TAR: Time-Adaptive Reweighting for Data-Efficient Vision-Language-Action Model Post-Training

***Abstract*—Supervised Vision-Language-Action (VLA) post-training typically aggregates per-action imitation losses with uniform temporal averaging, treating all actions equally despite success hinging on brief critical moments. To address this limitation, a method called Time-Adaptive Reweighting (TAR) is proposed. It changes only how action losses are aggregated over time during training, without altering the policy architecture or inference procedure. TAR derives a weight from each action's loss relative to other actions within the same segment, where segments are sampled from trajectories. It then replaces uniform temporal averaging with a weighted mean of the original per-action losses to form the segment loss. TAR improves the overall success rate by 3.04% on MetaWorld and 2.16% on LIBERO under the same demonstration data and evaluation protocol, averaged over five representative VLA models. It is also data-efficient by reducing horizon-normalized interaction steps under the same demonstration budget. The learned weights place more training emphasis on a small set of success-critical moments, which typically occur during early approach and around interaction onset (first contact and early interaction). For long-horizon multi-stage tasks, an additional peak can appear near completion. This shift in training emphasis enables a more effective policy, improving success rate and evaluation efficiency while making better use of limited demonstrations when data collection is costly.**

***Index Terms*—Vision-Language-Action model, behavior cloning, robot manipulation, loss weighting, data efficiency**

I. Introduction

Vision-Language-Action (VLA) models are proposed as generalist robot policies that map observations and task instructions to actions, enabling a single model to handle various manipulation tasks [1]–[8]. In practice, the VLA training is often limited by the cost of collecting high-quality demonstrations. Teleoperation and real-robot data collection are time-consuming and expensive, and they also require ongoing hardware maintenance and carefully safety management [9]–[12]. Even with simulation, building diverse and reliable demonstrations at scale still takes substantial engineering and curation effort [13]. When the demonstration budget is fixed, it becomes important to use each trajectory more effectively during training.

Not all actions in a manipulation demonstration provide the same training contribution. A typical trajectory contains routine actions, such as moving the end-effector toward the target and adjusting the pose, while only a few contact-centric interaction actions determine success (e.g. grasping, inserting, or releasing) [14]. For example, in a pick-and-place task, most actions move the gripper toward the object, but only a few actions during gripper closing and lifting decide whether the object is actually secured. Despite this imbalance, many post-training pipelines compute the loss of a trajectory segment by averaging action's losses uniformly [1], [7], [15]. As a result, routine actions can dominate the segment average, reducing the influence of interaction actions during learning.

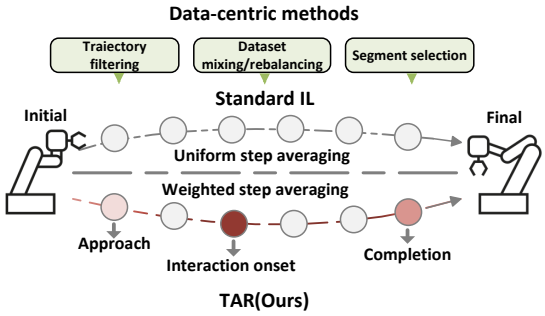

Fig. 1. Standard imitation learning uses uniform temporal averaging of per-action losses within a sampled segment. TAR replaces it with a weighted temporal average to emphasize a small set of success-critical actions. The highlighted phases (Approach, Interaction Onset, Completion) summarize the time windows where TAR most often assigns higher weights, as analyzed in Sec. IV.

Existing data-centric approaches improve imitation learning by modifying the training data. Typical choices include mixing datasets or rebalancing data sources [16]; removing low-quality demonstrations [17], [18]; and selecting trajectories or segments to reduce redundancy [19], [20]. These approaches are useful when demonstrations vary in quality or come from different sources. However, they usually keep the same training objective, where losses are still averaged uniformly over actions inside each sampled segment. This suggests a promising direction: keeping the demonstrations and model architecture fixed, while changing how much training emphasis is placed on different actions within a segment, as illustrated in Fig. 1.

To address this gap, Time-Adaptive Reweighting (TAR) adjusts how the loss of a trajectory segment is computed. Instead of treating all actions equally, TAR assigns each action a weight based on its imitation loss within the same segment. Time steps with larger imitation losses receive larger weights, while the weights are normalized so that their average over valid steps remains one. This keeps the overall loss scale of each segment unchanged, but shifts learning toward the actions that matter most for success. TAR can be applied to different VLA policy families and it only replaces uniform temporal averaging with a weighted average (architecture and inference remain unchanged). It supports common action prediction setups, including direct action regression and generative action models [8], [21]. Experiments on standard benchmarks indicate that TAR is data-efficient under a fixed demonstration budget, yielding higher overall success rates and

lower horizon-normalized interaction steps.

The main contributions of this paper are summarized as follows:

- This work quantifies success-critical temporal heterogeneity in per-action imitation losses and addresses it with TAR via loss-adaptive action weighting within each segment.
- TAR is proposed to compute loss-adaptive action's weights (unit-mean and capped) and to replace uniform temporal averaging with a weighted loss, without changing the policy architecture or inference.
- In MetaWorld [22] and LIBERO [23], TAR increases overall success rate by 3.04% and 2.16%, respectively, under the same protocol and demonstration budget. It is also data-efficient, reducing horizon-normalized interaction steps and yielding consistent gains across representative VLA models.

II. Method

A. Behavior Cloning and Temporal Averaging

Let $\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}$ be a set of expert demonstrations, where N is the number of trajectories. Each trajectory is $\tau^{(n)} = \{(\mathbf{o}_t^{(n)}, \mathbf{a}_t^{(n)}, \mathbf{x}^{(n)})\}_{t=1}^{T^{(n)}}$, where t is the time index and $T^{(n)}$ is the trajectory length. Here $\mathbf{o}_t^{(n)}$ denotes the observation, $\mathbf{a}_t^{(n)}$ denotes the expert action, and $\mathbf{x}^{(n)}$ denotes the language instruction (or task description).

Training extracts segments of length L from trajectories. If a segment is shorter than L, padding is applied and a binary mask $m_t^{(i)} \in \{0, 1\}$ marks valid steps (1) and padded steps (0) in segment i. The number of valid steps is $M^{(i)} = \sum_{t=1}^{L} m_t^{(i)}$. All temporal averages in this paper use $m_t^{(i)}$ to ignore padded steps.

Let f_θ be the policy with parameters θ. Let $\ell_t^{(i)}(\theta)$ be the imitation loss at step t of segment i, measuring the mismatch between the policy output and the expert action at that step.

Standard BC treats all valid steps equally and averages the step losses over time:

$$\bar{\ell}^{(i)}(\theta) = \frac{1}{M^{(i)}} \sum_{t=1}^{L} m_t^{(i)}\, \ell_t^{(i)}(\theta), \tag{1}$$

where $\bar{\ell}^{(i)}(\theta)$ is the segment-level loss under uniform temporal averaging.

In robot manipulation, the steps that determine success may occupy only a small portion of a trajectory segment. Uniform averaging can therefore reduce the influence of these success-sensitive steps when training data are limited. TAR addresses this issue by replacing the uniform temporal average in Eq. (1) with a weighted average.

B. Time-Adaptive Reweighting (TAR)

TAR replaces the uniform temporal average in Eq. (1) with a weighted average. An overview of the TAR pipeline is shown in Fig. 2. To avoid changing the overall loss scale of a segment, TAR normalizes the weights so that their average over valid steps equals one, i.e., $\frac{1}{M^{(i)}} \sum_{t=1}^{L} m_t^{(i)}\, w_t^{(i)} = 1$, where $w_t^{(i)} \geq 0$ is the weight at step t of segment i.

With TAR, the segment loss becomes

$$\bar{\ell}_{\mathrm{TAR}}^{(i)}(\theta) = \frac{1}{M^{(i)}} \sum_{t=1}^{L} m_t^{(i)}\, w_t^{(i)}\, \ell_t^{(i)}(\theta), \tag{2}$$

where $\ell_t^{(i)}(\theta)$ is the step imitation loss defined in Sec. II-A, $m_t^{(i)}$ masks padded steps, and $M^{(i)} = \sum_{t=1}^{L} m_t^{(i)}$ is the number of valid steps. Eq. (2) has the same form as Eq. (1), except that TAR multiplies each valid step loss by a loss-adaptive time-step weight.

C. Loss-based Weight Construction

This subsection defines how TAR computes step weights $w_t^{(i)}$ from the step losses within a segment. The weights satisfy two basic properties: (i) $w_t^{(i)} \geq 0$ for valid steps, and (ii) the average weight over valid steps equals one. The construction is performed independently for each segment.

Stop gradient on the weighting branch. This is achieved by applying a stop gradient operator:

$$\tilde{\ell}_t^{(i)} = \mathrm{sg}\Big(\ell_t^{(i)}(\theta)\Big), \tag{3}$$

where $\ell_t^{(i)}(\theta)$ is the imitation loss at step t of segment i, $\mathrm{sg}(\cdot)$ blocks gradient flow, and $\tilde{\ell}_t^{(i)}$ is used only to compute weights (it does not create an additional gradient path to θ).

Segment local normalization. Loss scales differ across tasks and action heads; therefore TAR relies on within-segment normalized losses and uses only each time step's relative deviation from the segment average. Define the masked mean and standard deviation of the detached losses:

$$\mu^{(i)} = \frac{1}{M^{(i)}} \sum_{t=1}^{L} m_t^{(i)}\, \tilde{\ell}_t^{(i)},$$
$$\sigma^{(i)} = \sqrt{\frac{1}{M^{(i)}} \sum_{t=1}^{L} m_t^{(i)} \left(\tilde{\ell}_t^{(i)} - \mu^{(i)}\right)^2}, \tag{4}$$

where $\mu^{(i)}$ and $\sigma^{(i)}$ are the masked mean and std of $\tilde{\ell}_t^{(i)}$. A normalized score $s_t^{(i)}$ is then computed for each step:

$$s_t^{(i)} = \frac{\tilde{\ell}_t^{(i)} - \mu^{(i)}}{\sigma^{(i)} + \varepsilon}, \tag{5}$$

where $\varepsilon > 0$ is a small constant for numerical stability. The score $s_t^{(i)}$ indicates whether step t has a larger (positive) or smaller (negative) loss than the segment average.

From scores to a segment distribution. The scores are converted into a distribution over time steps using a masked softmax:

$$p_t^{(i)} = \frac{m_t^{(i)} \exp\Big(\gamma s_t^{(i)}\Big)}{\sum_{k=1}^{L} m_k^{(i)} \exp\Big(\gamma s_k^{(i)}\Big)}, \tag{6}$$

where $\gamma > 0$ controls how concentrated the distribution is. The quantity $p_t^{(i)}$ represents the fraction of the segment-level emphasis assigned to step t before scaling: it satisfies $p_t^{(i)} \geq 0$ and $\sum_{t=1}^{L} p_t^{(i)} = 1$ over valid steps. The mask enforces $p_t^{(i)} = 0$ on padded steps.

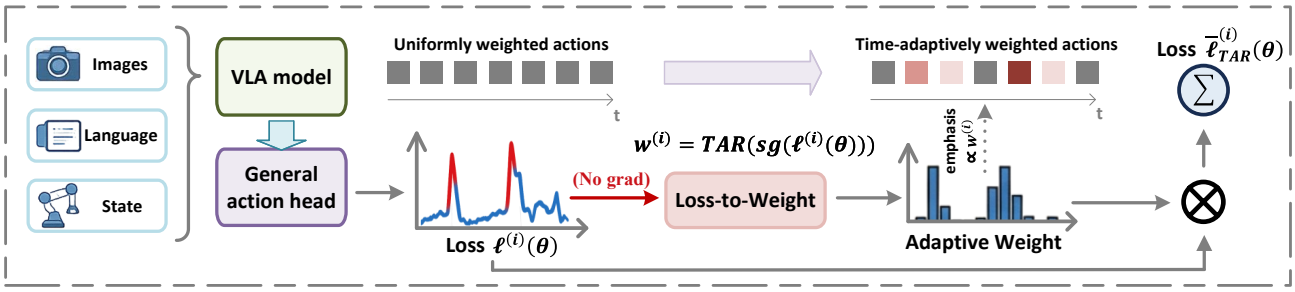

Fig. 2. **Time-adaptive reweighting pipeline.** A VLA model takes image, language, and state as input and produces per-step imitation losses $\ell_t^{(i)}(\theta)$ on a sampled segment. Standard imitation learning aggregates these losses with a uniform temporal average and can under-emphasize the few trajectory portions that are difficult to imitate and often determine success. TAR computes adaptive time-step weights $w_t^{(i)}$ from the within-segment loss pattern by applying stop-gradient to the per-step losses and mapping them to weights (no gradient flows through the weight branch). The segment loss is then formed by a weighted temporal reduction of the original per-step losses, $\bar{\ell}^{(i)}_{\mathrm{TAR}}(\theta) = \frac{1}{M^{(i)}} \sum_t m_t^{(i)} w_t^{(i)} \ell_t^{(i)}(\theta)$; training therefore places more emphasis on time steps that remain hard under uniform averaging, while keeping the model architecture and inference unchanged.

Unit-mean weights and a concentration cap. A provisional weight is obtained by scaling $p_t^{(i)}$ so that the average weight over valid steps equals one:

$$\tilde{w}_t^{(i)} = M^{(i)} p_t^{(i)}, \tag{7}$$

To reduce overly concentrated emphasis on a small number of steps, the provisional weights are first clipped at a threshold $w_{\max} > 0$ and then renormalized to preserve the unit-mean property:

$$\begin{aligned} \hat{w}_t^{(i)} &= \min\left(\tilde{w}_t^{(i)}, w_{\max}\right), \\ w_t^{(i)} &= \frac{M^{(i)} m_t^{(i)} \hat{w}_t^{(i)}}{\sum_{k=1}^{L} m_k^{(i)} \hat{w}_k^{(i)}}, \end{aligned} \tag{8}$$

where $w_t^{(i)}$ is the final step weight used in Eq. (2). By construction, $w_t^{(i)} \geq 0$, padded steps have $w_t^{(i)} = 0$, and the unit-mean property holds over valid steps: $\frac{1}{M^{(i)}} \sum_{t=1}^{L} m_t^{(i)} w_t^{(i)} = 1$.

D. Applying TAR to Different Action Heads

TAR applies to a broad range of action heads. It only requires a per-step training loss $\ell_t^{(i)}(\theta)$ for each valid step in a sampled segment. Once $\ell_t^{(i)}(\theta)$ is available, TAR applies the time-step weights $w_t^{(i)}$ and performs the weighted temporal reduction in Eq. (2).

Regression action heads. For standard regression heads, $\ell_t^{(i)}(\theta)$ is the per-step supervised prediction error (e.g., MSE under continuous actions). If the head predicts an action chunk or a short horizon, a scalar $\ell_t^{(i)}(\theta)$ is formed by averaging the supervised errors over the predicted horizon. This covers common VLA post-training setups with linear/regression heads (e.g., Octo with a linear action head) and chunked supervision (e.g., ACT-style training) [7], [15].

Generative action heads (diffusion / flow matching). For generative heads, $\ell_t^{(i)}(\theta)$ is taken directly from the head's standard training objective at step t. For diffusion-based policies, it corresponds to the denoising (score-matching) loss; for flow-matching policies, it corresponds to the regression loss. TAR does not alter these objectives and only reweights across time steps via Eq. (2) [24], [25].

III. Experiments

This section evaluates TAR in manipulation along three aspects:

- whether success rate gains are consistent across commonly used VLA policy families and action heads;
- how gains distribute across benchmark-defined difficulty levels and suites;
- whether interaction efficiency improves under metrics that remain comparable across tasks with different horizon caps.

A. Benchmarks, metrics, and evaluation protocol

1) Benchmarks: Two standard manipulation benchmarks are used: MetaWorld [22] and LIBERO [23]. MetaWorld contains 50 tasks grouped by difficulty *(Easy/Medium/Hard/Very Hard)* [26] with 28/11/6/5 tasks per group, respectively. LIBERO consists of four suites: *Spatial*, *Object*, *Goal*, and *LIBERO-10*; each suite contains 10 tasks (40 tasks total), and *LIBERO-10* typically involves longer multi-stage interaction chains.

For each rollout, let s denote the executed environment steps until termination or truncation, and let H denote the task-specific step budget (MetaWorld: H=500; LIBERO suites: H=280/280/300/520 for Spatial/Object/Goal/LIBERO-10). Here H is chosen per suite to exceed the longest trajectory length observed under the evaluation protocol, while remaining as tight as possible to reduce redundant truncation headroom. Define the normalized step fraction $\rho = s/H$, which lies in $(0, 1]$ under this evaluation protocol.

2) Metrics:

a) Success Rate.: Success rate (%) is the primary metric. To quantify interaction efficiency while avoiding ambiguity introduced by different horizon caps, two normalized step-based metrics are reported: Fast20 and NEPS.

b) Fast20.: Fast20 measures the mean normalized step fraction of the fastest 20% rollouts (lower is better), without conditioning on success. For a given task with $N{=}50$ rollouts, compute ρ for each rollout; sort the N values in ascending order; and average the smallest $K = \lceil 0.2N \rceil = 10$ values: Fast20$= \frac{1}{K}\sum_{j=1}^{K} \rho_{(j)}$. Because failures in these benchmarks typically consume the full horizon, their normalized fractions are close to 1 (i.e., close to 100% when reported). Fast20 therefore worsens (increases) whenever near-horizon rollouts enter the top-K fastest set. For readability, Fast20 is reported in percent form throughout, i.e., $100\times$Fast20

c) NEPS.: NEPS (Normalized Expected Steps per Success) measures normalized interaction cost per achieved success: NEPS $= \frac{\mathbb{E}[\rho]}{r}$, where $\mathbb{E}[\rho]$ averages ρ over all rollouts and r denotes the success rate as a fraction (not percent). Equivalently, for N rollouts with S successes, NEPS $= \frac{\sum_{i=1}^{N} \rho_i}{S}$, explicitly counting the step cost of failures (which are typically close to 1, i.e., close to 100% when reported). Lower values indicate fewer normalized interaction steps per successful completion. For readability, NEPS is reported in percent form throughout, i.e., $100\times$NEPS

3) Protocol: Unless stated otherwise, results are reported at a unified training checkpoint of 100K steps. Each task is evaluated with 50 independent rollouts under the same success criterion across all methods. All runs use a fixed random seed of 1000 for training and evaluation. For MetaWorld, the overall column averages over all 50 tasks (difficulty-group columns average within each group). For LIBERO, the overall column averages over all 40 tasks (10 per suite). For LIBERO efficiency metrics, suite-wise normalization by the corresponding H is applied before averaging to form overall.

4) Baseline Models: Representative VLA models with public training pipelines are evaluated to support reproducible comparisons. For each model, TAR keeps the same architecture and inference procedure and only changes the temporal reduction.

- **ACT** [15]. A transformer behavior cloning model with direct action regression. It is selected because it is a widely used and reproducible regression-head baseline, providing a clear reference point for standard imitation post-training.
- **Octo (Linear and Diffusion)** [7]. A generalist VLA model evaluated with two action heads: linear regression and diffusion. It is selected because the shared backbone with two heads makes action parameterization a controlled factor, enabling a direct comparison of TAR under regression versus diffusion without changing the backbone.
- **SmolVLA** [27]. A compact VLA model trained with a lightweight public dataset setting. It is selected because it represents a practical small-model regime with efficient training, testing whether TAR remains effective.
- π_0 [8]. A strong open foundation VLA model with high capacity. It is selected because it is a competitive modern baseline with substantially larger model capacity, testing whether TAR transfers to larger-scale VLA models while keeping inference unchanged.

B. TAR integration and training details

TAR changes only the temporal reduction used to aggregate per-step imitation losses (replacing the uniform mean in Eq. (1) with the weighted mean in Eq. (2)), leaving model architectures and inference unchanged. TAR is enabled after 20K training steps. Over the next 30K steps, the reduction is smoothly transitioned to TAR weighting using a mixing coefficient $\alpha \in [0, 1]$: $\tilde{w}_t \leftarrow (1{-}\alpha){\cdot}1{+}\alpha{\cdot}\tilde{w}_t$. After mixing, the same max-cap post-processing and unit-mean renormalization in Eq. (8) is applied to obtain the final weights used in Eq. (2). A fixed $w_{\max}$ is used across runs. Other training settings follow the official implementations of each policy.

C. Main results

1) TAR Improves Overall Success Rate on MetaWorld and LIBERO Across All Evaluated Policies: Tables I and II summarize success rates together with normalized efficiency metrics. Across all evaluated policy families and both benchmarks (10 policy-benchmark pairs), TAR increases the overall success rate in all cases. On MetaWorld, the average overall success rate gain is $+3.04\%$ (min/max: $+1.04\%$ to $+4.84\%$). On LIBERO, the average overall success rate gain is $+2.16\%$ (min/max: $+1.30\%$ to $+3.15\%$). These gains hold across regression-based heads (ACT, Octo linear), diffusion-based action modeling (Octo diffusion), compact recipes (SmolVLA), and a strong foundation policy family (π_0), indicating that TAR improves task completion reliability under a fixed demonstration budget across diverse action heads.

2) Gains Concentrate on Harder or Longer-Horizon Task Groups (MetaWorld Hard/Very Hard and LIBERO-10): Success rate gains are not uniform across benchmark groups, and the distribution is consistent across policy families. Here, "harder" corresponds to MetaWorld *Hard/Very Hard* tasks, which require more complex manipulation under the benchmark difficulty grouping [26], while "longer-horizon" corresponds to LIBERO *LIBERO-10*, which involves longer multi-stage task compositions [23]. As a result, errors are more likely to accumulate and a few success-critical moments have larger influence, so TAR tends to yield larger gains in these regimes.

On MetaWorld, the average success rate gain increases with difficulty. Averaged over 5 policy families, success rate improves by $+1.20\%$ (Easy), $+4.51\%$ (Medium), $+6.93\%$ (Hard), and $+5.44\%$ (Very Hard). Moreover, success rate improves in all 5 policy families for Medium/Hard/Very Hard, and in 4 out of 5 families for Easy.

On LIBERO, the largest average uplift appears on the long-horizon *LIBERO-10* suite. Averaged over 5 policy families, *LIBERO-10* success increases by $+6.08\%$, and all 5 families improve on *LIBERO-10*. The other suites show smaller changes: Spatial improves by $+2.36\%$ on average (4/5 families improve), Goal improves by $+1.40\%$ on average (4/5 families improve), while Object is mixed with an average change of -1.20% (1/5 families improve). Despite suite-level variation, the consistent and largest uplift on *LIBERO-10* keeps the overall success rate gains positive across all families.

TABLE I

METAWORLD RESULTS. SUCCESS RATE (%) IS REPORTED FOR EACH DIFFICULTY GROUP AND OVERALL. DATA-EFFICIENT INTERACTION IS MEASURED BY HORIZON-NORMALIZED INTERACTION STEPS (LOWER IS BETTER), REPORTED AS 100×FAST20 AND 100×NEPS. FOR EACH MODEL, THE BETTER OF BASELINE AND TAR IS HIGHLIGHTED.

Model	**Success rate (%)**					**Data-efficient steps (lower)**	
	Easy	**Medium**	**Hard**	**Very Hard**	**Overall**	**Fast20**	**NEPS**
ACT [15]	81.29	69.64	*58.67*	69.60	74.84	18.68	50.68
ACT (TAR)	**85.29**	**73.45**	***68.67***	**75.20**	**79.68**	**17.05**	**44.62**
SmolVLA [27]	**79.93**	45.45	*42.00*	43.60	64.16	19.63	62.09
SmolVLA (TAR)	78.64	**54.91**	***56.67***	**52.40**	**68.16**	**18.72**	**57.19**
Octo (Linear) [7]	79.86	76.36	71.33	*78.00*	77.88	17.31	48.47
Octo (Linear, TAR)	**82.14**	**79.27**	**76.00**	***85.60***	**81.12**	**15.88**	**41.89**
Octo (Diffusion) [7]	80.14	80.36	76.00	*79.20*	79.60	16.13	44.68
Octo (Diffusion, TAR)	**80.36**	**82.73**	**76.67**	***82.40***	**80.64**	**15.53**	**43.85**
π_0 [8]	86.36	69.64	*73.67*	82.40	80.76	15.04	40.38
π_0 (TAR)	**87.14**	**73.64**	***78.33***	**84.40**	**82.84**	**14.64**	**37.92**

Note. Fast20 and NEPS are reported in percent form, i.e., 100×Fast20 and 100×NEPS. Lower is better. Within each baseline/TAR pair, the difficulty group with the largest TAR gain is additionally italicized.

TABLE II

LIBERO RESULTS. SUCCESS RATE (%) IS REPORTED FOR EACH SUITE AND OVERALL. DATA-EFFICIENT INTERACTION IS MEASURED BY HORIZON-NORMALIZED INTERACTION STEPS (LOWER IS BETTER), REPORTED AS 100×FAST20 AND 100×NEPS. SUITE-WISE NORMALIZATION USES H=280/280/300/520 FOR SPATIAL/OBJECT/GOAL/LIBERO-10 BEFORE FORMING OVERALL. FOR EACH MODEL, THE BETTER OF BASELINE AND TAR IS HIGHLIGHTED.

Model	**Success rate (%)**					**Data-efficient steps (lower)**	
	Spatial	**Object**	**Goal**	**LIBERO-10**	**Overall**	**Fast20**	**NEPS**
ACT [15]	*64.00*	36.20	**6.20**	34.80	35.30	47.52	228.81
ACT (TAR)	***67.60***	**38.80**	4.20	**38.00**	**37.15**	**46.49**	**207.93**
SmolVLA [27]	85.00	**93.00**	89.20	*59.40*	81.65	40.76	55.07
SmolVLA (TAR)	**89.00**	91.60	**90.40**	***68.00***	**84.75**	**39.65**	**53.36**
Octo (Linear) [7]	76.60	**83.80**	85.40	*47.80*	73.40	35.36	80.18
Octo (Linear, TAR)	**82.20**	80.00	**87.80**	***56.20***	**76.55**	**34.35**	**72.22**
Octo (Diffusion) [7]	79.00	**85.80**	84.60	*51.20*	75.15	36.18	77.48
Octo (Diffusion, TAR)	**80.60**	83.20	**86.80**	***55.20***	**76.45**	**34.39**	**75.76**
π_0 [8]	**97.40**	**98.80**	95.80	*86.00*	94.50	36.09	52.49
π_0 (TAR)	94.40	98.00	**99.00**	***92.20***	**95.90**	**34.63**	**51.17**

Note. Fast20 and NEPS are reported in percent form, i.e., 100×Fast20 and 100×NEPS. Lower is better. Within each baseline/TAR pair, the suite with the largest TAR gain is additionally italicized.

Taken together, the results indicate that TAR is most beneficial on tasks that require longer rollouts and exhibit higher difficulty under the benchmark definitions (MetaWorld Hard/Very Hard and LIBERO-10). This benchmark-level concentration is summarized in Fig. 3(c), which aggregates group-level Δ success rate together with the corresponding change in Fast20. In such tasks, the policy benefits more from reallocating optimization emphasis across time within a fixed-length segment, leading to larger and more consistent success improvements.

3) Interaction Efficiency Improves Under Comparable Horizon-Normalized Metrics (Fast20 and NEPS): The success improvements are accompanied by lower interaction cost at evaluation, suggesting that TAR learns more effective policies rather than merely altering evaluation outcomes. To quantify interaction cost in a way that remains comparable across tasks with different horizon caps, two normalized step-based measures are reported: Fast20 and NEPS (both lower is better).

On MetaWorld, TAR reduces Fast20 (overall) by 0.99 on average (range: 0.40 to 1.63), and reduces NEPS (overall) by 4.17 on average (range: 0.83 to 6.58) across the five policy families. On LIBERO, TAR reduces Fast20 (overall) by 1.28 on average (range: 1.01 to 1.79), and reduces NEPS (overall) by 6.72 on average (range: 1.32 to 20.88) across the five families.

Across both benchmarks, all 10 policy-benchmark pairs achieve higher overall success together with lower Fast20 and NEPS, showing that TAR improves reliability and reduces evaluation-time interaction cost at the same time. Fig. 3(a,b) visualize this joint movement toward higher success rate and lower cost, and Fig. 4 summarizes the same trend across benchmark-defined groups and suites.

IV. DISCUSSION

This section discusses two aspects. First, it tests whether TAR's improvements depend on assigning larger weights to the appropriate actions within each training segment, rather than benefiting from non-uniform weighting in a generic way. Second, it summarizes what the recorded weights look like along trajectories and explains how these patterns can inform demonstration collection under a fixed budget. The summary includes approach, interaction onset and completion-critical moments in long-horizon, multi-stage tasks.

All variants in this section follow the same data, training schedule, and hyperparameter configuration as the main ex-

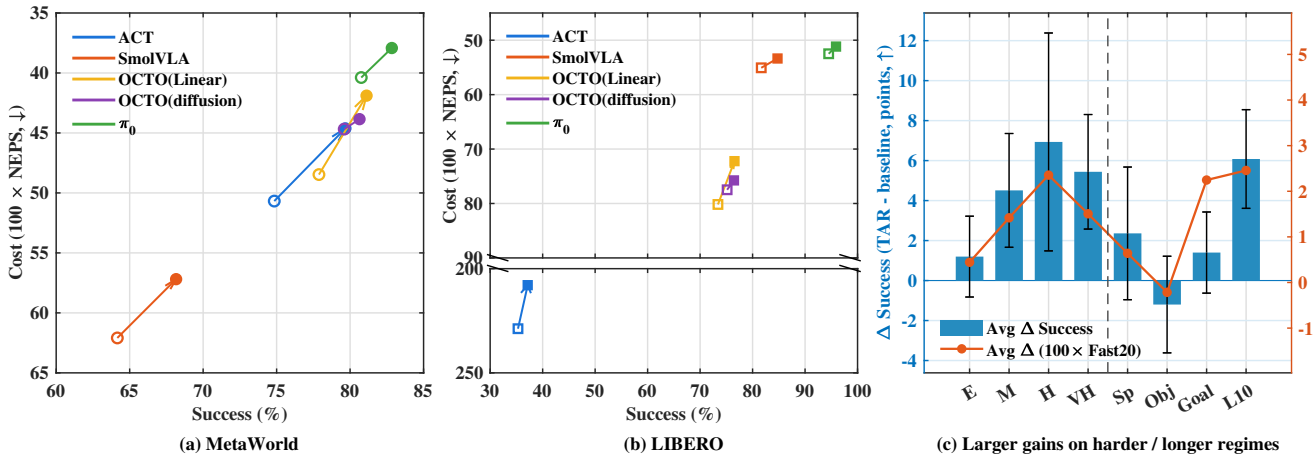

Fig. 3. **Performance–efficiency overview with subset-wise effects.** (a,b) Each policy family is shown as a paired marker: the unfilled marker denotes the original training recipe and the filled marker denotes the TAR variant; the arrow indicates the shift from baseline to TAR. The x-axis reports success rate (%; ↑ is better). The y-axis reports normalized deployment cost (100×NEPS; ↓ is better), and is plotted with an inverted direction so that upward movement indicates lower cost. Panel (b) further enlarges the upper-right region for readability. (c) Subset-level differences aggregated across policy families: bars report Δ success rate (TAR minus baseline, percentage points; ↑ is better) with error bars indicating across-family variability, while the curve reports the paired change in the interaction-efficiency score on the fastest rollouts, Δ(100×Fast20) (baseline minus TAR; ↑ indicates faster / lower interaction cost).

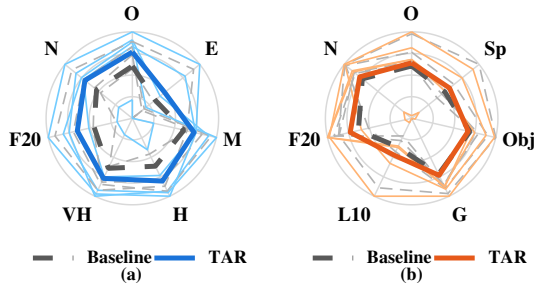

Fig. 4. **Compact multi-axis overview (normalized).** Radar plots summarize MetaWorld (left) and LIBERO (right) across all evaluated policy families. Dashed gray traces denote Baseline and solid colored traces denote TAR; faint polygons show per-policy outlines, and thick traces indicate the across-policy mean. Each spoke is independently rescaled for visualization, and efficiency spokes are inverted so that larger values correspond to lower interaction cost. Abbreviations: O=Overall success rate; for MetaWorld, E=Easy, M=Medium, H=Hard, VH=Very Hard; for LIBERO, Sp=Spatial, Obj=Object, G=Goal, L10=LIBERO-10. Efficiency spokes are F20=100×Fast20 and N=100×NEPS.

periments. Success rate (%) is reported as the sole metric, since the goal of this section is to validate the correctness of the weighting strategy itself and success rate is sufficient and directly comparable.

A. *Effectiveness and Correctness of the Weighting Strategy*

1) Control Variants.: All control variants modify only how per-action weights are assigned within each training segment. Shuffle-weights keeps the same set of TAR weights but permutes their temporal locations within each segment. random-keep assigns random weights (followed by the same normalization procedure) to remove any structure tied to time. invert-keep reverses TAR's temporal emphasis pattern, placing larger weights where TAR places smaller weights. For the perturbation test, Gaussian noise is injected into action supervision at a matched perturbation ratio and noise scale: perturb-high perturbs actions that TAR would upweight, while perturb-low perturbs actions that TAR would downweight.

2) Gains Depend on Temporal Weight Placement.: Relative to the baseline, TAR improves overall success rate on both benchmarks (+4.00% on MetaWorld and +3.10% on LIBERO), making it the only variant that increases overall success rate in both settings. The improvement concentrates on the harder or longer-horizon parts of the benchmarks: averaged over MetaWorld (Medium/Hard/Very Hard) and LIBERO-10, TAR yields a mean gain of +10.38%. By comparison, keeping the same weights but shuffling their time locations yields only +1.07% on the same subset, and random weights yield +0.31%, indicating that non-uniform weighting alone is not sufficient.

Across the two overall columns, TAR achieves a mean gain of +3.55%. The strongest control under the same averaging (+0.14%) does not improve both benchmarks simultaneously, and TAR exceeds the best control by +3.48% on MetaWorld overall and +1.90% on LIBERO overall. invert-keep further shows that reversing the temporal emphasis is harmful on average, consistent with the view that the time placement of larger weights matters.

3) Perturbation Validates the Importance of Upweighted actions.: With the same perturbation budget, corrupting supervision at TAR-upweighted actions is more damaging than corrupting supervision at TAR-downweighted actions. Across the full set of 10 reported groups/suites, perturb-high is lower than perturb-low by 2.17% on average. The gap is larger on the harder/longer subset (MetaWorld Medium/Hard/Very Hard and LIBERO-10), where perturb-high underperforms perturb-low by 5.73% on average, with the largest separation occurring on

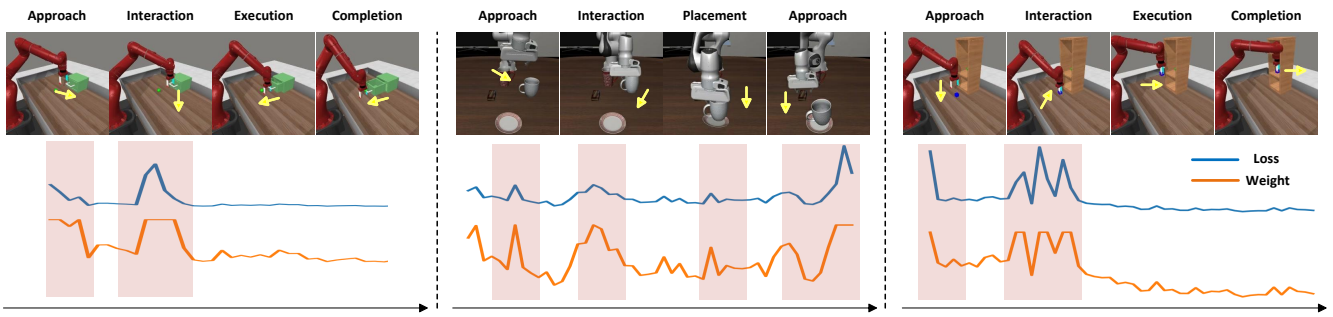

Fig. 5. **Trajectory-level weight visualization.** Three representative rollouts are segmented into phases and plotted with per-action imitation loss and TAR weight. Shaded regions indicate high-weight time windows. Higher weights often concentrate during approach and at interaction onset. For long-horizon, multi-stage tasks (middle), an additional high-weight window can appear near completion, where final placement/insertion/release actions are completion-critical. Routine execution actions are typically down-weighted.

TABLE III

CONTROLS ON SMOLVLA (SUCCESS RATE, %). THE METAWORLD COLUMNS REPORT DIFFICULTY-GROUP SUCCESS RATE AND OVERALL; THE LIBERO COLUMNS REPORT SUITE SUCCESS RATE AND OVERALL. HIGHER IS BETTER; THE BEST NUMBER IN EACH COLUMN IS HIGHLIGHTED IN BOLD.

Approach	**MetaWorld success rate (%)**					**LIBERO success rate (%)**				
	Easy	**Medium**	**Hard**	**Very Hard**	**Overall**	**Spatial**	**Object**	**Goal**	**LIBERO-10**	**Overall**
SmolVLA [27]	79.93	45.45	42.00	43.60	64.16	85.00	**93.00**	89.20	59.40	81.65
Shuffle-weights	78.93	43.82	43.33	42.00	63.24	**89.00**	86.80	90.00	65.60	82.85
random-keep	**80.36**	45.09	44.00	44.80	64.68	83.00	91.60	87.00	57.80	79.85
invert-keep	80.00	43.45	37.33	26.00	61.44	84.80	92.40	83.00	51.60	77.95
perturb-low	79.36	46.00	42.67	40.40	63.72	84.00	90.60	85.00	59.60	79.80
perturb-high	80.14	45.27	37.67	35.20	62.88	85.80	91.60	85.60	47.60	77.65
SmolVLA (TAR)	78.64	**54.91**	**56.67**	**52.40**	**68.16**	**89.00**	91.60	**90.40**	**68.00**	**84.75**

LIBERO-10 (perturb-low is 12.00% higher than perturb-high). This pattern is consistent with TAR assigning higher weights to time windows that are more strongly tied to task success.

B. Trajectory Visualization and Practical Implications

This section visualizes the per-action weights produced by TAR to clarify what the weighting strategy emphasizes during training, and representative examples are shown in Fig. 5. Across the evaluated rollouts, the weight curves show the high-weight windows rather than uniform emphasis over time. Higher weights frequently align with (i) early approach that requires precise alignment toward the target and (ii) interaction onset around first contact and early manipulation. For long-horizon, multi-stage tasks, an additional high-weight window can appear near completion, where the final placement/insertion/release actions determine whether the rollout finishes successfully. Outside these windows, routine execution actions receive smaller weights.

The same patterns suggest a practical guideline for demonstration collection under a fixed budget. If only a limited number of demonstrations can be obtained, improving data quality and coverage around the upweighted windows is expected to be more beneficial than uniformly increasing data elsewhere. Concrete actions include stabilizing teleoperation around interaction onset; ensuring clearer observations during contact-sensitive moments (e.g., reducing occlusion and motion blur); and increasing diversity in initial configurations and contact geometries. For multi-stage tasks, maintaining higher-quality supervision near completion is also important, since small errors in the final actions can determine success.

V. CONCLUSION

This study proposes TAR to address a common mismatch in supervised VLA post-training with imitation learning: per-action losses within a sampled segment are typically averaged uniformly, even though only a small subset of time steps is typically more important than the rest. TAR derives adaptive time-step weights from the within-segment loss pattern (stop-gradient, unit-mean normalization, and a concentration cap) to place more training emphasis on success-critical moments. Across MetaWorld and LIBERO, TAR consistently improves overall success rate while improving horizon-normalized interaction efficiency. These results suggest that reallocating optimization emphasis across time is an effective way to increase the return of a fixed demonstration budget. By emphasizing actions that remain hard under uniform averaging and are more tightly coupled to success (e.g., approach, interaction onset and completion-critical actions in long-horizon tasks), TAR improves data utilization without requiring additional data curation or architectural changes. The resulting weight patterns also provide a practical signal for data collection and refinement: higher-quality supervision and broader coverage around the upweighted windows are more likely to yield benefits than uniformly increasing effort elsewhere. Future work will validate TAR on real-robot post-training and examine robustness under sensing noise and distribution shifts.

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal *et al.*, "RT-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[2] B. Zitkovich, T. Yu, S. Xu *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023, pp. 2165–2183.

[3] D. Driess, F. Xia, M. S. M. Sajjadi *et al.*, "PaLM-E: An embodied multimodal language model," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023, pp. 8469–8488, arXiv:2303.03378.

[4] M. Ahn, A. Brohan, N. Brown *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[5] Y. Jiang, A. Gupta, Z. Zhang *et al.*, "VIMA: General robot manipulation with multimodal prompts," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023, arXiv:2210.03094.

[6] M. J. Kim, K. Pertsch, S. Karamcheti *et al.*, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[7] O. M. Team, D. Ghosh, H. Walke *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[8] K. Black, N. Brown, D. Driess *et al.*, "
pi_0: A Vision-Language-Action Flow Model for General Robot Control," *arXiv preprint arXiv:2410.24164*, 2024.

[9] A. Mandlekar, Y. Zhu, A. Garg *et al.*, "RoboTurk: A crowdsourcing platform for robotic skill learning through imitation," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2018, pp. 879–893.

[10] S. Dasari, F. Ebert, S. Tian *et al.*, "RoboNet: Large-scale multi-robot learning," *arXiv preprint arXiv:1910.11215*, 2019.

[11] H. R. Walke, K. Black, T. Z. Zhao *et al.*, "BridgeData V2: A dataset for robot learning at scale," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023, pp. 1723–1736.

[12] A. Khazatsky, K. Pertsch, S. Nair *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," *arXiv preprint arXiv:2403.12945*, 2024.

[13] A. O'Neill, A. Rehman, A. Maddukuri *et al.*, "Open X-Embodiment: Robotic learning datasets and RT-X models," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6892–6903.

[14] T. Tsuji, Y. Kato, G. Solak *et al.*, "A survey on imitation learning for contact-rich tasks in robotics," *arXiv preprint arXiv:2506.13498*, 2025.

[15] T. Z. Zhao, V. Kumar, S. Levine *et al.*, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[16] J. Hejna, C. Bhateja, Y. Jiang *et al.*, "Re-Mix: Optimizing data mixtures for large-scale imitation learning," *arXiv preprint arXiv:2408.14037*, 2024.

[17] S. Belkhale, Y. Cui, and D. Sadigh, "Data quality in imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 80 375–80 395.

[18] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 627–635.

[19] Y. Zhang, Y. Xie, H. Liu *et al.*, "SCIZOR: A self-supervised approach to data curation for large-scale imitation learning," *arXiv preprint arXiv:2505.22626*, 2025.

[20] J. Chen, H. Fang, H.-S. Fang *et al.*, "Towards effective utilization of mixed-quality demonstrations in robotic manipulation via segment-level selection and optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 16 884–16 891.

[21] C. Chi, Z. Xu, S. Feng *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[22] T. Yu, D. Quillen, Z. He *et al.*, "Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2020, pp. 1094–1100, arXiv:1910.10897.

[23] B. Liu, Y. Zhu, C. Gao *et al.*, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 44 776–44 791, arXiv:2306.03310.

[24] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 6840–6851.

[25] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu *et al.*, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[26] Y. Seo, D. Hafner, H. Liu *et al.*, "Masked world models for visual control," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023, pp. 1332–1344, arXiv:2206.14244.

[27] M. Shukor, D. Aubakirova, F. Capuano *et al.*, "SmolVLA: A vision-language-action model for affordable and efficient robotics," *arXiv preprint arXiv:2506.01844*, 2025.