening the ramp partially recovers frequency-domain smoothness at the cost of weaker overlap alignment.

C. Comparison with Training-Time RTC

We compare Legato with the recently proposed training-time RTC [9] on the *pour* task, which also introduces continuation during training by constraining overlapping action segments. We implement training-time RTC following the original formulation and compare it against Legato under the same experimental settings. As shown in table II, Legato achieves higher task scores, shorter completion times, and improved smoothness metrics compared to training-time RTC.

To contextualize this comparison, when the ramp length in our guidance schedule is set to zero, the schedule reduces to a hard overlap constraint that is similar in form to training-time RTC. However, this similarity is limited to the constraint shape: the two approaches differ fundamentally in how continuation is incorporated into the learned policy. Training-time RTC treats continuation as an external constraint via hard prefix conditioning while leaving the underlying flow dynamics unchanged. In contrast, Legato reshapes the learned flow dynamics to match the effective denoising behavior induced by repeated, schedule-shaped guidance, so continuation becomes a native property of the policy dynamics.

Overall, these results suggest that reshaping the policy dynamics (rather than enforcing hard overlap constraints alone) is important for effective chunk continuation, and that using a non-zero ramp further enables smoother transitions between consecutive action chunks.

D. Ablation Studies

In this section, we conduct a comprehensive set of ablation studies to analyze the applicability and robustness of the proposed method. Specifically, we examine: (i) the effect of different guidance schedule settings at inference time, (ii) the role of the condition row used in our policy, and (iii) the performance of Legato across different VLA models.

TABLE III

ABLATION STUDY ON ROBUSTNESS TO INFERENCE DELAY. WE VARY THE INFERENCE DELAY d WITH A FIXED EXECUTION STRIDE s, WHERE THE RAMP LENGTH r CHANGES ACCORDINGLY DUE TO THE SCHEDULE CONSTRAINT. VALUES ARE REPORTED AS MEAN ± STANDARD ERROR.

(d, s, r)	Method	NSPARC ↓	Overlap RMSE ↓
(10,30,20)	RTC	2.03 ± 0.08	9.23 ± 0.75
	Legato	**1.68 ± 0.09**	**7.00 ± 0.50**
(8,30,22)	RTC	2.10 ± 0.09	7.00 ± 0.52
	Legato	**1.50 ± 0.07**	**5.94 ± 0.38**
(6,30,24)	RTC	2.03 ± 0.08	9.23 ± 0.75
	Legato	**1.38 ± 0.05**	**5.44 ± 0.31**

TABLE IV

ABLATION STUDY ON THE EFFECT OF THE CONDITION ROW UNDER DIFFERENT GUIDANCE CONFIGURATIONS. WE VARY THE INFERENCE DELAY d AND RAMP LENGTH r TO CONSTRUCT DIFFERENT SCHEDULES. VALUES ARE REPORTED AS MEAN ± STANDARD ERROR.

(d, s, r)	Method	NSPARC ↓	Overlap RMSE ↓
(10,30,20)	w/o cond	**1.64 ± 0.07**	7.88 ± 0.70
	w/ cond	1.68 ± 0.09	**7.00 ± 0.50**
(8,30,22)	w/o cond	1.52 ± 0.09	7.21 ± 0.68
	w/ cond	**1.50 ± 0.07**	**5.94 ± 0.38**
(6,30,24)	w/o cond	1.49 ± 0.10	6.40 ± 0.52
	w/ cond	**1.38 ± 0.05**	**5.44 ± 0.31**

TABLE V

ABLATION RESULTS ON THE π_0 AND GR00T N1.6 MODEL COMPARING RTC AND LEGATO UNDER THE SAME GUIDANCE CONFIGURATION (d=8, s=30, r=22). VALUES ARE REPORTED AS MEAN ± STANDARD ERROR.

Metric	π_0 + RTC	π_0 + Legato
Completion Time ↓	92.93 ± 1.90	**88.30 ± 1.29**
NSPARC ↓	2.00 ± 0.09	**1.83 ± 0.08**
NLDLJ ↓	40.48 ± 0.21	**40.27 ± 0.09**
Overlap RMSE ↓	8.63 ± 0.65	**7.50 ± 0.49**
Metric	Gr00t + RTC	Gr00t + Legato
Completion Time ↓	91.60 ± 2.30	**84.90 ± 1.30**
NSPARC ↓	2.99 ± 0.17	**2.63 ± 0.06**
NLDLJ ↓	39.91 ± 0.07	**39.62 ± 0.12**
Overlap RMSE ↓	9.43 ± 0.46	**8.75 ± 0.36**

*1) **Varying the execution stride s:*** RTC recommends setting the execution stride s to at least half of the action chunk length. However, since s directly determines the effective inference frequency, a larger stride inevitably reduces the model's responsiveness. This reveals an inherent trade-off between inference efficiency and control reactivity, motivating a detailed ablation over guidance schedule configurations.

When the execution stride s becomes smaller than half of the chunk length, the ramp segment may extend beyond the immediate next chunk. To avoid that, we shorten the ramp segment as s decreases, ensuring that the ramp always remains confined within the next chunk. We evaluate several guidance schedule configurations on the *pour* task, as illustrated in fig. 7.

Our findings can be summarized as follows:

*a) **Legato consistently outperforms RTC on almost all metrics:*** The only exception is the overlap RMSE in the $d = s = r = 8$ setting, which is discussed in the appendix.

*b) **Reducing the execution stride s improves chunk-to-chunk consistency but can degrade global smoothness:*** Under the constraint $r+s+d = H$, a smaller s implies a larger ramp length r, which improves chunk-to-chunk continuity, as reflected by lower overlap RMSE. At the same time, smaller strides lead to more frequent overlap regions, causing high-frequency components to accumulate and resulting in degraded whole-trajectory smoothness metrics.

*c) **Shortening the ramp while keeping s small improves frequency-domain smoothness at the expense of overlap consistency:*** When s remains small but the ramp length is shortened, NSPARC improves, indicating smoother frequency-domain behavior. However, this reduces overlap consistency, reflecting a weaker coupling between adjacent chunks.

Overall, these results demonstrate that the execution stride s and ramp length jointly control a fundamental trade-off between local chunk connection quality and global frequency-domain smoothness. By adjusting their relative proportions, Legato enables flexible control over trajectory smoothness.

*2) **Varying the inference delay d:*** In addition to the execution stride, the inference delay d also plays an important role in shaping trajectory smoothness. To isolate its effect, we fix the execution stride s and vary the delay length d, conducting evaluations on the *pour* task. The quantitative results are summarized in table III.

Across all evaluated metrics, Legato consistently outperforms RTC, demonstrating the robustness of the proposed method to variations in inference latency. When analyzing Legato specifically, we find that reducing the delay length decreases the size of the overlap region while simultaneously increasing the relative length of the ramp segment. This leads to improved chunk-to-chunk continuity and smoother execution, as reflected by better overlap consistency and frequency-domain smoothness metrics.

Overall, these results indicate that both execution stride s and inference delay d provide effective control knobs for shaping smoothness properties of generated trajectories. Legato can flexibly adapt to different schedule configurations while consistently maintaining superior performance over RTC.

*3) **Condition Row:*** To evaluate whether the condition row is useful, we conduct an ablation study in which the guidance schedule is no longer provided as an explicit condition.

We perform this ablation on the *pour* task, and report the results in table IV. As shown, removing the condition row leads to a degradation in performance, particularly in trajectory smoothness and execution stability. This suggests that explicitly providing the guidance schedule helps the model disambiguate different continuation regimes induced by varying (d, r) pairs, and enables more reliable adaptation to dynamic inference conditions.

*4) **Different Models:*** In the main results table I, we evaluate our method on the $\pi_{0.5}$ model. To evaluate whether the proposed method generalizes across different VLA models, we further conduct experiments on the π_0 and Gr00t N1.6 model. We select the representative task *pour things*. As shown in

TABLE VI
COMPARISON OF TE AND LEGATO ON THE POUR TASK.

Method	Score↑	Time (s)↓	NSPARC↓
TE	9.17 ± 0.16	96.87 ± 2.79	2.96 ± 0.23
Legato	**9.67 ± 0.10**	**76.43 ± 1.46**	**1.50 ± 0.07**

TABLE VII
NSPARC METRIC RESULTS UNDER VARYING INFERENCE N (TRAINED FOR N=5 ON POUR TASK) OF LEGATO.

	N=1	N=3	N=10	N=15
w/o cond. row	1.98 ± 0.17	1.85 ± 0.07	1.77 ± 0.18	1.97 ± 0.09
w/ cond. row	*unstable motion*	2.65 ± 0.11	1.82 ± 0.16	2.11 ± 0.13

table V, Legato outperforms RTC on the π_0 and Gr00t N1.6 model on the pour task.

These results demonstrate that the proposed method is not tied to a specific policy backbone or training configuration, and can be effectively transferred across different flow-based VLA models, highlighting its robustness and model generality.

*5) **Different Smoothing Methods**:* As diffusion-based methods are not directly comparable, we compare with the trajectory smoothing baseline Temporal Ensembling (TE) [43]. This method maintains a buffer of predicted chunks and executes averaged actions for each timestep. As shown in table VI, Legato outperforms TE in both execution time and continuation quality, benefiting from its native continuation mechanism.

*6) **Sensitivity to denoising steps**:* Training velocity implicitly depends on the denoising step count N, while we additionally evaluate generalization under different inference-time denoising schedules. To improve robustness across varying N, we apply a heuristic rescaling $\omega_i' = \text{clamp}(1-(1-\omega_i)\cdot(N_{\text{train}}/N_{\text{infer}}), 0, 1)$ at inference time. Although this rescaling introduces a mild training-inference mismatch, since the velocity field is trained under a fixed N_{train}, we find that it substantially reduces sensitivity to the inference denoising step count in practice. Without the condition row, Legato even remains effective under single-step denoising, as shown in table VII.

E. Simulation Results on Kinetix

We test Legato in Kinetix, using identical settings to Training-time RTC. As shown in Figure 8 (left): Legato outperforms all methods when delay>0. In Kinetix, the policy is a based on a UNet, shown the generalization of Legato across different flow-based policies.

We also ablate schedule randomization during training in Kinetix. As shown in Figure 8 (right), fixing d during training degrades performance at other d values during evaluation, highlighting the importance of delay randomization. Notably, when the randomized delay range does not include d=0, the policy becomes severely out-of-distribution at the first denoising step without guidance, leading to near-complete failure during evaluation.

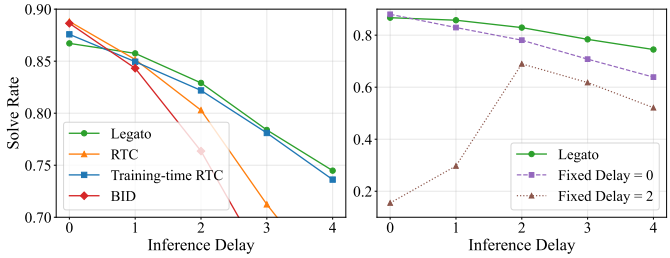

Fig. 8. Left: Legato vs. RTC, Training-time RTC, and BID in Kinetix. Right: Ablation result on the schedule randomization.

F. Deployment guidelines

We recommend the following hyper-parameters for deployment: $d = \max(\text{recent delays})$, $s = 0.5H$, $r = H - d - s$. In practice, following RTC, we maintain a short history of recent inference delays and use the maximum observed delay as the deployment-time d. For variable-N inference, we recommend using the w/o-condition-row variant.

Additionally, we find that finetuning from a strong base policy leads to more stable convergence and reduces the tendency of excessive guidance training to impair reactivity. Therefore, we recommend initializing Legato from a strong pretrained base policy.

V. CONCLUSION

In this work, we propose Legato, a training-time continuation method for action-chunked flow-based VLA policies. Legato reshapes the learned flow dynamics to align training and inference under schedule-shaped, per-step continuation, making chunk continuation a native property of the policy. This design improves trajectory smoothness and reduces spurious multimodal switching at chunk boundaries, leading to smoother action and more consistent action modes, less hesitation, and shorter task completion time. By conditioning on randomized schedules, a single policy can adapt to different inference delays and flexibly control trajectory smoothness.

In the current formulation, the denoise step is specified at training time, limiting the ability to adjust it during inference. Future work could investigate more flexible native continuation schemes with consistent training and inference dynamics.

ACKNOWLEDGMENTS

This research was conducted with the support of the Shanghai Qi Zhi Institute & Spirit AI Innovation Program and the Tsinghua University Dushi Program. Funding and support for this work were also provided by the Tsinghua University - Keystone Electrical (Zhejiang) Co.,Ltd Joint Research Center for Embodied Multimodal Artificial Intelligence (JCEMAI). Additionally, we would like to extend our thanks to the Xiongan AI Institute.

REFERENCES

[1] Nadun Ranawaka Arachchige, Zhenyang Chen, Wonsuhk Jung, Woo Chul Shin, Rohan Bansal, Pierre Barroso, Yu Hang He, Yingyang Celine Lin, Benjamin Joffe,

Shreyas Kousik, et al. Sail: Faster-than-demonstration execution of imitation learning policies. *arXiv preprint arXiv:2506.11948*, 2025.

[2] Sivakumar Balasubramanian, Alejandro Melendez-Calderon, Agnès Roby-Brami, and Etienne Burdet. On the analysis of movement smoothness. *Journal of NeuroEngineering and Rehabilitation*, 12, 2015.

[3] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025.

[4] Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024.

[5] 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.

[6] 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.

[7] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

[8] Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

[9] Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025.

[10] Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5144–5151. IEEE, 2024.

[11] Chilam Cheang, Sijin Chen, Zhongren Cui, Yingdong Hu, Liqun Huang, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Xiao Ma, et al. Gr-3 technical report. *arXiv preprint arXiv:2507.15493*, 2025.

[12] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

[13] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[14] 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.

[15] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[17] Sigmund H Høeg, Yilun Du, and Olav Egeland. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models. *arXiv preprint arXiv:2406.04806*, 2024.

[18] Chanhyuk Jung, Dasom Ahn, Sangwon Kim, In-su Jang, Kwang-Ju Kim, Sungkeun Yoo, and Byoung Chul Ko. Rolling diffusion policy for robotic action prediction: Enhancing efficiency and temporal awareness. In *ICRA 2025 Workshop on Foundation Models and Neuro-Symbolic AI for Robotics*, 2025.

[19] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[20] Lucy Lai, Ann Zixiang Huang, and Samuel J Gershman. Action chunking as policy compression. *PsyArXiv*, 2022.

[21] Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Tian Nian, Liuao Pei, Shunbo Zhou, Xiaokang Yang, Jiangmiao Pang, et al. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072*, 2025.

[22] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *ArXiv*, abs/2505.11917, 2025.

[23] Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. *arXiv preprint arXiv:2511.04555*, 2025.

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

[25] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[26] Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via closed-loop resampling. *arXiv preprint arXiv:2408.17355*, 2024.

[27] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcar-

cel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[28] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[29] Ashwini Pokle, Matthew Muckley, Ricky T. Q. Chen, and Brian Karrer. Training-free linear image inverses via flows. *Trans. Mach. Learn. Res.*, 2024, 2023.

[30] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, et al. Eo-1: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint arXiv:2508.21112*, 2025.

[31] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[32] Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023.

[33] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[34] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[35] Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. *ArXiv*, abs/2507.01016, 2025.

[36] Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. dvla: Diffusion vision-language-action model with multimodal chain-of-thought. *arXiv preprint arXiv:2509.25681*, 2025.

[37] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[38] Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. Llada-vla: Vision language diffusion action models. *arXiv preprint arXiv:2509.06932*, 2025.

[39] Bin Yu, Shijie Lian, Xiaopeng Lin, Yuliang Wei, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Xinming Wang, Bailing Wang, Cong Huang, et al. Twinbrainvla: Unleashing the potential of generalist vlms for embodied tasks via asymmetric mixture-of-transformers. *arXiv preprint arXiv:2601.14133*, 2026.

[40] Hang Yu, Juntu Zhao, Yufeng Liu, Kaiyu Li, Cheng Ma, Di Zhang, Yingdong Hu, Guang Chen, Junyuan Xie, Junliang Guo, et al. Point what you mean: Visually grounded instruction policy. *arXiv preprint arXiv:2512.18933*, 2025.

[41] Juntu Zhao, Wenbo Lu, Di Zhang, Yufeng Liu, Yushen Liang, Tianluo Zhang, Yifeng Cao, Junyuan Xie, Yingdong Hu, Shengjie Wang, et al. Do you need proprioceptive states in visuomotor policies? *arXiv preprint arXiv:2509.18644*, 2025.

[42] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1702–1713, 2025.

[43] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[44] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[45] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[46] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.