APAS-VLA: ADAPTIVE PHYSICAL ACTION SCALING FOR FASTER AND RELIABLE ROBOT EXECUTION

Anonymous authors
Paper under double-blind review

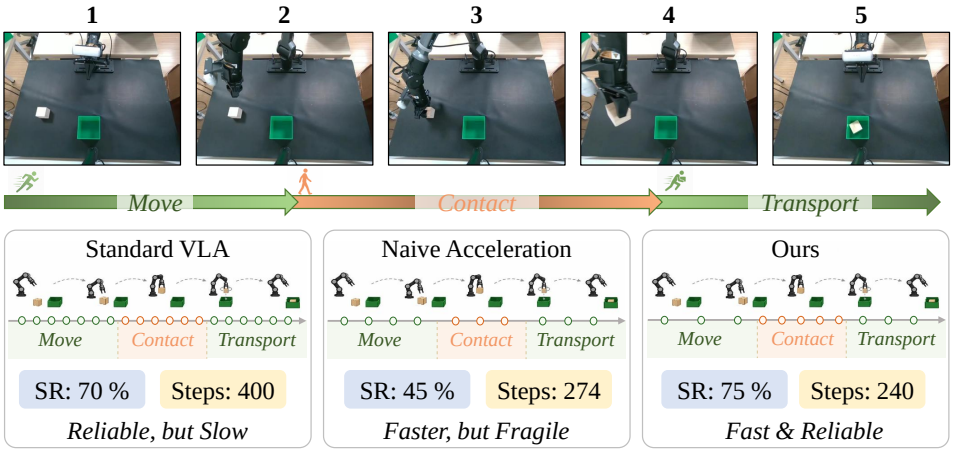

Figure 1: **Execution strategies on a real-robot task.** APAS takes larger strides during move and transport while retaining dense coverage around contact.

ABSTRACT

Physical execution of standard Vision-language-action (VLA) is limited by the robot demonstration speed, because there is a one-to-one mapping between action token and future demonstration step. To break this implicit coupling, we introduce APAS, an adaptive action-generation framework that allows each action token to represent a different amount of progress along the demonstration trajectory. APAS combines a lightweight Token-Speed Predictor, which predicts an observation-dependent token-speed profile, with a Profile-Conditioned Action Expert, which directly generates actions under the predicted profile. To adapt actions with different precision requirements, we develop an offline teacher to assign larger token speeds for stable motion and smaller token speeds for precise interactions, and then distill this knowledge into a lightweight predictor for fast deployment. On LIBERO, APAS retains near-base success (96.25% versus 96.85%) while reducing successful-episode execution from 148 to 86 control steps. On CALVIN, it improves average completed sequence length from 3.887 to 4.096 while reducing successful-subtask execution from 88.96 to 55.89 steps. On the real robot, APAS reduces successful-trial completion time by 32.3–49.0% while matching or exceeding Vanilla's success rate across three evaluated conditions.

1 INTRODUCTION

Vision-language-action (VLA) policies have emerged as a promising paradigm for general-purpose robot control. By combining vision-language representations with action generation, they can follow natural-language instructions and perform diverse tasks across different environments (Zitkovich et al., 2023; Kim et al., 2025b; Black et al., 2025b;a). However, their physical execution remains inefficient even when action generation itself is sufficiently fast (Arachchige et al., 2025; Kim et al., 2026). A policy may produce an action chunk in real time, but the robot can still require considerable *physical* control steps to complete a task.

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

This inefficiency is largely inherited from imitation-learning data (Arachchige et al., 2025; Kim et al., 2026), since robot demonstrations are commonly collected by teleoperation at cautious and slow speeds (Guo et al., 2025; Arachchige et al., 2025; Kim et al., 2026). In chunk-based policies such as ACT (Zhao et al., 2023), Diffusion Policy (Chi et al., 2023), Octo (Ghosh et al., 2024), and flow-based VLAs (Black et al., 2025b;a), each action token is typically aligned with one consecutive future demonstration step. The horizon therefore determines not only how many actions are generated, but also how much of the future demonstration trajectory the chunk can cover. E.g., a 10-token chunk is usually trained to represent the next 10 control steps, though the same budget could cover a longer segment during stable, low-precision motion.

A natural way to accelerate execution is to resample the predicted trajectory using fewer actions (Guo et al., 2025). However, naive horizon resampling reduces the number of executed actions but does not expand the future trajectory covered by each policy query, leaving the number of queries largely unchanged. One may also generate a longer action chunk and then execute it sparsely. Since action-chunk policies are sensitive to the horizon used during training (Jing et al., 2026), extrapolating beyond the task-specific adaptation horizon can degrade reliability. Thus, neither strategy is satisfactory: native-horizon resampling leaves the number of policy queries largely unchanged, while horizon extrapolation can degrade reliability. More importantly, our analysis indicates the appropriate amount of future-trajectory coverage varies across tasks and execution phases (see Fig. 3 later). Fig. 1 shows conservative execution with dense action coverage in standard VLAs, whereas naive acceleration reduces physical steps at the cost of reliable interaction.

To achieve a trade-off between speed and reliability, we introduce APAS (Adaptive Physical Action Scaling), an adaptive temporal-generation framework that lets each action token represent an observation-dependent future trajectory. We formalize this concept as *token speed* and the cumulative sums of token speeds specify where the action tokens lie along the future trajectory. This decouples the action-token budget from the future coverage of a fixed-length action chunk, allowing larger token speeds during stable motion and denser coverage around precision-sensitive interactions. APAS comprises a lightweight Token-Speed Predictor (τ-Predictor), which estimates a token-speed profile from frozen vision-language features, and a Profile-Conditioned Action Expert (τ-Action Expert), which directly generates actions rather than applying post-hoc resampling. To provide clear supervision for temporal action tokens, we establish an offline Temporal Teacher that combines generation stability and local motion complexity to assign smaller token speeds to uncertain regions and larger token speeds to stable motion. Our extensive experiments show that APAS reduces physical control steps by 41.7% on LIBERO Liu et al. (2023) and 37.2% on CALVIN Mees et al. (2022) while maintaining or improving task performance, with similar gains when applied to X-VLA Zheng et al. (2026). On the real robot, APAS further reduces successful-trial completion time by 32.3–49.0% while matching or exceeding Vanilla's success rate across three evaluated conditions. These results demonstrate that adaptive physical action scaling can accelerate robot execution without the reliability degradation of uniform acceleration.

2 Related Work

Vision-Language-Action Policies. Vision-language-action (VLA) models extend the representation capability of vision-language models to robot control, with representative works including RT-1, RT-2, and OpenVLA (Brohan et al., 2023; Zitkovich et al., 2023; Kim et al., 2025b). Meanwhile, methods such as ACT and Diffusion Policy show the effectiveness of action chunking for predicting multiple consecutive future actions in a single prediction (Zhao et al., 2023; Chi et al., 2023). Recent architectures such as π_0, $\pi_{0.5}$, and X-VLA further introduce flow matching into VLA action generation while retaining a chunk-based prediction paradigm (Black et al., 2025b;a; Zheng et al., 2026). APAS builds on flow-matching VLAs to break the fixed temporal mapping in action representations.

Efficient VLA Inference. Existing work improves VLA computational efficiency from several aspects, including model architecture, visual feature processing, action generation, and inference strategies. Representative approaches include efficient action tokenization (Pertsch et al., 2025), parallel decoding (Kim et al., 2025a), token merging or pruning (Bolya et al., 2023; Wang et al., 2026), and quantization (Kim et al., 2025b). Other approaches use early exiting (Yue et al., 2024), token or feature caching (Xu et al., 2025; Ji et al., 2026), and diffusion distillation (Wang et al.,

2025). These works mainly focus on reducing model-side inference latency. Instead, this paper focuses on optimizing the execution steps by adaptively expanding the trajectory coverage.

Faster-than-Demonstration Execution. Existing faster-than-demonstration methods can be broadly divided into execution and training-side approaches. Classical time-optimal path parameterization (Bobrow et al., 1985; Shin & McKay, 1985; Pham & Pham, 2018), and recent methods such as SAIL and RACE, belong to the execution-side category (Verscheure et al., 2009; Arachchige et al., 2025; Kim et al., 2026). Their main acceleration operates at the execution stage after policy prediction. On the other hand, training-side methods such as DemoSpeedup (Guo et al., 2025) and ISR (Yang et al., 2026) resample demonstrations before policy training, implicitly encoding temporal adaptation into the overall policy. These methods either apply post-hoc trajectory scaling or commit to a fixed temporal mapping offline. In contrast, APAS operates at generation time, by introducing a decoupled architecture that separates temporal-resolution selection from action generation and enables observation-dependent adaptive trajectory coverage

3 PRELIMINARIES

We consider flow-matching VLAs with action chunking. At timestep t, the policy receives the current observation $\mathbf{o}_t = (\mathbf{I}_t, \ell_t, \mathbf{q}_t)$, where $\mathbf{I}_t$, ℓ_t, and $\mathbf{q}_t$ denote the visual observation, language instruction, and robot state, respectively. The vision-language backbone produces a context representation $\mathbf{z}_t = f_\psi(\mathbf{I}_t, \ell_t)$. Conditioned on $\mathbf{z}_t$ and $\mathbf{q}_t$, the action expert parameterizes a conditional vector field $\mathbf{v}_\theta$, whose integration transforms Gaussian noise into an H-token action chunk $\mathbf{A}_t = [\mathbf{a}_t^1, \ldots, \mathbf{a}_t^H] \in \mathbb{R}^{H \times d_a}$, where d_a is the action dimension (Black et al., 2025b). Such action-chunk policies are commonly trained by imitation learning on demonstrations (Zhao et al., 2023). At timestep t, the standard target is the consecutive future chunk $\mathbf{A}_t^{\text{demo}} = [\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H-1}]$, so the k-th action token is supervised by the k-th consecutive future action. Following conditional flow matching, the target is interpolated with Gaussian noise as $\mathbf{A}_t^\lambda = \lambda \mathbf{A}_t^{\text{demo}} + (1-\lambda)\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\lambda \in [0, 1]$.

In this work, we use *synchronous action-chunk execution*, where each policy query finishes before its returned chunk begins execution. Under this protocol, task completion time can be approximated by policy inference plus physical execution (Black et al., 2025c; Arachchige et al., 2025):

$$T_{\text{task}} \approx Q \cdot T_{\text{inf}} + \frac{K}{f_c}, \tag{1}$$

where Q is the number of inference queries, T_{inf} is the inference latency per query, K is the total number of executed control steps, and f_c is the robot control frequency. Since we do not target T_{inf} and f_c is fixed, minimizing T_{task} requires smaller Q and K: reducing Q requires each action chunk to cover a longer future trajectory segment per query, while reducing K requires each token to represent a larger trajectory stride. This raises two questions: 1) Can generators with fixed action-token budget cover a longer future trajectory reliably? 2) Should every observation use the same trajectory coverage? We investigate these questions in the following.

4 EMPIRICAL ANALYSIS

4.1 NAIVE TEMPORAL ACCELERATION IS INSUFFICIENT

Horizon Resampling. A direct approach is to temporally resample the trajectory produced by the original policy (Guo et al., 2025). With a resampling factor of N, we keep the action expert with H-token horizon and compress the trajectory to H/N executable actions. The task time becomes $T_{\text{naive}} \approx Q \cdot T_{\text{inf}} + K/(N \cdot f_c)$ and the second term decreases with N, whereas the first inference term remains unchanged. Since the H/N resampled actions still traverse the same segment predicted by the H-token chunk, it requires a similar number of inference calls, as illustrated in Appendix A.1. Beyond this inference bottleneck, the temporal subsampling itself introduces information loss analogous to pooling in neural networks. By discarding intermediate waypoints, it risks degrading the reliability of contact-sensitive or precision-critical operations (Guo et al., 2025; Arachchige et al., 2025).

Horizon Extrapolation. For the first term in Eq.(1), reducing Q requires an inference to cover a longer future trajectory segment. Expanded-Horizon Resampling asks the frozen action expert to generate $N \times H$ action tokens and uniformly resamples the resulting trajectory into H executed actions. Because downstream adaptation uses a fixed horizon H, inference-time expansion introduces a horizon mismatch with the task-specific training configuration (Jing et al., 2026). To validate this, we use the frozen $\pi_{0.5}$ action expert on Task 0 from LIBERO-Spatial/Object/Goal/10, set $H = 10$, $N = 1 - 3$. As shown in Fig. 2, success decreases across all four tasks as the generation horizon expands. Thus, neither discarding generated actions nor extrapolating to a longer horizon provides reliable acceleration for physical execution. Next, we answer whether a globally fixed span is sufficient in terms of Inter/intra-task variation.

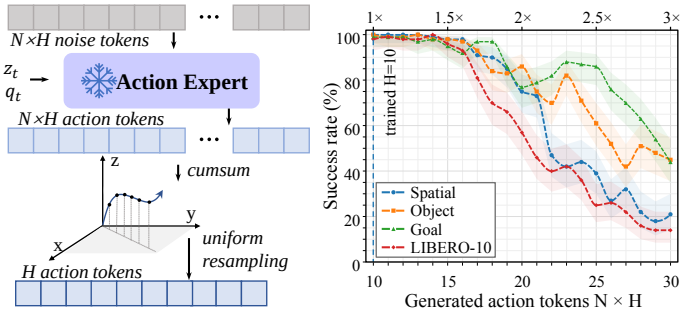

Figure 2: **Expanded-Horizon Resampling.** Expanding the generation horizon beyond the downstream $H = 10$ setting reduces task success.

4.2 Trajectory Coverage Should Adapt to Context

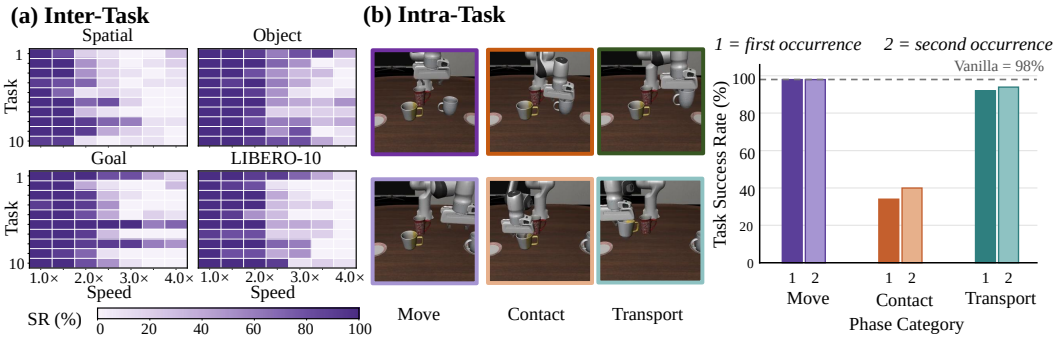

Figure 3: **Context-dependent acceleration tolerance. (a)** Fixed $1-4\times$ rates reveal task-dependent sensitivity across 40 LIBERO tasks. **(b)** $3\times$ acceleration is better tolerated during Move and Transport than Contact.

Inter-Task Variation. First, we keep a fixed temporal rate over all 40 LIBERO tasks. As shown in Fig. 3(a), at $1.5\times$, all four suites remain reliable with over 90% success. The profiles begin to diverge from $2\times$ as Spatial falls below 50% while the rest stays above 80%. The heatmap shows that different tasks have different sensitivity to acceleration. Thus, a globally fixed trajectory coverage cannot guarantee success across different tasks.

Intra-Task Variation. We consider one LIBERO-10 manipulation containing two repeated move-contact-transport sequences in Fig. 3(b), apply the same $3\times$ temporal rate to one phase at a time, and keep every other phase at $1\times$. Compared to the vanilla setting at 98% success, accelerating move and transport retains over 90% success. In contrast, the two contact occurrences reduce success to at most 40%. This explains that coarse motion and contact-sensitive interaction should be treated differently regarding the acceleration ratios.

Thus, temporal coverage varies not only across tasks but also within a single trajectory. A task-level or globally fixed rate is insufficient. Coverage should depend on the current observation, allowing wider strides during redundant motion while retaining dense action tokens around contact-sensitive interactions. These observations have laid the foundations for the design of APAS described next.

5 APAS: Adaptive Physical Action Scaling

APAS decouples a fixed action-token budget from future trajectory coverage by allowing each action token to represent a variable amount of demonstration-trajectory progress. It consists of an Offline

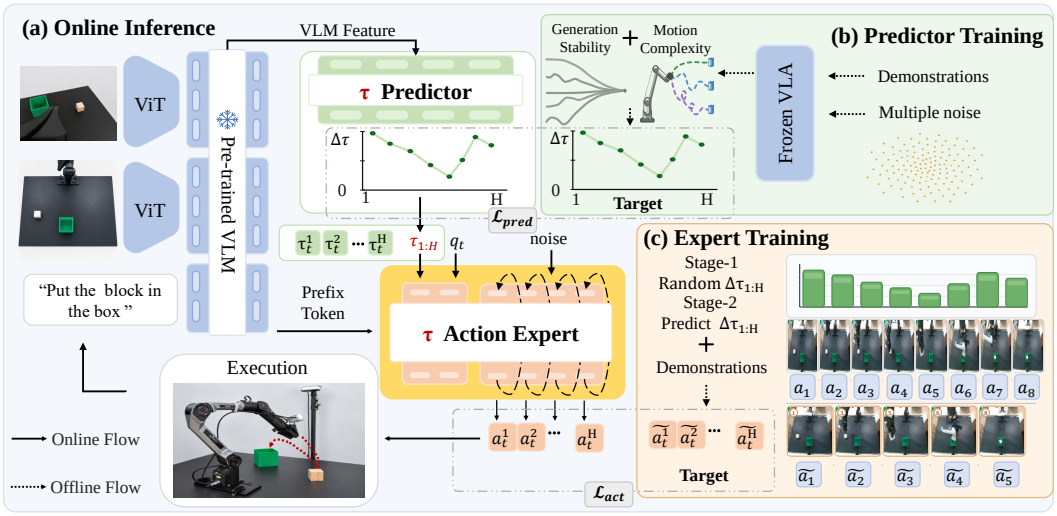

Figure 4: **APAS inference and training.** **(a)** Online inference with the τ-Predictor and τ-Action Expert. **(b)** Offline Temporal Teacher supervision and Predictor training. **(c)** Two-stage τ-Action Expert training.

Temporal Teacher that constructs observation-dependent token-speed targets, a lightweight Token-Speed Predictor (τ-Predictor) that predicts these targets online, and a Profile-Conditioned Action Expert (τ-Action Expert) that generates a fixed-length action chunk under the predicted temporal profile. The entire architecture is illustrated in Fig. 4. We first define token speed below.

Definition 1 (Token Speed). For an H-token action chunk, the token speed $\Delta\tau_t^k$ denotes the amount of demonstration-trajectory progress represented by token k, measured in demonstration control-step units. We define the cumulative trajectory coordinate as,

$$\tau_t^k = \tau_t^{k-1} + \Delta\tau_t^k, \qquad \tau_t^0 = 0. \tag{2}$$

$\Delta\boldsymbol{\tau}_t = (\Delta\tau_t^1, \ldots, \Delta\tau_t^H)$ is the token-speed profile. A larger $\Delta\tau_t^k$ lets token k advance further along the demonstration trajectory, while τ_t^H determines the total future trajectory covered by the chunk.

5.1 Adaptive Token Speed Prediction

To achieve adaptive token speed prediction, we need supervised targets indicating where the future trajectory requires dense action coverage and where larger token speeds are appropriate. Since such temporal allocation labels are absent in native demonstrations, we propose an Offline Temporal Teacher first.

Offline Temporal Teacher. To find supervised targets, we utilize two signals from stochastic generations of the frozen base VLA: ❶ *Generation Instability* measures how much the clean-action estimate continues to change during late refinement; ❷ *Action Variation* measures local variation between neighboring actions in the final generated trajectory. Large values of either signal indicate that finer temporal resolution is desirable. We convert the two signals to percentile scores $u_{D,t}$ and $u_{P,t}$ respectively, and fuse them as,

$$\rho_t(s) = \mathcal{A}\big(1 - (1 - u_{D,t})(1 - u_{P,t})\big)(s), \tag{3}$$

where $\mathcal{A}(\cdot)$ aggregates the fused scores along the demonstration trajectory and converts them into a smooth temporal demand density. The resulting temporal demand density $\rho_t(s)$ is high wherever either generation instability or local action variation indicates that denser coverage is needed. Given $\rho_t(s)$, the Teacher chooses token locations so that each token covers approximately the same amount of temporal demand,

$$\int_{\tau_t^{k-1,\star}}^{\tau_t^{k,\star}} \rho_t(s)\, ds \approx \gamma, \qquad \Delta\tau_t^{k,\star} = \tau_t^{k,\star} - \tau_t^{k-1,\star}. \tag{4}$$

Here, $\star$ denotes Teacher targets and γ is the target demand per token. High-demand regions therefore produce smaller $\Delta\tau_t^{k,\star}$ and denser action coverage, while low-demand regions produce larger token

speeds. The same H-token budget can thus cover different amounts of future trajectory according to the current context.

Online Token Speed τ-Predictor. In practice, the computation time of the Offline Temporal Teacher is high for real-time deployment. Hence, we distill its outputs into a lightweight token speed τ-Predictor. Given frozen VLM features $\mathbf{z}_t$, the Predictor outputs the token-speed profile,

$$\Delta\boldsymbol{\tau}_t = g_\phi(\mathbf{z}_t) \approx \Delta\boldsymbol{\tau}_t^\star, \tag{5}$$

where g_ϕ denotes the τ-Predictor parameterized by ϕ. The cumulative coordinates $\boldsymbol{\tau}_t$ then follow from Eq. (2). We train the Predictor with the following weighted Huber objective (Huber, 1964):

$$\mathcal{L}_{\text{pred}}(\phi) = \mathbb{E}\left[\frac{\lambda_\Delta}{H}\sum_{k=1}^{H}\mathcal{H}_\delta(\Delta\tau_t^k - \Delta\tau_t^{k,\star}) + \frac{\lambda_\tau}{H}\sum_{k=1}^{H}\mathcal{H}_\delta(\tau_t^k - \tau_t^{k,\star}) + \lambda_L\mathcal{H}_\delta(\tau_t^H - \tau_t^{H,\star})\right]. \tag{6}$$

with $\mathcal{H}_\delta$ denoting the Huber loss, and the expectation is over training samples. Here, only the τ-Predictor is optimized.

5.2 Profile-Conditioned Action Expert

The τ-Predictor provides an observation-dependent token-speed profile $\Delta\boldsymbol{\tau}_t$, but this profile alone does not accelerate execution. The Action Expert itself must learn to condition its generation on the requested temporal coverage. This subsection describes how APAS injects the temporal profile into action generation and constructs the corresponding demonstration targets.

Token-Speed Conditioning. In standard VLA, $\Delta\tau_t^k = 1$: each token represents exactly one demonstration step, so the Expert needs no additional information to decide how far its k- -th output should advance. APAS breaks this fixed mapping, e.g., $\Delta\tau_k$ may be 3 during stable transport (large stride) or 0.5 during contact (fine resolution). If the Expert is not explicitly told, it has no way to distinguish and the only remaining option is post-hoc resampling, which we have shown is unreliable. The τ-Action Expert keeps the output length fixed at H, receives both the local token speed $\Delta\tau_t^k$ and cumulative coordinate τ_t^k through a lightweight temporal branch,

$$\widetilde{\mathbf{z}}_{t,k}^a = \mathbf{z}_{t,k}^a + f_\tau\left(\tau_t^k, \Delta\tau_t^k\right), \tag{7}$$

where $\mathbf{z}_{t,k}^a$ is the original action-token representation and f_τ is a small MLP operating on sinusoidal positional encodings of its two arguments (See Appendix B.1), in which τ_t^k tells where on the demonstration trajectory it lies and $\Delta\tau_t^k$ indicates how much trajectory progress it must cover. $\widetilde{\mathbf{z}}_{t,k}^a$ is passed to the original flow-based action generator. The VLM backbone and its denoising-time conditioning are kept unchanged.

Retimed Demonstration Targets. The Expert's k-th output token now corresponds to position τ_t^k on the original demonstration, which is generally non-integer (e.g., $\tau_t^k = 3.5$), but raw demonstrations are stored at integer step indices. Hence, we need to retime the demonstration without altering the native action format of the underlying policy. In particular, for delta-action policies (e.g., $\pi_{0.5}$ + LIBERO), we first accumulate the per-frame translational increments into a continuous trajectory and then convert it back into relative increments between consecutive temporal coordinates. For absolute-waypoint policies (e.g., X-VLA + LIBERO), we directly interpolate the demonstrated pose at each τ_t^k. Rotation and gripper commands are retimed according to their native action representations. Since the same $\boldsymbol{\tau}_t$ is used for both conditioning and supervision, it guarantees there is no train-test mismatch.

Expert Training. The τ-Predictor must be trained itself, so its early outputs are unreliable. If the Expert were trained from the start on the Predictor's output, it would face a chicken-and-egg problem such that adapting to a signal source that has not yet converged. We resolve this with a two-stage approach that first teaches the Expert the general skills, then adapts it to the Predictor's mature output distribution. We first train the τ-Action Expert with diverse synthetic token-speed profiles. Pairing training observations with different profiles and their corresponding retimed targets teaches the Expert how its actions should change with the requested trajectory progress. After Predictor distillation, we further adapt the Expert using a mixture of Predictor-generated and synthetic profiles. Both stages use the base policy's flow-matching objective with the retimed target $\widetilde{\mathbf{A}}_t^{\text{demo}}$,

$$\mathcal{L}_{\text{act}}(\theta) = \mathbb{E}\left[\left\|\mathbf{v}_\theta\left(\widetilde{\mathbf{A}}_t^\lambda, \lambda \mid \mathbf{z}_t, \mathbf{q}_t, \boldsymbol{\tau}_t\right) - \left(\widetilde{\mathbf{A}}_t^{\text{demo}} - \boldsymbol{\epsilon}\right)\right\|_F^2\right]. \tag{8}$$

$\widetilde{\mathbf{A}}_t^{\lambda} = \lambda \widetilde{\mathbf{A}}_t^{\text{demo}} + (1-\lambda)\boldsymbol{\epsilon}$, and the expectation is over training samples, temporal profiles, flow time, and noise. The VLM stays frozen throughout and τ-Predictor is frozen during the second stage.

6 EXPERIMENTS

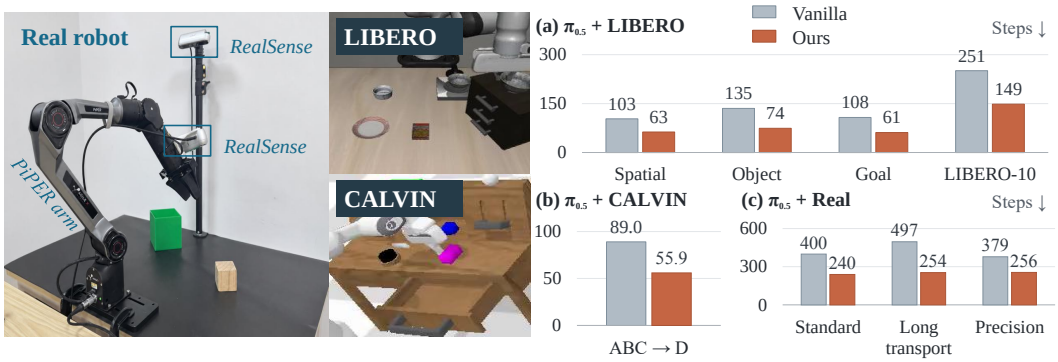

Figure 5: **Evaluation environments and execution steps.** APAS reduces execution steps across LIBERO, CALVIN, and real-robot settings.

6.1 EXPERIMENTAL SETUP

Benchmarks and Baselines. We evaluate $\pi_{0.5}$ (Black et al., 2025a) and X-VLA (Zheng et al., 2026) on LIBERO (Liu et al., 2023) (40 tasks, 50 episodes each) and $\pi_{0.5}$ on CALVIN (Mees et al., 2022) (ABC→D, 1,000 five-subtask chains). Under the same low-level execution protocol, we compare APAS with Vanilla and Expanded-Horizon Resampling (EHR) at $1.5-2.5\times$, and training-side acceleration methods ISR and DemoSpeedup. Execution-side approaches that require modifying to the controller are outside the scope. Appendix C gives training and baseline configurations. Fig. 5 shows the environments and execution-step comparison with Vanilla.

Evaluation metrics. We report average task success on LIBERO and completion of at least k subtasks (Len-k), average completed length on CALVIN. **Steps** and **Queries** are averaged over successful episodes or successfully completed CALVIN subtasks. Real-robot trials also report completion time. Appendix E.6 reports all-rollout Steps and Queries, including failures.

6.2 MAIN RESULTS ON $\pi_{0.5}$ + LIBERO

We can see that APAS maintains near-Vanilla success while reducing successful-rollout Steps by 41.7% (Table 1). As the fixed EHR retiming rate increases, success rate drops sharply without further reducing execution cost. This pattern supports adapting trajectory coverage to the current context rather than uniformly increasing the retiming rate.

Table 1: **Main results on $\pi_{0.5}$ + LIBERO**. In Tables 1–4, best values are bold, with ties included; Table 4 compares methods within each condition.

Method	Success rate (%) ↑					Execution cost ↓	
	Spatial	Object	Goal	LIBERO-10	Avg.	Steps	Queries
Vanilla	**98.8**	**98.2**	**98.0**	92.4	**96.85**	147.7	15.26
EHR-1.5×	91.6	97.6	96.0	89.8	93.75	124.5	12.88
EHR-2.0×	67.0	73.4	80.8	61.0	70.55	133.4	13.74
EHR-2.5×	34.4	58.0	61.6	32.4	46.60	141.2	14.52
ISR	94.8	98.0	88.4	79.0	90.05	102.4	10.68
DemoSpeedup	91.2	97.4	82.2	91.2	90.50	115.9	12.05
APAS	96.2	97.8	97.6	**93.4**	96.25	**86.2**	**9.04**

6.3 LONG-HORIZON EVALUATION ON CALVIN

On CALVIN, APAS increases five-subtask completion from 62.7% to 67.4% while reducing successful-subtask Steps by 37.2% (Table 2). Under CALVIN's fixed per-subtask step budget, selective temporal allocation may spend fewer control steps on stable motion and leave more room for precise adjustments, consistent with the joint improvement in completion and execution cost.

Table 2: **Long-horizon results on $\pi_{0.5}$ + CALVIN ABC→D.** Steps and Queries are successful-subtask means; Chain Steps are averaged over successful five-subtask chains.

Method	Completion rate (%) ↑					Avg. Len ↑	Execution cost ↓		
	Len-1	Len-2	Len-3	Len-4	Len-5		Steps	Queries	Chain Steps
Vanilla	93.4	85.3	77.2	70.1	62.7	3.887	89.0	18.18	443.3
EHR-1.5×	94.6	88.1	81.2	73.5	65.5	4.029	68.5	14.10	339.8
EHR-2.0×	92.5	81.7	69.8	59.9	52.1	3.560	62.3	12.86	305.7
EHR-2.5×	87.4	72.5	60.7	49.9	40.5	3.110	61.7	12.73	295.9
ISR	91.2	79.5	70.9	61.9	51.4	3.549	67.5	13.90	332.7
DemoSpeedup	87.5	74.6	61.6	50.7	40.9	3.153	60.8	12.57	292.9
APAS	**95.6**	**89.3**	**82.4**	**74.9**	**67.4**	**4.096**	**55.9**	**11.57**	**274.5**

6.4 GENERALIZATION ACROSS VLA BACKBONES

In our LIBERO configuration, X-VLA uses absolute waypoints rather than $\pi_{0.5}$'s delta actions. APAS matches Vanilla's success while reducing successful-rollout Steps by 41.9% (Table 3). It also outperforms ISR and DemoSpeedup in overall success with fewer Steps and Queries. The similar efficiency gains across $\pi_{0.5}$ and X-VLA, despite their different architectures and action representations, support adaptive temporal allocation as a design principle applicable across VLA backbones.

Table 3: **X-VLA + LIBERO results.** Steps and Queries are successful-rollout means.

Method	Success rate (%) ↑					Execution cost ↓	
	Spatial	Object	Goal	LIBERO-10	Avg.	Steps	Queries
Vanilla	**98.0**	**99.6**	**99.0**	92.4	**97.25**	151.2	15.56
EHR-1.5×	74.8	55.4	73.0	38.8	60.50	115.2	11.96
EHR-2.0×	63.4	51.4	59.6	40.8	53.80	115.8	12.05
EHR-2.5×	57.4	45.4	48.8	40.8	48.10	109.2	11.41
ISR	93.6	98.8	97.6	89.0	94.75	95.7	9.99
DemoSpeedup	94.0	94.8	94.6	85.4	92.20	94.8	9.94
APAS	96.2	98.8	**99.0**	**95.0**	**97.25**	**87.8**	**9.22**

6.5 REAL-WORLD DEPLOYMENT

We evaluate Standard, Long Transport, and Precision Placement pick-and-place conditions, with 20 trials per method and condition. Methods share training data and evaluation settings (see Appendix C.3).

Table 4: **Real-robot pick-and-place evaluation.** Time, Steps, and Queries: successful-trial means.

Condition	Method	Success ↑	Time (s) ↓	Steps ↓	Queries ↓
Standard	Vanilla	70%	15.92	400.00	27.29
	EHR	45%	11.26	274.22	18.89
	APAS	**75%**	**9.52**	**239.67**	**16.53**
Long Transport	Vanilla	**75%**	19.85	497.00	33.73
	EHR	45%	11.75	295.00	20.44
	APAS	**75%**	**10.12**	**254.33**	**17.47**
Precision Placement	Vanilla	**70%**	14.90	378.93	26.00
	EHR	30%	10.12	**240.83**	**16.67**
	APAS	**70%**	**10.09**	255.71	17.50

We can see that APAS matches or exceeds Vanilla's success rate across all three conditions while reducing completion time by 32.3–49.0% (Table 4). The larger time reduction in Long Transport than in Precision Placement is consistent with broader coverage during extended motion and finer resolution during precise placement. This contrast connects faster transport and reliable placement to context-dependent temporal allocation rather than uniform scaling.

6.6 Component Ablations

Table 5 isolates coverage selection and action generation on LIBERO-10. APAS w/o τ-Predictor retains the conditioned Expert but uses a fixed token-speed profile; APAS w/o τ-Expert applies the predicted profile to the base policy's trajectory through post-hoc retiming.

Table 5: **Component ablations on LIBERO-10.** Steps and Queries are successful-rollout means.

Variant	Components		SR (%) ↑	Execution cost ↓	
	τ-Expert	τ-Predictor		Steps	Queries
Vanilla	×	×	92.40	250.9	25.52
APAS w/o τ-Predictor (2.0×)	✓	×	82.60	151.3	15.61
APAS w/o τ-Predictor (2.5×)	✓	×	67.20	169.4	17.41
APAS w/o τ-Predictor (3.0×)	✓	×	37.80	162.5	16.70
APAS w/o τ-Expert	×	✓	49.60	226.1	23.00
APAS	✓	✓	**93.40**	**148.5**	**15.27**

Without the τ-Predictor, using a fixed 3.0× token speed reduces success from 93.4% for full APAS to 37.8%. This contrast highlights the importance of adaptive slowdown since APAS can take larger trajectory strides during stable motion while retaining finer temporal resolution for precision-sensitive interactions. Without the τ-Expert, success falls to 49.6%, showing that the predicted temporal profile must directly condition action generation rather than be applied only through post-hoc retiming. Further controls and diagnostic analyses are provided in Appendices D–E.

6.7 Visualization Analysis

Fig. 6 further visualizes this adaptive behavior. At selected Move, Contact, and Place observations, Vanilla and APAS receive the same input, and we report the predicted relative future span τ_t^H / H covered by the H-token action chunk. APAS expands future coverage during Move while using shorter spans around Contact and Place, consistently in both simulation and real-robot examples.

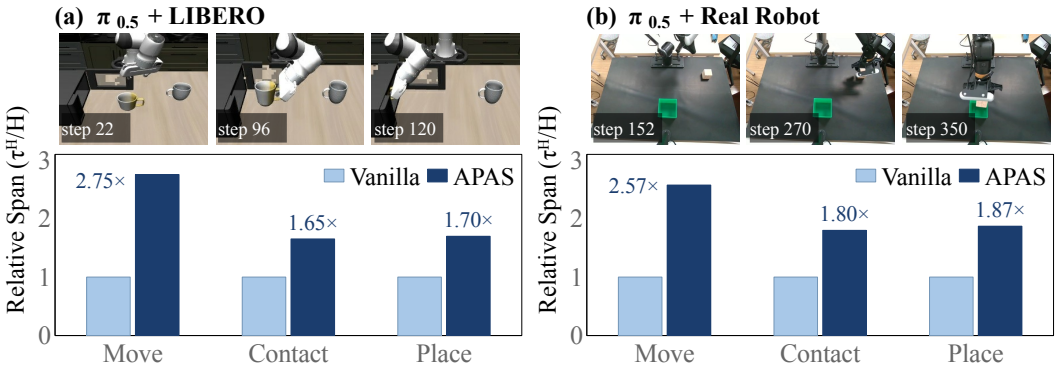

Figure 6: Visualization of adaptive future coverage in simulation and on a real robot.

7 Conclusion

In this work, we propose an adaptive temporal-generation framework that decouples a fixed action-token budget from future trajectory coverage. By adapting temporal allocation to the current context, APAS accelerates stable motion while retaining dense action coverage around precision-sensitive interactions. Extensive simulation and real-world experiments demonstrate vast execution speedups compared to existing schemes.

AI USE STATEMENT

Generative AI tools were used for language polishing, including improvements to grammar, wording, and sentence clarity. All AI-assisted edits were reviewed and verified by the authors. The research ideas, methodology, experimental design, data analysis, scientific claims, and conclusions were developed entirely by the authors. The authors take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide detailed implementation and evaluation settings in the appendix, including APAS training, model and baseline configurations, benchmark protocols, and the real-robot setup. We also describe the construction of the Offline Temporal Teacher and the Token-Speed Predictor in detail. LIBERO and CALVIN are publicly available benchmarks with standardized evaluation protocols.

REFERENCES

Nadun Ranawaka Arachchige, Zhenyang Chen, Wonsuhk Jung, Woo Chul Shin, Rohan Bansal, Pierre Barroso, Yu Hang He, Yingyan Celine Lin, Benjamin Joffe, Shreyas Kousik, and Danfei Xu. Sail: Faster-than-demonstration execution of imitation learning policies. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 721–749, 2025. URL https://proceedings.mlr.press/v305/arachchige25a.html.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025c. doi: 10.52202/085713-1122. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/300ccb2187dedd4edcc07f7e76d8e553-Abstract-Conference.html.

James E. Bobrow, Steven Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985. doi: 10.1177/027836498500400301.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Anthony Brohan et al. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://www.roboticsproceedings.org/rss19/p025.html.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems XIX*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://doi.org/10.15607/RSS.2023.XIX.026.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://www.roboticsproceedings.org/rss20/p090.html.

Lingxiao Guo, Zhengrong Xue, Zijing Xu, and Huazhe Xu. Demospeedup: Accelerating visuomotor policies via entropy-guided demonstration acceleration. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 599–609, 2025. URL https://proceedings.mlr.press/v305/guo25a.html.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. doi: 10.1214/aoms/1177703732.

Kangye Ji, Yuan Meng, Hanyun Cui, Ye Li, Jianbo Zhou, Shengjia Hua, Lei Chen, and Zhi Wang. Block-wise adaptive caching for accelerating diffusion policy. In *International Conference on Learning Representations*, pp. 22805–22858, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/2712b17bb58ea5b2b65c45857b024744-Abstract-Conference.html.

Dong Jing, Gang Wang, Jiaqi Liu, Weiliang Tang, Zelong Sun, Yunchao Yao, Zhenyu Wei, Yunhui Liu, Zhiwu Lu, and Mingyu Ding. Mixture of horizons in action chunking. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2511.19433. arXiv:2511.19433.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025a. doi: 10.15607/RSS.2025.XXI.017. URL https://www.roboticsproceedings.org/rss21/p017.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Sunwoo Kim, Jeongjun Kim, and Joseph J. Lim. Time optimal execution of action chunk policies beyond demonstration speed. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/fab80bb9d97e9b9ff5c19f91f72838c6-Abstract-Conference.html. RACE: Reachability-aware Accelerated Chunk Execution.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks*, volume 36, 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108. URL https://ieeexplore.ieee.org/document/9788026.

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. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://www.roboticsproceedings.org/rss21/p012.html.

Hung Pham and Quang-Cuong Pham. A new approach to time-optimal path parameterization based on reachability analysis. *IEEE Transactions on Robotics*, 34(3):645–659, 2018. doi: 10.1109/TRO.2018.2819195.

Kang G. Shin and Neil D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985. doi: 10.1109/TAC.1985.1104009.

Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327, 2009. doi: 10.1109/TAC.2009.2028959.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. SpecPrune-VLA: Accelerating vision-language-action models via action-aware self-speculative pruning. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64525.

Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63399–63416, 2025. URL https://proceedings.mlr.press/v267/wang25ba.html.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems*, volume 38, pp. 164448–164473, 2025. doi: 10.52202/085713-5484. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f062da1973ac9ac61fc6d44dd7fa309f-Abstract-Conference.html.

Licheng Yang, Lingfeng Qian, Fei Zheng, Yonghao He, Wei Sui, Shuangshuang Li, and Hu Su. Improving robotic imitation learning via trajectory standardization. *arXiv preprint arXiv:2606.22907*, 2026. URL https://arxiv.org/abs/2606.22907. Accepted to IROS 2026; proceedings metadata pending.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. In *Advances in Neural Information Processing Systems*, volume 37, pp. 56619–56643, 2024. doi: 10.52202/079017-1803. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/67b0e7c7c2a5780aeefe3b79caac106e-Abstract-Conference.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://www.roboticsproceedings.org/rss19/p016.html.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/630d293833e09e1ecd892a898a20b074-Abstract-Conference.html.

Brianna Zitkovich et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Additional Analysis of Temporal Acceleration

A.1 Native-Horizon Resampling and the Query Bottleneck

Figure 7 expands the execution-timeline comparison in Section 4.1. Vanilla alternates one policy query with the execution of an H-action chunk. Native-Horizon Resampling (NHR) instead compresses each predicted H-token chunk into H/N executed actions, while the chunk still represents approximately the same future trajectory segment.

NHR therefore reduces the number of executed control actions but does not substantially expand the trajectory covered by each policy query. The number of policy queries consequently remains largely unchanged as the resampling rate increases. As physical execution becomes shorter, inference occupies an increasingly large fraction of the overall execution timeline.

This distinction motivates expanding trajectory coverage at generation time rather than only sparsifying the actions returned by a native-horizon policy.

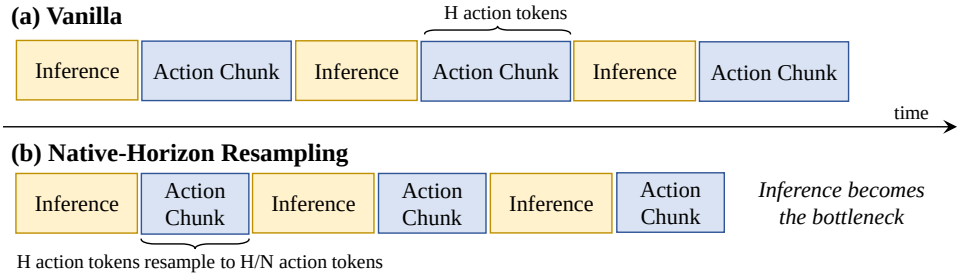

Figure 7: **Execution timelines for Vanilla and Native-Horizon Resampling.** NHR shortens physical execution within each predicted chunk while preserving approximately the same future trajectory coverage per policy query.

B Additional Method Details

B.1 Temporal Reparameterization and τ-Action Expert

Given an action horizon H and a token-speed profile

$$\Delta\tau_{1:H} = (\Delta\tau_1, \ldots, \Delta\tau_H), \qquad \Delta\tau_k \in [d^-, d^+], \tag{9}$$

we construct cumulative temporal coordinates

$$\tau_k = \sum_{j=1}^{k} \Delta\tau_j, \qquad k = 1, \ldots, H. \tag{10}$$

These coordinates specify the positions along the demonstration trajectory represented by the H output action tokens. Algorithm B.1 summarizes temporal reparameterization for the action representations used in our experiments.

The reparameterization preserves the native action format of the underlying policy. For delta-action spaces, the demonstration is first accumulated into a continuous trajectory and then converted back into relative increments between consecutive temporal boundaries. Rotation is retimed over the same source interval using the native rotation representation. Absolute-waypoint policies instead interpolate poses directly at the requested coordinates τ_k. Discrete gripper commands are taken from the nearest source-time control step.

Temporal conditioning. The τ-Action Expert receives both the cumulative trajectory coordinate and the local token speed. We encode these quantities with sinusoidal positional embeddings and inject them through a residual temporal branch:

$$e_k^\tau = \left[\mathrm{PE}\left(\frac{\tau_k}{\tau_{\max}}\right); \mathrm{PE}\left(\frac{\Delta\tau_k - d^-}{d^+ - d^-}\right) \right], \tag{11}$$

$$\tilde{z}_k^a = z_k^a + f_\tau(e_k^\tau). \tag{12}$$

Algorithm B.1 Temporal Reparameterization

Require: Source demonstration A^{demo}, profile $\Delta\tau_{1:H}$, action representation
Ensure: Retimed targets $\widetilde{A}^{\text{demo}}$ and temporal coordinates $\tau_{1:H}$

$\tau_0 \leftarrow 0$
for $k = 1, \dots, H$ **do**
  $\tau_k \leftarrow \tau_{k-1} + \Delta\tau_k$
end for
if representation is delta action **then**
  $p \leftarrow \textsc{AccumulateTranslation}(A^{\text{demo}})$
  for $k = 1, \dots, H$ **do**
    $\Delta p_k \leftarrow p(\tau_k) - p(\tau_{k-1})$
    $\Delta R_k \leftarrow \textsc{RetimeRotation}(A^{\text{demo}}, \tau_{k-1}, \tau_k)$
    $g_k \leftarrow \textsc{NearestGripperCommand}(A^{\text{demo}}, \tau_k)$
    $\tilde{a}_k \leftarrow (\Delta p_k, \Delta R_k, g_k)$
  end for
else if representation is absolute waypoint **then**
  for $k = 1, \dots, H$ **do**
    $w_k \leftarrow \textsc{InterpolateWaypoint}(A^{\text{demo}}, \tau_k)$
    $g_k \leftarrow \textsc{NearestGripperCommand}(A^{\text{demo}}, \tau_k)$
    $\tilde{a}_k \leftarrow (w_k, g_k)$
  end for
end if
return $\widetilde{A}^{\text{demo}}, \tau_{1:H}$

Here, f_τ maps the temporal embedding to the action-token dimension. Its output layer is zero-initialized, so the temporal residual is initially zero. The backbone's original flow-denoising-time conditioning is retained separately.

B.2 Offline Temporal Teacher

The Offline Temporal Teacher converts stochastic generation behavior of the frozen base VLA into observation-dependent token-speed targets. The construction follows the sequence

$$\text{stochastic generations} \rightarrow (D, P) \rightarrow \text{fused resolution demand} \rightarrow \text{source alignment} \rightarrow \Delta\tau^\star. \quad (13)$$

For each valid observation origin t, we draw N stochastic generations from the frozen VLA. Generation instability D_t^k measures how much the clean estimate of token k continues to change during late refinement, while action variation P_t^k measures local variation in the final generated trajectory.

Let $\hat{a}_{n,k}^{(m)}$ denote the clean-action estimate for token k at refinement step m in stochastic generation n, and let a_n^k denote its final generated action. We compute

$$D_t^k = \text{Median}_n \, \text{Mean}_{m \in \mathcal{M}} \, \text{RMS}_7 \left(\hat{a}_{n,k}^{(m+1)} - \hat{a}_{n,k}^{(m)} \right), \quad (14)$$

and

$$P_t^k = \text{Median}_n \, \frac{g_{t,n}^{k-1} + g_{t,n}^k}{2}, \qquad g_{t,n}^k = \text{RMS}_6 \left(a_n^{k+1} - a_n^k \right). \quad (15)$$

Endpoint tokens reuse the nearest available action difference.

The raw values of D and P are converted to empirical percentile scores $u_D, u_P \in [0, 1]$. We combine them with Dual-OR fusion,

$$R_t^k = 1 - \left(1 - u_{D,t}^k\right) \left(1 - u_{P,t}^k\right), \quad (16)$$

so elevated demand from either trace is retained.

Algorithm B.2 summarizes the complete Teacher procedure.

Predictions from different observation origins that refer to the same demonstration source step are aggregated along the source trajectory and smoothed to form $\widetilde{R}_e(s)$. The corresponding resolution density is

$$\rho_e(s) = \left(\epsilon + \widetilde{R}_e(s) \right)^\alpha. \quad (17)$$

High-density regions therefore receive denser token placement, while low-density regions permit larger temporal strides. Token boundaries are obtained by sequentially inverting cumulative resolution demand under the supported token-speed bounds.

Algorithm B.2 Offline Temporal Teacher

Require: Demonstrations $\{E_e\}$, frozen base VLA π_{base}, horizon H, bounds $[d^-, d^+]$, demand γ
Ensure: Teacher targets $\{\Delta\tau_t^\star\}$

```
for each demonstration episode E_e do
    for each valid observation origin t do
        Sample N stochastic generations from π_base
        Compute generation instability D_t
        Compute action variation P_t
        Calibrate D_t, P_t to percentiles u_{D,t}, u_{P,t}
        R_t ← 1 − (1 − u_{D,t})(1 − u_{P,t})
    end for
    R_e ← SourceAlignAndAggregate({R_t})
    R̃_e ← Smooth(R_e)
    ρ_e ← (ϵ + R̃_e)^α
    for each valid observation origin t do
        Δτ_t^⋆ ← EqualDemandAllocate(ρ_e, t, H, [d^−, d^+], γ)
    end for
end for
return {Δτ_t^⋆}
```

B.3 Predictor Distillation and APAS Training

The τ-Predictor amortizes the Offline Temporal Teacher into a lightweight online module. We pool the final-layer multimodal VLM representation into z_t, normalize the feature using Predictor-training statistics, and process it with a two-layer GELU MLP.

The output head separates the total future span from its within-chunk allocation. It predicts a scalar span logit ℓ and token-wise allocation logits $\beta_{1:H}$. The bounded scalar

$$b(\ell) = \epsilon_b + (1 - 2\epsilon_b)\sigma(\ell) \tag{18}$$

controls the mean allocation, while centered token logits redistribute that allocation across the chunk:

$$\bar{\beta}_k = \beta_k - \frac{1}{H}\sum_{j=1}^{H}\beta_j, \tag{19}$$

$$p_k = \sigma(\bar{\beta}_k + c), \qquad \frac{1}{H}\sum_{k=1}^{H} p_k = b(\ell), \tag{20}$$

$$\widehat{\Delta\tau}_k = d^- + (d^+ - d^-)p_k. \tag{21}$$

The shared offset c enforces the predicted mean allocation, while the bounded mapping keeps all token speeds inside $[d^-, d^+]$.

Algorithm B.3 summarizes the complete APAS training procedure.

Deployment. The Offline Temporal Teacher is not used during execution. For each policy query, the frozen VLM produces the context representation, the τ-Predictor outputs $\widehat{\Delta\tau}_{1:H}$, and the corresponding cumulative coordinates condition the τ-Action Expert:

$$o_t \to z_t \to \widehat{\Delta\tau}_{1:H} \to \widehat{\tau}_{1:H} \to \tau\text{-Action Expert}. \tag{22}$$

C Experimental Setup

C.1 Benchmarks and Training Configurations

We evaluate APAS with $\pi_{0.5}$ and X-VLA on LIBERO, and with $\pi_{0.5}$ on CALVIN. The four LIBERO suites contain 40 tasks, each with 30 training and one validation demonstration (1,200 training and 40 validation demonstrations). CALVIN contains 34 canonical tasks, each with 40 training and one validation demonstration (1,360 training and 34 validation demonstrations); evaluation follows the ABC→D setting.

The three settings use different native action representations: delta end-effector actions for $\pi_{0.5}$-LIBERO, absolute end-effector waypoints for X-VLA, and padded delta actions for $\pi_{0.5}$-CALVIN.

Algorithm B.3 APAS Training Pipeline

Require: Demonstrations $\mathcal{D}$, base Action Expert θ_{base}, frozen VLM, bounds $[d^-, d^+]$

Stage I: Temporal-Conditioning Pre-training

Initialize θ from θ_{base}
Attach temporal-conditioning branch f_τ
for each Stage-I update **do**
  Sample a demonstration trajectory from $\mathcal{D}$
  Sample $\Delta\tau_{1:H}$ from the synthetic-profile distribution
  $\widetilde{A}^{\text{demo}}, \tau_{1:H} \leftarrow \textsc{TemporalReparameterization}(A^{\text{demo}}, \Delta\tau_{1:H})$
  Update θ with the flow-matching objective conditioned on $\tau_{1:H}$
end for

Predictor Distillation

Construct Teacher targets $\{\Delta\tau_t^\star\}$ using Algorithm B.2
Initialize Predictor parameters ϕ
for each Predictor update **do**
  Sample context z_t and Teacher target $\Delta\tau_t^\star$
  Predict $\widehat{\Delta\tau}_t$
  Update ϕ using losses on token speeds, cumulative coordinates, and final span
end for

Stage II: Predictor-Conditioned Expert Adaptation

Freeze ϕ
for each Stage-II update **do**
  Sample a demonstration trajectory from $\mathcal{D}$
  Obtain $\Delta\tau_{1:H}$ from the frozen Predictor or synthetic-profile sampler
  $\widetilde{A}^{\text{demo}}, \tau_{1:H} \leftarrow \textsc{TemporalReparameterization}(A^{\text{demo}}, \Delta\tau_{1:H})$
  Continue updating θ with the same flow-matching objective
end for

Table 6: **Backbone and training configurations.**

Setting	$\pi_{0.5}$ + LIBERO	X-VLA + LIBERO	$\pi_{0.5}$ + CALVIN
Action representation	Delta EE	Absolute EE	Padded delta
Action horizon H	10	10	5
$\Delta\tau$ range	$[1, 3]$	$[1, 3]$	$[1, 3]$
Stage-I peak LR	2.5×10^{-5}	2.5×10^{-5}	2.5×10^{-5}
Stage-II peak LR	5×10^{-6}	5×10^{-6}	5×10^{-6}
LR schedule	cosine	cosine	cosine
Training epochs	2	2	2

The VLM backbone remains frozen during temporal Expert training. Stage I samples token-speed profiles from a mixture of 20% native, 30% constant, and 50% smooth profiles, with $J = 3$ anchors for smooth profiles. Stage II is initialized from the Stage-I checkpoint.

The Offline Temporal Teacher uses $N = 8$ stochastic generations with $M = 10$ refinement steps. Generation instability is computed over transitions $m = 5, \ldots, 9$.

The Token-Speed Predictor uses the pooled final-layer VLM representation followed by a two-layer MLP with hidden dimensions 1024 and 256. It is trained with AdamW using learning rate 10^{-3}, batch size 512, and weight decay 10^{-4}. Its objective uses $(\lambda_\Delta, \lambda_\tau, \lambda_L) = (1, 0.05, 2)$ with Huber threshold $\delta = 0.3$.

Stage II continues Expert training. The VLM and Predictor remain frozen, and Predictor-generated and synthetic token-speed profiles are sampled with equal probability.

C.2 Baselines and Evaluation Protocols

Table 7: **Baseline implementations.**

Method	Configuration
Vanilla	Original benchmark-specific policy executed at its native temporal resolution.
EHR-$N\times$	Generates an expanded NH-token trajectory and uniformly resamples it to H executed actions. We report $N \in \{1.5, 2.0, 2.5\}$.
ISR	Uses $d_{\text{target}} = 0.022\,\text{m}$, $\lambda_{\text{acc}} = 0.05$, and $\lambda_{\text{dist}} = 1$.
DemoSpeedup	Uses $N = 8$, low/high temporal speeds of $1/3$, KDE bandwidth 1, quantile normalization, HDBSCAN with `min_cluster_size=5`, and the `strict_official` merge rule.
APAS	Predicts an observation-dependent token-speed profile and directly conditions action generation on the corresponding cumulative trajectory coordinates.

Training-side baselines are matched to APAS in Action Expert optimization budget and action horizon within each evaluation setting. ISR and DemoSpeedup use a peak learning rate of 2.5×10^{-5} with a cosine schedule.

Table 8: **Simulation evaluation protocols.**

Setting	Evaluation	H	Performance	Execution metrics
$\pi_{0.5}$ + LIBERO	40 tasks $\times$ 50 episodes	10	SR	Steps, Queries
X-VLA + LIBERO	40 tasks $\times$ 50 episodes	10	SR	Steps, Queries
$\pi_{0.5}$ + CALVIN	ABC$\rightarrow$D, 1,000 chains	5	Len-k, Avg. Len	Steps, Queries, Chain Steps

For LIBERO, we retain each backbone's standard per-task rollout limit; for CALVIN, we retain the standard per-subtask limit.

LIBERO uses the standard task-success and termination protocol with sequential, non-overlapped action-chunk execution. **Steps** counts executed control actions and **Queries** counts policy inference calls. Main-table execution costs are averaged over successful episodes.

For CALVIN, we use the standard ABC$\rightarrow$D long-horizon evaluation with 1,000 five-subtask chains. Steps and Queries are averaged over successfully completed subtasks, while Chain Steps is averaged over chains that complete all five subtasks.

C.3 Real-Robot Setup

Hardware and observations. We use an AgileX PIPER robotic arm with the matching PIPER gripper and an Intel RealSense camera. The policy receives RGB observations and predicts absolute joint targets.

Task conditions. We evaluate three pick-and-place conditions using a wooden block and a green target box. **Standard** places the target box near the block. **Long Transport** increases the distance between the block and the target while using the same box. **Precision Placement** keeps the target nearby but uses a smaller box, requiring more precise placement. The same trained policy is evaluated across all three conditions without condition-specific fine-tuning.

Training and execution. Vanilla and APAS are trained on the same 200 real-robot demonstrations, collected with randomized block and box positions, for 20k optimization steps. All methods use synchronous execution at 30 Hz with action horizon $H = 15$.

Vanilla directly executes its 15 predicted actions. EHR expands the generation horizon to $G = \lceil 15r \rceil$, with $r = 2$, and uniformly resamples the generated trajectory to 15 executed actions.

APAS predicts a 15-token profile with $\Delta\tau \in [1, 3]$, using a Teacher calibrated to target mean token speed $\bar{r} = 2.0$.

Evaluation. We run 20 trials per method and condition. A trial is successful when the block is placed inside the target box. Success rate is computed over all trials, while completion time, Steps, and policy Queries are averaged over successful trials.

D Additional Ablations

We examine the contribution of continued native-resolution training and the dual-trace supervision used by the Temporal Teacher.

D.1 Native-Resolution Fine-Tuning Control

APAS further optimizes the Action Expert after initialization. We test whether its execution gains can be reproduced by continued fine-tuning at the native temporal resolution.

Starting from Vanilla, we fine-tune the Action Expert at native temporal resolution for the same number of updates as APAS's Action Expert, retaining the standard consecutive action targets, $\Delta\tau_k = 1$ for all k.

The VLM remains frozen and the Action Expert remains trainable.

Table 9: **Native-resolution fine-tuning control on LIBERO-10.**

Method	SR (%) ↑	Success Steps ↓	Success Queries ↓
Vanilla	92.4	250.9	25.52
Native fine-tuning	92.8	250.8	25.53
APAS	**93.4**	**148.5**	**15.27**

Native fine-tuning changes success from 92.4% to 92.8% while leaving successful-rollout execution cost essentially unchanged. APAS instead reaches 93.4% success while reducing successful-rollout Steps from 250.9 to 148.5 and Queries from 25.52 to 15.27. Continued native-resolution fine-tuning therefore does not reproduce the acceleration obtained from temporally reparameterized action generation.

D.2 Dual-Trace Teacher Ablation

The Temporal Teacher combines generation instability D and action variation P to estimate where finer temporal resolution is needed. We isolate their contributions using three Teacher variants: $R = u_D$ (D-only), $R = u_P$ (P-only), and $R = 1 - (1 - u_D)(1 - u_P)$ (Dual).

All three variants use the same downstream allocation rule, τ-Action Expert, Predictor architecture, and evaluation protocol. Within each operating point, the same allocation quota γ is used across variants.

We evaluate the three Teachers on three LIBERO tasks at target mean token speeds $\bar{r} = 2.0$ and $\bar{r} = 2.25$.

The Dual Teacher retains higher average success at both operating points. At $\bar{r} = 2.0$, Dual reaches 96.0% success, compared with 92.7% for D-only and 92.0% for P-only. The separation becomes larger at $\bar{r} = 2.25$, where Dual retains 93.3% success while both single-trace variants reach 80.0%. The two traces therefore provide complementary cues for temporal allocation.

E Additional Analysis

Table 10: **Dual-trace Teacher ablation on three LIBERO tasks.**

$\bar{r}$	Teacher	LIBERO-10 T9	Goal T2	Goal T3	Avg. SR ↑	All Steps ↓	All Queries ↓
2.0	D-only	92	92	94	92.7	118.6	12.21
	P-only	88	90	**98**	92.0	118.5	12.28
	Dual	**94**	**96**	**98**	**96.0**	**113.9**	**11.73**
2.25	D-only	80	74	86	80.0	152.2	15.57
	P-only	72	84	84	80.0	149.4	15.30
	Dual	**90**	**100**	**90**	**93.3**	**111.2**	**11.57**

We further examine the behavior of APAS beyond the primary operating point reported in the main experiments, including success across temporal operating points, runtime implications, Predictor–Teacher agreement, online temporal adaptation, and the allocation induced by the two Teacher signals.

E.1 Success Across Temporal Operating Points

We compare temporal operating points on a four-task LIBERO subset: Task 0 from LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10. Each point pools 20 episodes per task (80 episodes). APAS uses target rates $\bar{r} \in \{1.5, 2.0, 2.5\}$. ISR varies $d_{\text{target}} \in \{0.006, 0.0145, 0.03\}$, while DemoSpeedup fixes $\text{low_v} = 1$ and uses $\text{high_v} \in \{2, 3, 4, 6\}$. Temporal rate is the mean number of original demonstration frame intervals spanned by an action target token across all valid token intervals in the training targets.

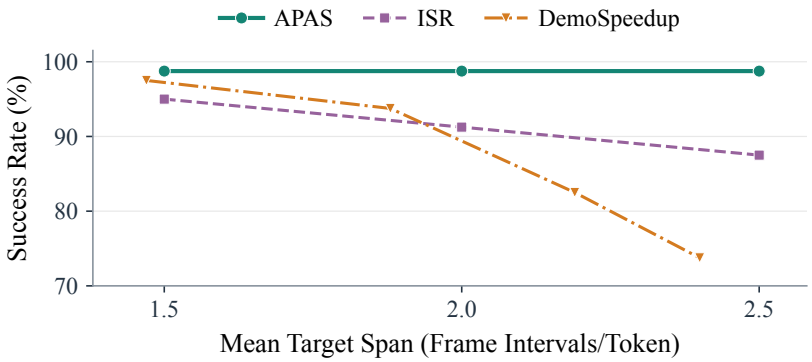

Figure 8: **Success across temporal operating points.** APAS, ISR, and DemoSpeedup on four LIBERO tasks (80 episodes per point).

Across the three APAS operating points, mean all-rollout Steps decrease from 118.5 to 98.1 to 90.5 as the target span increases, while success remains at 98.75%. ISR decreases from 95.0% at $1.5\times$ to 87.5% at $2.5\times$, while DemoSpeedup decreases from 97.5% at $1.47\times$ to 73.75% at $2.4\times$.

E.2 Runtime Implications

E.2.1 Predictor Inference Overhead

The Token-Speed Predictor reuses the frozen VLM representation already computed for action generation and therefore adds only a small module to each policy query.

We measure per-query inference latency on $\pi_{0.5}$-LIBERO over 1,000 timed queries on each GPU after 50 warmup queries. All methods use the same observation, external noise, batch size, action horizon, and number of denoising steps.

APAS increases per-query latency by less than 2% relative to Vanilla on both GPUs. Relative to the corresponding τ-Expert, the Predictor adds approximately 0.9 ms per query.

E.2.2 End-to-End Task-Time Analysis

Table 11: **Per-query inference latency (mean $\pm$ standard deviation).**

GPU	Vanilla	τ-Expert w/o Predictor	APAS
RTX 6000D	69.66 ± 0.78 ms	69.17 ± 0.74 ms	70.05 ± 0.81 ms
RTX 5090	63.52 ± 0.29 ms	63.81 ± 0.62 ms	64.73 ± 0.40 ms

Using Eq. (1) with the measured $\pi_{0.5}$+LIBERO Steps and Queries, $(Q, K)_{\text{Vanilla}} = (15.26, 147.7)$ and $(Q, K)_{\text{APAS}} = (9.043, 86.159)$, APAS yields a stable estimated end-to-end task-time reduction across deployment regimes. Using each method's measured per-query latency from Table 11, the predicted task-time reduction is 41.2–41.6% for control frequencies from 10 to 50 Hz across the two GPUs. The limiting reductions are 41.67% when physical execution dominates and 39.6–40.4% when inference dominates. The similar reductions in Steps and policy Queries therefore translate into consistent end-to-end acceleration rather than shifting the bottleneck between physical execution and inference.

E.3 Predictor–Teacher Agreement Across Settings

The Offline Temporal Teacher constructs temporal-allocation targets from stochastic generation traces, while deployment uses the lightweight τ-Predictor. We therefore compare Predictor outputs with Teacher targets across the evaluated simulation settings.

We evaluate token-speed error, final-span error, and the ordering of future coverage across held-out trajectories.

Table 12: **Predictor–Teacher agreement across evaluated settings.**

Setting	$\Delta\tau$ MAE $\downarrow$	Final-span MAE $\downarrow$	Pearson $\uparrow$	Spearman $\uparrow$	Teacher mean $\Delta\tau$	Predictor mean $\Delta\tau$
$\pi_{0.5}$ + LIBERO	0.165	0.986	0.830	0.827	1.992	1.997
X-VLA + LIBERO	0.112	0.832	0.826	0.849	2.002	1.995
$\pi_{0.5}$ + CALVIN	0.117	0.465	0.902	0.895	2.161	2.149

The Predictor closely follows the Teacher across all three settings. On X-VLA+LIBERO, token-speed and final-span MAEs are 0.112 and 0.832, with Pearson and Spearman correlations of 0.826 and 0.849. On $\pi_{0.5}$+CALVIN, the corresponding correlations increase to 0.902 and 0.895, while the mean token speeds remain closely matched (2.161 for the Teacher and 2.149 for the Predictor). Together with the $\pi_{0.5}$+LIBERO results, these measurements show that the lightweight Predictor preserves both the scale and observation-dependent variation of the Teacher's temporal allocation across different backbones and benchmarks.

Figure 9 further shows that final-span agreement follows the same diagonal structure across both LIBERO backbones and CALVIN, despite their different action horizons.

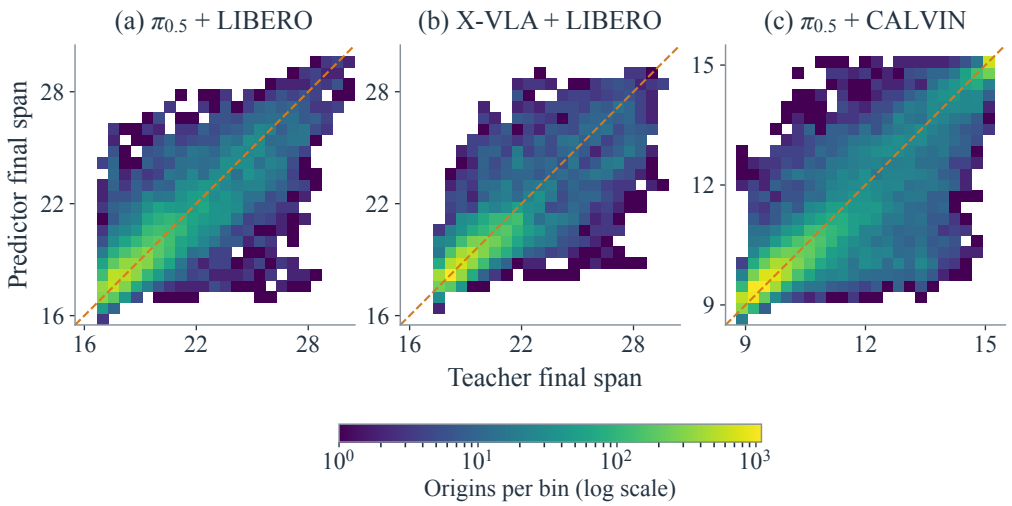

Figure 9: **Teacher–Predictor agreement across three settings.** Each density bin counts held-out observations according to Teacher and Predictor final span; the dashed diagonal indicates exact agreement. Panels share a log-count color scale. LIBERO uses $H = 10$ and CALVIN uses $H = 5$, so their absolute span ranges differ.

E.4 Online Temporal Adaptation

We next inspect how the predicted token-speed profile changes during online execution.

Figure 10 shows an illustrative successful rollout for the instruction *"pick up the chocolate pudding and place it in the basket."* We divide the trajectory into Approach, Grasp, and Transport. The Transport phase also contains the terminal alignment, placement, and release motions.

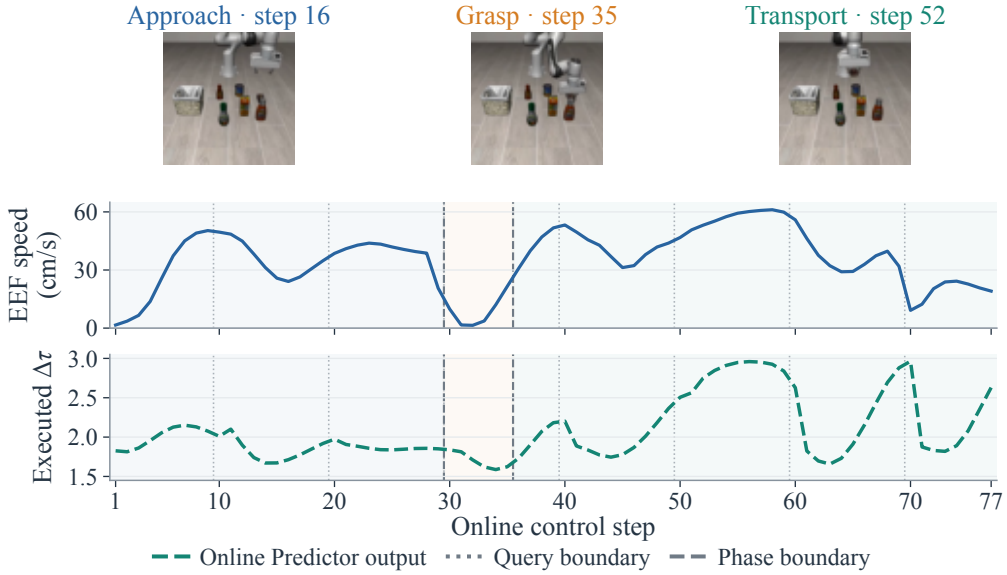

Figure 10: **Online temporal adaptation in an illustrative successful rollout.** The upper row shows observations from Approach, Grasp, and Transport. The aligned lower panels show end-effector translational speed and the executed token speeds predicted online. Dotted lines denote policy-query boundaries and dashed lines denote phase boundaries.

The Predictor changes temporal coverage substantially within the same trajectory. Mean $\Delta\tau$ is 1.898 during Approach and decreases to 1.699 during Grasp, where mean end-effector speed is also

lowest at 8.4 cm/s. During Transport, mean $\Delta\tau$ increases to 2.257, while mean end-effector speed reaches 39.9 cm/s. Local decreases appear again near terminal alignment, placement, and release.

Thus, the Predictor does not select a single temporal rate for the entire task. Precision-sensitive interaction receives denser temporal allocation, while stable transport receives broader future coverage.

E.5 Dual-Trace Allocation Visualization

Section D.2 quantitatively compares the D-only, P-only, and Dual Teachers. Here we directly visualize how the two resolution-demand signals contribute to the final temporal allocation.

Figure 11 provides a qualitative illustration using three demonstration trajectories, denoted Example A, Example B, and Example C. For each trajectory, we plot the source-aligned generation instability percentile u_D, action variation percentile u_P, and the resulting D-only, P-only, and Dual Teacher allocations.

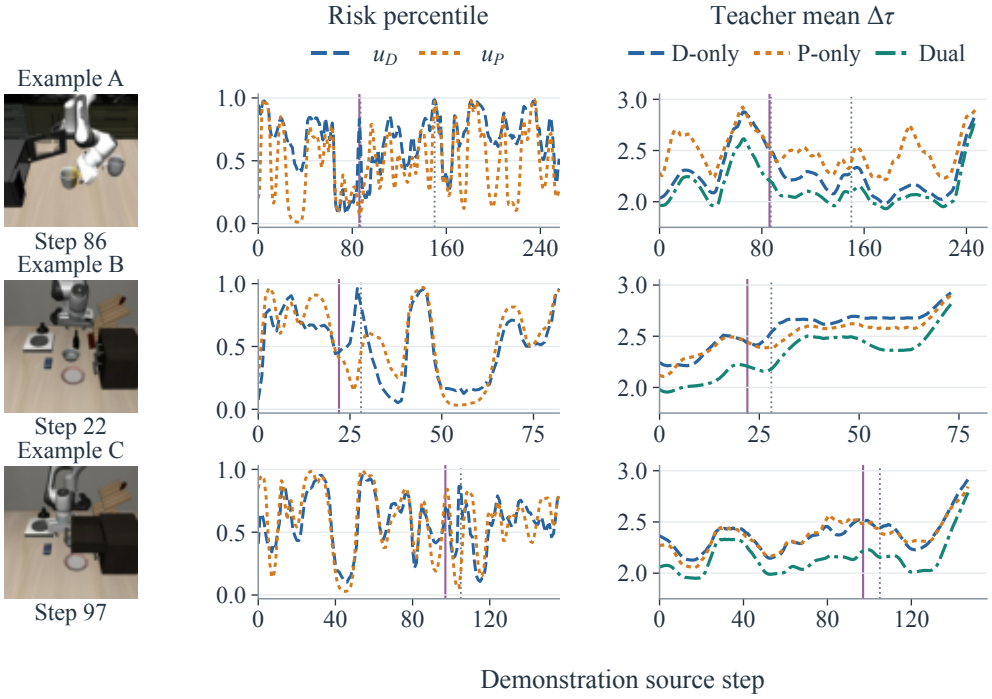

Figure 11: **Qualitative illustration of dual-trace temporal allocation.** Examples A–C show observations and the source-aligned u_D and u_P signals on the left, with corresponding mean token-speed allocations on the right. Vertical lines mark the displayed source positions and dotted lines indicate gripper-command transitions.

The two traces respond differently across the trajectory. Regions with elevated u_D indicate actions that remain unstable late in generation, while elevated u_P marks rapid local variation in the final action trajectory. The Dual Teacher preserves high resolution demand from either source and converts it into smaller local token speeds. This mechanism provides the qualitative counterpart to the quantitative ablation in Section D.2.

E.6 ALL-ROLLOUT EXECUTION COST

Tables 13 and 14 complement the successful-only costs in Tables 1–3 by including failed evaluations. Across the reported acceleration methods, APAS has the lowest all-rollout Steps and Queries in all three settings, alongside the highest success rates reported in the main tables.

Table 13: **All-rollout cost on LIBERO.** Means over all 2,000 episodes per method, including failures.

Method	$\pi_{0.5}$ + LIBERO		X-VLA + LIBERO	
	All Steps ↓	All Queries ↓	All Steps ↓	All Queries ↓
EHR-1.5×	139.303	14.327	400.984	40.365
EHR-2.0×	195.963	19.880	446.726	44.920
EHR-2.5×	247.946	24.981	482.520	48.486
ISR	132.475	13.647	135.456	13.939
DemoSpeedup	136.365	14.052	153.479	15.768
APAS	**96.938**	**10.105**	**108.641**	**11.288**

For CALVIN, we report costs per attempted subtask and per evaluation chain separately. The former includes successful and failed subtask attempts; the latter sums all attempted subtasks within each chain and averages over all 1,000 chains, including early terminations.

Table 14: **All-rollout cost on $\pi_{0.5}$ + CALVIN.** N counts attempted subtasks; chain means use all 1,000 chains.

Method	N	All attempted subtasks		All chains	
		Steps ↓	Queries ↓	Steps ↓	Queries ↓
EHR-1.5×	4,374	100.965	20.561	441.620	89.933
EHR-2.0×	4,039	111.839	22.716	451.717	91.750
EHR-2.5×	3,705	128.858	26.103	477.419	96.710
ISR	4,035	117.224	23.790	472.998	95.992
DemoSpeedup	3,744	126.962	25.735	475.345	96.352
APAS	4,422	**87.156**	**17.790**	**385.405**	**78.668**