HERA: HIERARCHICAL RESPONSE-AWARE SUPERVISION ALLOCATION FOR EFFICIENT VLA FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models provide a foundation for robotic manipulation. Fine-tuning on demonstrations collected in target environments enables adaptation, but can require lengthy training. One source of this inefficiency is repeatedly training on demonstrations with little additional benefit at the current learning state. Existing data-selection methods concentrate training on useful supervision through subsets selected by behavioral structure or estimated utility. Yet fixed subsets cannot track changing utility: retained samples may become less useful, while excluded samples remain unavailable when their value grows. *Our key insight is that training utility evolves at two coupled granularities: learning from one action chunk can change the value of physically related chunks, while the remaining benefits within a chunk can evolve differently, or oppositely, across its action subspaces.* These effects depend on trajectory structure and the current learning state. We introduce **HERA**, a **H**i**E**rarchical **R**esponse-**A**ware Supervision Allocation framework for flow-based VLA fine-tuning. A shared response estimator combines physical correspondences with the current learning state to predict changes to a fixed training-side reference objective, sharing sparse probe responses across related chunks. Dynamic chunk sampling adapts training frequencies while keeping all chunks eligible. Dynamic action-subspace weighting assigns normalized, chunk-specific weights over native action subspaces. Both allocations are jointly optimized and periodically refreshed. On LIBERO, HERA reaches 80% success with 53.6% fewer optimizer updates than LoRA, yielding a $1.93\times$ wall-clock fine-tuning speedup including response-estimation and allocation overhead. At 20K updates, HERA improves average success rate over LoRA by 9.38 and 10.75 percentage points on LIBERO and PiPER-X, respectively.

1 INTRODUCTION

Vision-language-action (VLA) models unify visual perception, language understanding, and action generation, providing a foundation for general-purpose robotic manipulation. Recent advances, including RT-2, OpenVLA, the π family, and GR00T, demonstrate the potential of large-scale pre-training for diverse robot tasks (Zitkovich et al., 2023; Kim et al., 2025; Physical Intelligence et al., 2025; NVIDIA et al., 2025). However, reliable execution in a specific deployment environment can require adapting to unfamiliar visual conditions, embodiments, and task requirements. Fine-tuning on demonstrations collected in the target setting provides a way to bridge this gap (Dass et al., 2026; Kim et al., 2026). Yet obtaining a reliable adapted policy can require lengthy fine-tuning, delaying deployment and slowing subsequent cycles of evaluation and improvement. We therefore aim to accelerate the fine-tuning of flow-based VLAs by reducing the total training time required to reach a given task success rate.

Conventional VLA fine-tuning repeatedly processes demonstration batches through forward and backward passes. For the flow-based policies considered here, each training sample supervises a short sequence of future actions, termed an *action chunk*. The total time to the target success rate depends both on the cost of each update and on how many updates are needed. Yet revisited chunks can offer little additional benefit at the current learning state, consuming computation with limited progress toward that target. A natural response is *data selection*: concentrating training

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

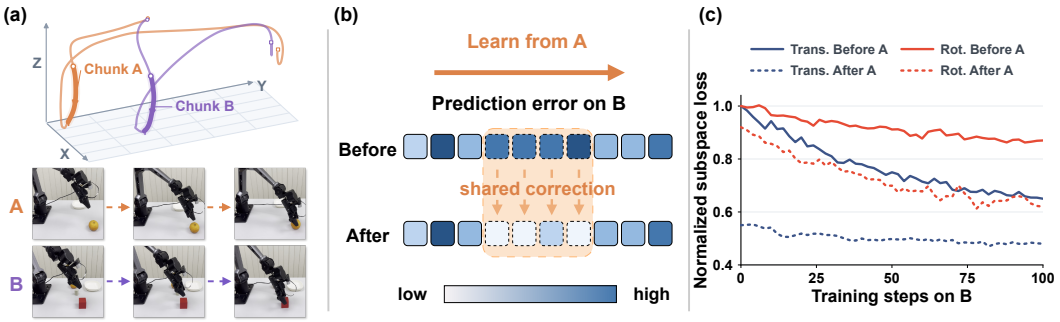

Figure 1: **Motivation of HERA.** Learning from one action chunk can change local prediction errors on physically related chunks and alter the benefits of further full-chunk training differently across action subspaces.

on demonstrations that provide useful supervision. SIEVE and DataMIL pursue this direction by constructing subsets based on behavioral structure or estimated training utility (Wu et al., 2026; Dass et al., 2026). However, their selected subsets retain fixed membership during subsequent policy training, while the value of training each chunk evolves with the policy. Retained chunks can become less useful, whereas excluded chunks cannot be revisited when their utility increases. The challenge is therefore *to reassess which chunks remain useful as training progresses, at a cost low enough to yield a net reduction in training time*.

To understand these changing learning needs, we draw inspiration from human motor learning, where shared structure across motor experiences can facilitate learning related tasks (Braun et al., 2009). We therefore examine how training on one action chunk affects physically related chunks during VLA fine-tuning. We find two key observations. First, *learning from one action chunk can change the local prediction errors and the subsequent training benefit of physically related chunks*. Second, an action chunk provides supervision over distinct *action subspaces*, such as translation, rotation, and gripper control, and *the change in training benefit can differ across these subspaces*. In particular, after learning from related chunks, further full-chunk training can yield a smaller benefit in translation but a larger benefit in rotation (Figure 1). These observations reveal a further challenge beyond dynamic chunk sampling: even a dynamic chunk sampler cannot accommodate such differences through sampling frequency alone, because revisiting a chunk more or less often changes the training exposure of all its action subspaces together.

These observations motivate two coupled decisions: *which chunks to revisit, and how to distribute supervision within each chunk*. We introduce HERA (Hierarchical Response-Aware Supervision Allocation), a framework that jointly adapts training frequency and within-chunk supervision to the policy's evolving learning needs. A shared response estimator combines local physical trajectory correspondences with the current learning state to predict how candidate supervision would change a fixed training-side reference objective. Physical correspondences enable sparse calibration responses to inform estimates for related chunks. *Dynamic chunk sampling* uses these predictions to adapt training frequencies while keeping all available chunks eligible for sampling. These frequency changes scale the exposure of a chunk's subspace losses together. *Dynamic action-subspace weighting* further adapts their relative contributions through normalized, chunk-specific loss weights defined over the policy's native action subspaces. The two decisions are jointly optimized, allowing sampling probabilities to reflect the learning value of each chunk's reweighted supervision. Periodic refresh tracks the evolving policy state, while a bounded, rotating candidate pool controls the cost of reassessment.

Our contributions are summarized as follows:

- We characterize local, state-dependent learning dependencies across action chunks and heterogeneous changes in the benefits of further training across action subspaces, revealing two coupled granularities for supervision allocation.

- We introduce HERA, which combines physical trajectory correspondences and the current learning state in a shared response estimator to jointly optimize dynamic chunk sampling and chunk-specific action-subspace weighting under a common reference objective.
- We evaluate HERA with $\pi_{0.5}$ on LIBERO and PiPER-X, demonstrating higher success rates at matched optimizer-update budgets. On LIBERO, HERA reaches 80% success with 53.6% fewer optimizer updates and a $1.93\times$ wall-clock fine-tuning speedup over LoRA, including response-estimation and allocation overhead.

2 Related Work

Efficient Fine-Tuning of Vision-Language-Action Models. VLA models such as OpenVLA, $\pi_{0.5}$, and GR00T transfer large-scale pretraining to robotic control (Kim et al., 2025; Physical Intelligence et al., 2025; NVIDIA et al., 2025). Adapting these models to specific environments and tasks has motivated parameter-efficient fine-tuning and multitask adaptation. LoRA learns low-rank weight updates while freezing pretrained parameters (Hu et al., 2022), and LoRA-SP adjusts adaptation capacity across inputs and layers to accommodate the varying demands of robotic transfer (Kim et al., 2026). MetaVLA introduces context-aware meta co-training, combining target tasks with auxiliary task information through a shared model and contextual representations (Li et al., 2026). These approaches improve adaptation through parameter capacity allocation and task-level information sharing. A complementary question concerns how training effort should be distributed over the available supervision. Even with an effective parameterization, action chunks can differ in their contribution to further policy improvement, and those contributions can change during fine-tuning. Accelerating adaptation therefore also requires identifying which supervision remains useful at the current learning state and allocating training time accordingly.

Data Selection and Adaptive Supervision. Robot data curation uses complementary signals to identify useful training examples. STRAP retrieves relevant sub-trajectories through visual representations and temporal alignment (Memmel et al., 2025); SCIZOR filters suboptimal and redundant state-action pairs (Zhang et al., 2026); and SIEVE selects demonstrations to expose reusable behavioral structure (Wu et al., 2026). DataMIL estimates training utility with datamodels, while CUPID relates demonstration influence to policy returns (Dass et al., 2026; Agia et al., 2025). SIEVE and DataMIL construct subsets that remain fixed during subsequent policy training, leaving their membership unchanged as the policy's learning needs evolve. Utility estimation also introduces overhead through attribution training, evaluation rollouts, or influence computation, making its cost relevant to adaptation speed. Dynamic methods revisit data allocation during optimization: GLISTER iteratively selects subsets toward a validation objective (Killamsetty et al., 2021), while InfoBatch prunes samples using losses and rescales retained gradients (Qin et al., 2024). When the sampling unit is an action chunk, sample-level allocation changes the training exposure of its action subspaces together. Adaptive loss balancing provides finer control over supervision: GradNorm adjusts task weights to balance learning rates (Chen et al., 2018), and learned pose-regression losses balance translation and rotation objectives (Kendall & Cipolla, 2017). However, these shared task-level weights do not represent the potentially different weighting needs of individual action chunks. Together, these lines of work leave open how to coordinate dynamic chunk sampling with weights that vary across both chunks and action subspaces. The challenge is to assess their coupled effects on policy improvement as learning progresses, while keeping the cost of assessment low enough to yield a net reduction in adaptation time. In this paper, we study joint supervision allocation across action chunks and their action subspaces, using physical trajectory correspondences and the current learning state to assess candidate supervision under a common training-side reference objective.

3 Method

HERA turns the learning dependencies identified above into two coupled allocation decisions. *Dynamic chunk sampling* directs training opportunities toward useful chunks; *dynamic action-subspace weighting* adjusts the supervision supplied by each selected chunk. A shared, trajectory-guided response estimator evaluates both decisions against a fixed training-side reference objective. Figure 2 shows how these allocations enter a single shared-policy update.

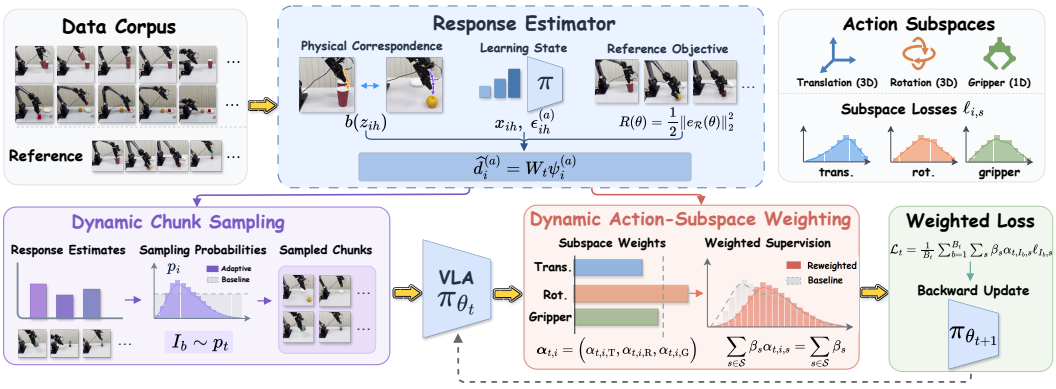

Figure 2: **Overview of HERA.** A shared response estimator guides dynamic chunk sampling and chunk-specific action-subspace weighting under one reference objective. Refreshes evaluate a rotating candidate pool and aggregate background responses; a baseline sampling branch maintains full-data access. Cached allocations drive subsequent policy updates. The illustrated Cartesian partition is replaced by native joint groups for joint-command policies.

3.1 Preliminaries

Let $\mathcal{D} = \{(o_i, A_i)\}_{i=1}^{N}$ contain observations, robot states, instructions, and encoded action chunks $A_i \in \mathbb{R}^{H\times d}$ after the base trainer's preprocessing. An *action subspace* is a coordinate subspace supported on a predefined, physically meaningful block of the native action representation. The physical groups $\mathcal{S}$ comprise translation, rotation, and gripper control for Cartesian commands, or joints and gripper control for joint commands. Their coordinate blocks remain fixed during training. Additional supervised coordinates form auxiliary groups $\mathcal{U}$; $\mathcal{G} = \mathcal{S} \cup \mathcal{U}$ indexes all supervised groups, with disjoint coordinate blocks $\mathcal{C}_s$.

We retain the base flow-matching supervision (Lipman et al., 2023). For a draw ξ containing flow time τ, noise Z, and training transformations, let $X_i^\tau = (1-\tau)A_i + \tau Z$ and $V_i^\star = Z - A_i$. The predicted velocity $v_\theta(X_i^\tau, \tau, o_i)$ gives

$$\ell_{i,s}(\theta;\xi) = \frac{1}{H|\mathcal{C}_s|}\sum_{h=1}^{H}\|v_{\theta,ihs} - V^\star_{ihs}\|_2^2, \qquad \ell_i^0 = \sum_{s\in\mathcal{G}}\beta_s\ell_{i,s}. \tag{1}$$

The fixed positive coefficients β_s sum to one and recover the base reduction; an unmasked coordinate mean uses $\beta_s = |\mathcal{C}_s|/d$. Here θ denotes the trainable parameters. Native preprocessing, masks, and global loss scaling follow Appendix A.3.

3.2 Problem Formulation

Our goal is to minimize total fine-tuning time to a given task success rate. Let u be the full-support baseline distribution over $\mathcal{D}$. A fixed training-side reference defines $R(\theta) = \frac{1}{2}\|e_{\mathcal{R}}(\theta)\|_2^2$. Reference chunks cover task–trajectory-progress strata with weights preserving their baseline probability masses. The instances, flow draws, and readout weights remain fixed, while their residuals evolve with the policy. This common target guides allocation; independent rollouts measure policy success.

HERA uses a sampling distribution p_t and subspace multipliers $\alpha_{t,i,s}$ determined before drawing each batch. For effective batch size B_t,

$$I_b \overset{\text{iid}}{\sim} p_t, \qquad \mathcal{L}_t = \frac{1}{B_t}\sum_{b=1}^{B_t}\sum_{s\in\mathcal{G}}\beta_s\alpha_{t,I_b,s}\ell_{I_b,s}(\theta_t;\xi_b). \tag{2}$$

Each occurrence receives fresh flow randomness. The multipliers satisfy $\sum_s \beta_s\alpha_{t,i,s} = 1$, preserving each chunk's loss-coefficient mass, and auxiliary multipliers equal one. Allocations are detached during backpropagation. Total adaptation time includes their estimation and optimization costs.

3.3 Dynamic Chunk Sampling

Reassessing useful experience within a fixed budget. Observation 1 motivates updating training frequencies as related chunks are learned. Each refresh evaluates a rotating pool $\mathcal{V}_t$ of at most M distinct chunks, proposed through task–progress queues and limited carry-over from the previous pool. This bounds policy evaluation cost while spreading candidate access across demonstrations. All selected chunks are reread at one frozen model and optimizer state.

Suppressing the refresh index, let $\mathcal{O}$ be the pool complement and $U_{\mathcal{V}} = \sum_{i\in\mathcal{V}} u_i$. We combine baseline sampling with nonnegative candidate masses a_i:

$$p_i = (1-\gamma)u_i + \mathbf{1}\{i \in \mathcal{V}\}a_i, \qquad \sum_{i\in\mathcal{V}} a_i = \gamma, \qquad 0 \le \gamma \le \gamma_{\max} < 1. \tag{3}$$

A pool-conditional bound $a_i \le \kappa_{\text{pool}}\gamma u_i/U_{\mathcal{V}}$ limits concentration, with $\kappa_{\text{pool}} > 1$. These constraints define $\mathcal{P}_{\mathcal{V}}$ and preserve $p_i \ge (1-\gamma_{\max})u_i > 0$ throughout the dataset.

Choosing complementary corrections. Initially set $\alpha_{i,s} = 1$. The shared estimator predicts each complete chunk's reference correction $\widehat{d}_i^0(\xi)$ under a local optimizer-response model with intercept r_t (Section 3.5). Candidate moments $m_i = \widehat{\mathbb{E}}[\widehat{d}_i^0]$ and $\nu_i = \widehat{\mathbb{E}}[\|\widehat{d}_i^0\|^2]$ summarize its mean effect and variability. Independent background probes estimate $m_{\mathcal{O}}, \nu_{\mathcal{O}}$ under $u(\cdot \mid \mathcal{O})$. Writing $\pi_{\mathcal{O}} = (1-\gamma)(1-U_{\mathcal{V}})$, define $\mu(p) = \pi_{\mathcal{O}}m_{\mathcal{O}} + \sum_{i\in\mathcal{V}} p_i m_i$ and $\nu(p) = \pi_{\mathcal{O}}\nu_{\mathcal{O}} + \sum_{i\in\mathcal{V}} p_i\nu_i$. We solve

$$\begin{aligned} &\min_{(\gamma,a)\in\mathcal{P}_{\mathcal{V}}} \quad \mathcal{F}_{t,B_t}(\mu(p),\nu(p)) + \lambda_{\text{KL}} D_{\text{KL}}(p\|u), \\ &\mathcal{F}_{t,B}(\mu,\nu) = \tfrac{1}{2}\|r_t - \mu\|^2 + \tfrac{1}{2B}(\nu - \|\mu\|^2). \end{aligned} \tag{4}$$

Here $\lambda_{\text{KL}} > 0$ controls departure from baseline sampling. The two terms in $\mathcal{F}$ capture residual error after the mean batch correction and variation across sampled updates. Their combined effect accounts for complementary and overlapping responses, including the background contribution. This is the expected squared error of the local moment model for any $B \ge 1$.

From training frequency to supervision composition. A chunk's expected subspace coefficients are $p_i\beta_s$, so sampling preserves their relative ratios $\beta_s/\beta_{s'}$. Observation 2 identifies different benefit changes within those same chunks. Addressing these differences requires a second allocation level that adjusts the relative contributions of their subspace losses.

3.4 Dynamic Action-Subspace Weighting

We extend candidate mass to nonnegative coefficients $\Gamma_{i,s}$, with $a_i = \sum_s \Gamma_{i,s}$ and $\gamma = \sum_{i\in\mathcal{V}} a_i$. The resulting global supervision allocation is

$$Q_{i,s} = (1-\gamma)u_i\beta_s + \mathbf{1}\{i\in\mathcal{V}\}\Gamma_{i,s}, \qquad p_i = \sum_s Q_{i,s}, \qquad \alpha_{i,s} = \frac{Q_{i,s}}{p_i\beta_s}. \tag{5}$$

The row mass controls training frequency and its normalized entries control within-chunk supervision. For physical subspaces, impose $a_-\beta_s a_i \le \Gamma_{i,s} \le a_+\beta_s a_i$, where $0 < a_- < 1 < a_+$; auxiliary groups satisfy $\Gamma_{i,s} = \beta_s a_i$. Together with $\mathcal{P}_{\mathcal{V}}$, these define the feasible set $\mathcal{F}_{\mathcal{V}}$. They preserve normalized, bounded multipliers; chunks outside the pool have $\alpha_{i,s} = 1$.

Let column s of $\widehat{D}_i(\xi)$ predict the reference response to $g_{i,s} = \nabla_\theta \ell_{i,s}$. All columns share the same linear map, so $\widehat{d}_i^0 = \widehat{D}_i\beta$ at a fixed draw. Estimate $\widehat{\bar{D}}_i = \widehat{\mathbb{E}}[\widehat{D}_i]$ and $\widehat{M}_i = \widehat{\mathbb{E}}[\widehat{D}_i^\top \widehat{D}_i]$ on the pool. For row Q_i written as a column vector, the global moments become

$$\mu(Q) = \pi_{\mathcal{O}}m_{\mathcal{O}} + \sum_{i\in\mathcal{V}} \widehat{\bar{D}}_i Q_i, \qquad \nu(Q) = \pi_{\mathcal{O}}\nu_{\mathcal{O}} + \sum_{i\in\mathcal{V}} \frac{Q_i^\top \widehat{M}_i Q_i}{p_i}. \tag{6}$$

Joint moments retain interactions between subspace updates. We optimize

$$\min_{\Gamma\in\mathcal{F}_{\mathcal{V}}} \mathcal{F}_{t,B_t}(\mu(Q),\nu(Q)) + \lambda_{\text{KL}} D_{\text{KL}}(Q\|u\beta^\top). \tag{7}$$

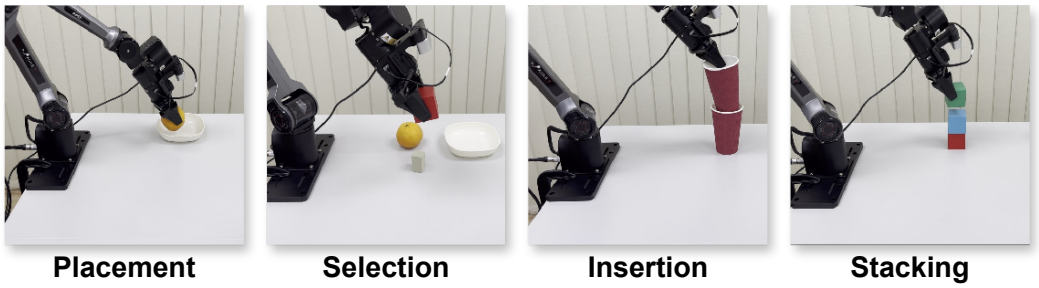

Figure 3: **Real-world manipulation on PiPER-X.** Placement, selection, insertion, and stacking.

Setting $\Gamma_{i,s} = a_i\beta_s$ recovers chunk-only allocation. Starting from this solution, joint optimization updates both row masses and subspace proportions, so sampling responds to each chunk's reweighted learning effect. The problem is convex for frozen moments and is solved with constrained Frank–Wolfe updates; the derivation, global KL expansion, and solver are given in Appendix A.6. The final marginal p draws complete chunks, and each selected ID uses its final α on either sampling branch for one shared-policy update.

3.5 Trajectory-Guided Training-Response Estimation

A common response model for both allocations. HERA estimates how supervision changes the entire reference residual, including cross-subspace effects. At a frozen refresh state, draw an independent baseline gradient g_0. Let U_t denote the actual functional optimizer update and $\bar{U}_t$ its stabilized virtual counterpart, anchored so that $\bar{U}_t(g_0) = U_t(g_0)$. The local response operator is $\widetilde{K}_t = -D[e_{\mathcal{R}}(\bar{U}_t(g))]_{g=g_0}$, with $r_t = e_{\mathcal{R}}(U_t(g_0)) + \widetilde{K}_t g_0$. A sparse set of source gradients provides labels $y_j = \widetilde{K}_t g_j$ through Jacobian–vector products on the full fixed reference. The derivative follows the stabilized virtual optimizer; policy training uses the original optimizer. Appendices A.5 and A.6 specify the anchored map and approximation analysis.

Sharing responses through physical correspondences. Local native-action descriptors z_{ih} assign trajectory positions to fixed physical prototypes through $b(z_{ih})$. Current conditioning x_{ih} includes flow time, chunk position, and projected action-token features. For a complete-loss or subspace-contrast source $\ell_i^{(a)}$, let $\epsilon_{ih}^{(a)} = \nabla_{v_{\theta,ih}} \ell_i^{(a)}$ be its signed output derivative. The shared predictor is

$$\psi_i^{(a)} = \sum_h [b(z_{ih}) \otimes x_{ih}] \otimes \epsilon_{ih}^{(a)}, \qquad \widehat{d}_i^{(a)} = W_t \psi_i^{(a)}. \tag{8}$$

Physical correspondences link related supervision, while current features and signed errors specify the corrections still available. We fit W_t by ridge regression on current probe pairs (ψ_j, y_j). Its linearity preserves combinations of subspace supervision within a chunk. With $C = |\mathcal{S}|$ physical subspaces, a complete-loss direction and $C - 1$ subspace contrasts span the permitted weighted updates (Appendix A.3).

A forward-only scan of the current pool produces candidate moments. Independent background batches supply aggregate complete-loss response moments for the remaining data. After warmup, reference readouts, probes, the response map, moments, and allocations are refreshed together every L optimizer updates. Cached p, α drive the intervening training steps, and numerical failures revert to baseline allocation. The budget M bounds each reassessment and the rotating pool extends candidate access over training. Appendix A.1 provides sampling, caching, and timing details. Inference uses the adapted policy.

Table 1: **Comparison on LIBERO and real-world manipulation.** Success rates (%) are reported at 20K and 50K optimizer updates.

Method	Optimizer Updates	LIBERO					Real-World				
		Spatial	Object	Goal	Long	**Avg.**	Placement	Selection	Insertion	Stacking	**Avg.**
LoRA	20K (40%)	79.4	88.6	80.6	55.6	76.05	58	52	43	25	44.50
+ SIEVE		**89.6**	90.2	72.5	61.2	78.38	**69**	57	48	33	51.75
+ DataMIL		77.3	89.7	80.2	65.8	78.25	67	62	41	31	50.25
+ InfoBatch		74.7	88.6	**87.1**	63.0	78.35	57	56	46	32	47.75
+ FAMO		60.2	88.0	81.0	54.9	71.03	62	51	37	29	44.75
+ HERA (ours)		89.5	**92.6**	83.4	**76.2**	**85.43**	68	**63**	**52**	**38**	**55.25**
LoRA	50K (100%)	95.6	96.2	88.4	84.7	91.23	72	67	59	42	60.00
+ SIEVE		**95.8**	97.4	85.1	83.5	90.45	73	70	62	45	62.50
+ DataMIL		93.3	**97.8**	86.4	85.0	90.63	71	68	59	44	60.50
+ InfoBatch		90.9	94.7	89.2	88.6	90.85	68	65	60	46	59.75
+ FAMO		85.4	90.3	88.5	82.8	86.75	63	64	58	54	59.75
+ HERA (ours)		95.5	95.9	**91.7**	**94.6**	**94.43**	**78**	**73**	**64**	**58**	**68.25**

All methods use the same LoRA configuration. One shared policy is trained jointly on the four real-world tasks. Averages are computed separately over the four LIBERO suites and four real-world tasks.

4 Experiments

4.1 Experiment Setup

Base model and training setup. We use $\pi_{0.5}$ (Physical Intelligence et al., 2025) as the flow-based VLA backbone and adapt it with LoRA (Hu et al., 2022). All methods start from the same pre-trained checkpoint and share the same trainable modules, optimizer, learning-rate schedule, action representation, and training budget. We use an effective batch size of 32, with action horizons of 10 on LIBERO and 16 on PiPER-X. All training and wall-clock measurements are conducted on a single NVIDIA RTX 5090. Unless otherwise specified, HERA evaluates $M = 1{,}024$ candidate chunks per refresh, performs 250 baseline warm-up updates, and refreshes its response estimates and supervision allocation every 1,000 optimizer updates. Full optimization, calibration, and runtime configurations are provided in the supplementary material.

Datasets and evaluation. We evaluate HERA in simulation and real-world manipulation. Our simulation evaluation uses LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long (Liu et al., 2023b). At each evaluated checkpoint and for each training seed, we evaluate 50 initial states per task, yielding 500 episodes per suite and 2,000 episodes across the four LIBERO suites. All reported LIBERO results are averaged over three independent training seeds. We report policy success at matched optimizer-update budgets together with the total fine-tuning time required to reach predefined success-rate targets. All timing results include method-specific data-selection, response-estimation, and allocation overhead.

Real-world experiments. We evaluate one shared instruction-conditioned policy on four PiPER-X tasks shown in Figure 3: orange placement (**Placement**), block selection and placement (**Selection**), cup insertion (**Insertion**), and block stacking (**Stacking**). The robot observes each task through a wrist-mounted camera and a third-person camera, together with proprioceptive state and the task instruction. We collect 200 demonstrations, with 50 demonstrations per task and 83,625 recorded frames in total. The real-world policy is trained with a single random seed. For each reported evaluation, every task is executed for 100 physical trials; its success rate is the fraction of successful trials, and the reported real-world average is the unweighted mean over the four task success rates. Following the native control interfaces, LIBERO uses translation, rotation, and gripper action subspaces, while PiPER-X uses six joint-coordinate subspaces and one gripper subspace.

Baselines. LoRA with baseline data sampling and fixed loss weights serves as our fine-tuning reference. We compare HERA with four representative supervision-allocation approaches under the same $\pi_{0.5}$ LoRA configuration. SIEVE (Wu et al., 2026) selects demonstrations using reusable behavioral structure, and DataMIL (Dass et al., 2026) estimates training utility for data selection.

Table 2: **Time to target success on LIBERO.**

Method	70% SR		80% SR		90% SR	
	Steps	Time OH	Steps	Time OH	Steps	Time OH
LoRA	13K	1.00 0.0%	28K	1.00 0.0%	46K	1.00 0.0%
+ SIEVE	12K	0.78 4.6%	26K	0.81 2.2%	50K	0.95 1.1%
+ DataMIL	13K	1.13 29.9%	22K	0.82 20.1%	41K	0.86 11.9%
+ InfoBatch	12K	**0.73** $< 0.1\%$	26K	0.77 $< 0.1\%$	43K	0.79 $< 0.1\%$
+ FAMO	18K	1.51 26.8%	31K	1.26 26.8%	NR	—
+ HERA	**9K**	0.76 6.3%	**13K**	**0.52** 6.5%	**22K**	**0.53** 6.7%

Time is normalized to LoRA and includes method-specific overhead. OH denotes the overhead fraction; NR denotes target not reached.

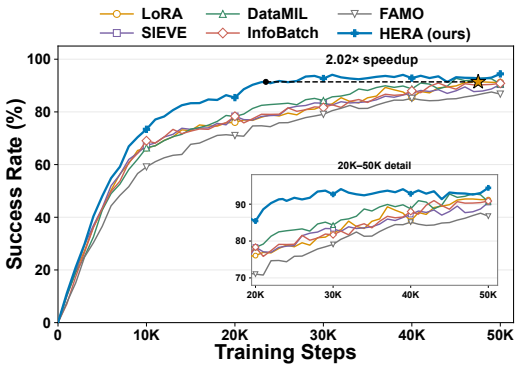

Figure 4: **Learning curves on LIBERO.** Curves show success rates averaged over three independent training seeds. The $2.02\times$ annotation denotes the optimizer-update speedup for HERA to first match LoRA's peak mean success rate within the 50K-update budget.

InfoBatch (Qin et al., 2024) dynamically adjusts sample usage, while FAMO (Liu et al., 2023a) performs adaptive loss balancing; for FAMO, we use the native action-subspace losses as its balancing objectives. Baseline adaptations and complete implementation details are provided in the supplementary material.

4.2 Main Results

Policy quality at matched update budgets. Table 1 evaluates how effectively each method uses its optimizer updates. HERA achieves the highest average success rate in both simulation and real-world manipulation at 20K and 50K updates. At 20K updates, HERA improves the average success rate over LoRA by 9.38 percentage points on LIBERO and 10.75 points on PiPER-X. The advantage persists at 50K, with particularly strong results on LIBERO-Long and real-world Stacking. The consistent gains across Cartesian-command simulation and joint-command real-world control demonstrate the generality of HERA's supervision-allocation mechanism across different native action representations.

Learning dynamics. Figure 4 shows how the advantage of HERA develops throughout fine-tuning. HERA improves rapidly in the early and middle stages and sustains stronger learning progress as training continues. To quantify this gain at matched policy quality, we use the three-seed mean learning curves: we take the highest mean success rate reached by LoRA within the 50K-update budget and identify the first HERA checkpoint whose mean success rate reaches the same level. The ratio of the corresponding optimizer-update counts gives a $2.02\times$ update speedup, as annotated in Figure 4.

The learning trajectories also reveal clear differences among supervision allocation strategies. SIEVE and DataMIL provide strong early improvements, while their relative gains diminish later as their selected subsets remain fixed throughout policy optimization. InfoBatch continually adapts sample usage, but changes the exposure of all action subspaces within an action chunk together. FAMO adapts loss weights across action subspaces, while chunk visitation continues to follow the underlying sampling distribution. HERA coordinates both decisions and periodically updates them with the current learning state, yielding sustained improvement across the fine-tuning trajectory.

Time to target success. Table 2 complements the learning curves with fixed success-rate targets and end-to-end fine-tuning time. At 80% success, HERA reaches the target in 13K updates, compared with 28K for LoRA, reducing the required optimizer updates by 53.6%. Including response estimation, candidate evaluation, and allocation overhead, HERA requires 0.52 times the LoRA wall-clock time, corresponding to a $1.93\times$ fine-tuning speedup. HERA retains the strongest time-to-target result at 90% success, while InfoBatch is marginally faster at the lower 70% target due

Table 3: **Component ablation on LIBERO with $\pi_{0.5}$.** All variants are trained for the same 50K-update schedule. We report average LIBERO success rate at matched update budgets, together with the optimizer updates and normalized end-to-end fine-tuning time required to first reach 80% success. Overhead denotes the fraction of total adaptation time spent on method-specific additional computation up to the 80% success target.

Variant	SR@20K ↑	SR@50K ↑	Updates@80% ↓	Time@80% ↓	Overhead ↓
LoRA	76.05	91.23	28K	1.00	0.0%
Sampling only	82.4	92.8	17K	0.63	4.3%
Sequential allocation	83.8	93.6	15K	0.57	6.5%
Static allocation	80.9	91.9	20K	0.72	0.7%
w/o Physical Correspondence	81.8	92.7	18K	0.69	6.7%
HERA	**85.43**	**94.43**	**13K**	**0.52**	**6.5%**

to its negligible additional computation. These results show that HERA's higher update efficiency translates into substantial end-to-end acceleration as the target policy quality increases.

4.3 Ablation Study

Table 3 examines HERA's allocation and estimation choices. Dynamic chunk sampling reduces the updates required to reach 80% success from LoRA's 28K to 17K. Extending this allocation to action subspaces and jointly optimizing both levels reduces the requirement to 13K, with higher success rates at both matched update budgets. Sampling controls how often each chunk is revisited, and subspace weighting controls its internal supervision contributions. Sequential allocation reaches the target in 15K updates and 0.57 normalized time, compared with HERA's 13K and 0.52. Joint optimization thus aligns sampling probabilities with each chunk's reweighted learning effect.

Sustaining these gains requires periodically reassessing the allocation. Static allocation freezes the first post-warmup decision and reaches 80% success in 20K updates, with 0.72 normalized time and 0.7% overhead. HERA's periodic refresh reduces the normalized time to 0.52 while incurring 6.5% overhead. The net time saving demonstrates that reassessment delivers sufficient learning progress to offset its recurring computational cost.

Physical trajectory structure strengthens the shared response estimator. Removing physical-correspondence features increases the required updates from 13K to 18K and normalized time from 0.52 to 0.69; success rates also decrease at both update budgets. These results support physical correspondences as an effective basis for sharing calibration information across related chunks. Together, the ablations establish the contributions of hierarchical allocation, joint optimization, periodic reassessment, and trajectory-guided response estimation.

5 Conclusion

We presented HERA, a framework for accelerating flow-based VLA fine-tuning through hierarchical, response-aware supervision allocation. HERA uses physical trajectory correspondences and the current learning state to jointly determine which action chunks receive training opportunities and how supervision is distributed across their native action subspaces. Periodic reassessment maintains this coordination throughout adaptation. Experiments on LIBERO and PiPER-X demonstrate higher policy quality at matched update budgets across Cartesian-command and joint-command interfaces. On LIBERO, HERA reaches 80% success with 53.6% fewer optimizer updates and a $1.93\times$ wall-clock speedup over LoRA, including response-estimation and allocation overhead. Together, these findings identify learning-dependent supervision allocation as an effective route to more productive updates and faster VLA adaptation.

AI Use Statement

We used generative AI tools to assist with language polishing, literature retrieval and discovery, and selected mathematical derivations. Specifically, AI tools were used to improve grammar, phrasing,

and readability; to help identify potentially relevant prior work; and to assist with intermediate steps in a limited subset of mathematical derivations. All retrieved references were manually inspected and verified by the authors. The mathematical formulation and reasoning remained author-led, and all AI-assisted derivations were independently checked by the authors for correctness. We did not use generative AI tools for research ideation or execution, drafting substantive sections of the paper, or generating synthetic datasets. The authors take full responsibility for the final content of this work.

Reproducibility Statement

We have taken several steps to ensure the reproducibility of our results. Sec. 4.1 describes the experimental settings and evaluation protocols. Appendix A provides the complete HERA procedure, including pseudocode and detailed formulations of response estimation and supervision allocation. Appendix B specifies the benchmarks, compared methods, evaluation metrics, and timing protocol, while Appendix C reports the training configurations, random seeds, implementation details, hyperparameters, and runtime accounting. Appendix D provides additional ablations and response-estimation diagnostics. Together, these materials are intended to facilitate transparent replication of our findings.

Ethics Statement

This work studies efficient fine-tuning of vision-language-action policies for robotic manipulation. The study does not involve human subjects or sensitive personal information, and does not involve decision-making about individuals or sensitive attributes. As with learning-based robotic systems more broadly, deployment outside the evaluated settings may introduce safety and reliability risks due to unintended robot behavior or distribution shifts. We therefore regard our real-world results as research-stage evaluations rather than evidence of readiness for safety-critical deployment, and recommend appropriate physical safeguards and task-specific validation for real-world use. We have followed the ICLR Code of Ethics throughout this work.

References

Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. CUPID: Curating data your robot loves with influence functions. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2907–2932. PMLR, 2025.

Daniel A. Braun, Ad Aertsen, Daniel M. Wolpert, and Carsten Mehring. Motor task variation induces structural learning. *Current Biology*, 19(4):352–357, 2009. doi: 10.1016/j.cub.2009.01.036.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 794–803. PMLR, 2018.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. DataMIL: Selecting data for robot imitation learning with datamodels. In *International Conference on Learning Representations*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5974–5983, 2017.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. GLISTER: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8110–8118, 2021. doi: 10.1609/aaai.v35i9.16988.

Donghoon Kim, Minji Bae, Unghui Nam, Gyeonghun Kim, Suyun Lee, Kyuhong Shim, and Byonghyo Shim. Adaptive capacity allocation for vision language action fine-tuning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

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. PMLR, 2025.

Chen Li, Zhantao Yang, Han Zhang, Fangyi Chen, Chenchen Zhu, Anudeepsekhar Bolimera, and Marios Savvides. MetaVLA: Unified meta co-training for efficient embodied adaptation. In *International Conference on Learning Representations*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. FAMO: Fast adaptive multitask optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023a.

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*, volume 36, 2023b.

Marius Memmel, Jacob Berg, Bingqing Chen, Abhishek Gupta, and Jonathan Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. In *International Conference on Learning Representations*, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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. arXiv preprint arXiv:2504.16054, 2025.

Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, Zhaopan Xu, Daquan Zhou, Lei Shang, Baigui Sun, Xuansong Xie, and Yang You. InfoBatch: Lossless training speed up by unbiased dynamic data pruning. In *International Conference on Learning Representations*, 2024.

Changti Wu, Bin Yu, Zhaolong Shen, Shijie Lian, Xiaopeng Lin, Cong Huang, Zhirui Zhang, Lei Zhang, and Kai Chen. SIEVE: Structure-aware data selection for imitation learning with VLA models. arXiv preprint arXiv:2607.06442, 2026.

Yu Zhang, Yuqi Xie, Huihan Liu, Rutav Shah, Michael Wan, Linxi Fan, and Yuke Zhu. SCIZOR: A self-supervised approach to data curation for large-scale imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. 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.

A METHOD DETAILS

This section completes the connection between HERA's response model and its two supervision-allocation decisions. We first describe the training loop, then specify the common reference, the budgeted candidate family, and the response statistics used by joint allocation.

A.1 HERA FINE-TUNING ALGORITHM

HERA alternates between reassessing supervision and training with the resulting allocation. A refresh evaluates the current policy at one frozen model and optimizer state; its sampling probabilities and subspace weights then guide a block of ordinary updates. Algorithm 1 summarizes this procedure.

Algorithm 1 HERA Fine-Tuning

Require: Initial policy and optimizer state (θ_0, o_0); training data $\mathcal{D}$ and baseline distribution u; subspace groups $\mathcal{G}$ and coefficients β; fixed reference $\mathcal{R}$; candidate budget M; warmup W, refresh interval L, batch size B, update budget T

Ensure: Adapted policy parameters θ_T

Initialize $p \leftarrow u$, $\alpha_{i,s} \leftarrow 1$, and candidate queues
Prepare fixed reference draws, physical prototypes, and probe IDs
for $t = 0, \dots, T-1$ **do**
  if $t \geq W$ and $(t - W) \bmod L = 0$ **then**
    Freeze (θ_t, o_t) during the refresh
    Propose $\mathcal{V}$ with $|\mathcal{V}| \leq M$; set $\mathcal{O} \leftarrow \mathcal{D} \setminus \mathcal{V}$
    Recompute reference readouts and an independent anchor gradient g_0
    Construct the anchored response operator $\widetilde{K}_t$ and intercept r_t
    Measure source-direction responses and fit the shared estimator W_t
    Scan $\mathcal{V}$ to estimate candidate response moments
    Estimate background moments from independent batches in $\mathcal{O}$
    Solve chunk-only allocation, then initialize and solve joint allocation
    Recover (p, α) from the final joint coefficients Γ
    if the response or allocation checks fail **then**
      Set $p \leftarrow u$ and $\alpha_{i,s} \leftarrow 1$
    end if
    Cache (p, α); retain the original (θ_t, o_t)
  end if
  Draw $I_1, \dots, I_B \overset{\text{iid}}{\sim} p$ and fresh flow draws $\xi_1, \dots, \xi_B$
  $\mathcal{L}_t \leftarrow \dfrac{1}{B} \displaystyle\sum_{b=1}^{B} \sum_{s \in \mathcal{G}} \beta_s \alpha_{I_b,s} \ell_{I_b,s}(\theta_t; \xi_b)$
  $(\theta_{t+1}, o_{t+1}) \leftarrow \mathrm{OptimizerStep}(\theta_t, o_t, \nabla_\theta \mathcal{L}_t)$
end for
return θ_T

The first refresh follows W completed updates and supplies update $W + 1$. Reference evaluation, calibration, and virtual updates leave the policy parameters and optimizer moments unchanged. The cached allocation is held fixed during subsequent policy updates, with independent flow randomness for every sampled occurrence. The final multiplier associated with a chunk is applied regardless of the sampling branch that selected it. Numerically invalid responses or infeasible allocations use the baseline allocation for the next interval. Inference uses the adapted policy directly.

A.2 FIXED TRAINING-SIDE REFERENCE

The reference provides a common target against which both allocation levels are assessed. Training chunks are stratified by task and trajectory progress. For a stratum q with baseline mass $\omega_q = \sum_{i \in q} u_i$, reference slots are sampled according to u_i/ω_q and share total weight ω_q. Reference chunks remain eligible for ordinary training.

Let ω_r be the weight of reference slot r, S_R its number of fixed flow draws, and η_s the fixed physical-subspace readout weights, with $\sum_r \omega_r = 1$ and $\sum_{s\in\mathcal{S}} \eta_s = 1$. For flow-prediction residual $E_{r,k,h,j}(\theta) = v_{\theta,r,k,h,j} - V^{\star}_{r,k,h,j}$, define

$$e_{\mathcal{R}}(\theta) = \mathrm{concat}_{r,k,h,\, s\in\mathcal{S},\, j\in\mathcal{C}_s} \left[\sqrt{\frac{2\omega_r \eta_s}{S_R H |\mathcal{C}_s|}} E_{r,k,h,j}(\theta) \right], \qquad R(\theta) = \tfrac{1}{2}\|e_{\mathcal{R}}(\theta)\|^2. \tag{9}$$

The reference instances, weights, and flow draws remain fixed throughout training. Each refresh recomputes their residuals under the current policy. Thus the target remains common across refreshes while its learning signal evolves. Reference weights η_s are distinct from the training coefficients β_s and adaptive multipliers $\alpha_{i,s}$.

A.3 Action-Subspace Formulation

HERA partitions the policy's native supervised coordinates into fixed physical subspaces and any auxiliary groups. Subspace losses retain the base preprocessing, loss masks, and reduction convention:

$$\ell_i^0 = \sum_{s\in\mathcal{G}} \beta_s \ell_{i,s}, \qquad \sum_{s\in\mathcal{G}} \beta_s = 1, \qquad \sum_{s\in\mathcal{G}} \beta_s \alpha_{i,s} = 1. \tag{10}$$

Physical multipliers vary across chunks, and auxiliary multipliers remain one. Unit multipliers recover the baseline loss and its gradient. Section C.4 gives the Cartesian and joint-command partitions used in the experiments.

Compact source directions. Normalization allows the weighted gradient to be represented using only $C = |\mathcal{S}|$ source directions. Choose an algebraic reference subspace $s_\star$, and write $g_{i,s} = \nabla_\theta \ell_{i,s}$, $g_i^0 = \nabla_\theta \ell_i^0$, and $\zeta_{i,s} = \beta_s(\alpha_{i,s} - 1)$. Then

$$g_i^{\mathrm{weighted}} = g_i^0 + \sum_{s\in\mathcal{S}\setminus\{s_\star\}} \zeta_{i,s}(g_{i,s} - g_{i,s_\star}). \tag{11}$$

A complete-loss direction retains the baseline supervision, including auxiliary coordinates; $C - 1$ contrasts redistribute physical supervision. The coefficient of $s_\star$ remains adaptive through normalization. All directions for a source chunk share the same flow draw.

A.4 Budgeted Candidate Reassessment

Each refresh evaluates at most M distinct chunks. Persistent shuffled task–progress queues provide at least three quarters of the pool, and up to one quarter is retained from the previous pool. Queue positions advance across refreshes, and duplicate IDs are excluded within a pool. Every selected chunk is reevaluated at the current frozen state. This combines continued access to new candidates with reuse of previously considered supervision.

Let $\mathcal{V}$ denote the current pool, $\mathcal{O}$ its complement, and $U_{\mathcal{V}} = \sum_{i\in\mathcal{V}} u_i$. Adaptive masses satisfy

$$p_i = (1-\gamma)u_i + \mathbf{1}\{i \in \mathcal{V}\}a_i, \qquad \sum_{i\in\mathcal{V}} a_i = \gamma, \qquad 0 \le \gamma \le \gamma_{\max} < 1, \qquad 0 \le a_i \le \kappa_{\mathrm{pool}}\gamma \frac{u_i}{U_{\mathcal{V}}}. \tag{12}$$

A training slot draws from u with probability $1-\gamma$ and from a_i/γ with probability γ when $\gamma > 0$. This gives every chunk probability at least $(1-\gamma_{\max})u_i$. Chunks outside the pool retain baseline subspace proportions and contribute through aggregate background responses to the allocation objective.

A.5 Trajectory-Guided Training-Response Estimation

Anchored optimizer response. An independent baseline batch provides anchor gradient g_0 at the frozen training state. Let $U_t(g)$ be the functional policy update and $U_t^\delta(g)$ its stabilized virtual

version, retaining the optimizer state and update order with a positive regularizer inside the adaptive second-moment square root. We use

$$\begin{aligned} \bar{U}_t(g) &= U_t^{\delta}(g) + \mathrm{sg}[U_t(g_0) - U_t^{\delta}(g_0)], \\ \widetilde{K}_t &= -D[e_{\mathcal{R}}(\bar{U}_t(g))]_{g=g_0}, \qquad r_t = e_{\mathcal{R}}(U_t(g_0)) + \widetilde{K}_t g_0. \end{aligned} \tag{13}$$

Here sg stops differentiation. The virtual update matches the actual update at the anchor, and its stabilized derivative defines the local response model $r_t - \widetilde{K}_t g$. Calibration labels $y_j = \widetilde{K}_t g_j$ are computed by Jacobian–vector products over the complete fixed reference, with detached source gradients. Ordinary training uses the original optimizer.

Physical response sharing. Fixed prototype assignments $b(z_{ih})$ relate local native-action patterns. Current features x_{ih} contain flow time, chunk position, and projected action-token features. For a complete-loss or contrast source $\ell_i^{(a)}$, its signed output derivative $\epsilon_{ih}^{(a)} = \nabla_{v_{\theta,ih}} \ell_i^{(a)}$ gives

$$\psi_i^{(a)} = \sum_h [b(z_{ih}) \otimes x_{ih}] \otimes \epsilon_{ih}^{(a)}, \qquad \widehat{d}_i^{(a)} = W_t \psi_i^{(a)}. \tag{14}$$

Physical assignments connect related supervision, while current features and signed errors encode its remaining correction needs. For P fixed probe chunks, the C directions yield $n = PC$ calibration pairs. Writing their columns as Ψ and Y, the shared map is fitted by ridge regression:

$$W_t = \arg\min_W \|Y - W\Psi\|_F^2 + \lambda_{\mathrm{fit}} \|W\|_F^2, \qquad \lambda_{\mathrm{fit}} > 0. \tag{15}$$

The small dual system is solved by Cholesky factorization. The common linear map preserves combinations of source directions. A forward-only scan with independent candidate draws produces response means and joint second moments, including cross-subspace interactions.

Background responses. The baseline sampling branch gives the pool complement a nonzero role in each training batch. To represent this contribution, draw J independent batches of b_{bg} chunks from $u(\cdot \mid \mathcal{O})$. Applying $\widetilde{K}_t$ to each complete-loss mean gradient gives y_j^{bg}. The individual-chunk response moments are estimated by

$$\begin{aligned} \bar{y} &= \frac{1}{J} \sum_j y_j^{\mathrm{bg}}, \qquad S_y^2 = \frac{1}{J-1} \sum_j \|y_j^{\mathrm{bg}} - \bar{y}\|^2, \\ \widehat{m}_{\mathcal{O}} &= \bar{y}, \qquad \widehat{\nu}_{\mathcal{O}} = \|\bar{y}\|^2 + \left(b_{\mathrm{bg}} - \frac{1}{J}\right) S_y^2. \end{aligned} \tag{16}$$

The correction accounts for averaging within each background batch: for individual-response variance trace σ^2, $\mathbb{E}S_y^2 = \sigma^2/b_{\mathrm{bg}}$ and $\mathbb{E}\|\bar{y}\|^2 = \|m_{\mathcal{O}}\|^2 + \sigma^2/(Jb_{\mathrm{bg}})$. These moments therefore match the single-slot statistics used by the allocation objective. Background gradients and their reference responses are evaluated separately from the calibration batches at the same snapshot.

Compact response space. Ridge predictions lie in the calibration-label span. An orthonormal basis V spanning these labels and the background responses supports allocation in at most $PC + J$ dimensions. For any represented mean response μ,

$$\|r_t - \mu\|^2 = \|V^\top r_t - V^\top \mu\|^2 + \|(I - VV^\top) r_t\|^2. \tag{17}$$

The final term is constant with respect to allocation, so projection onto the complete span preserves the represented objective.

A.6 Joint Supervision Allocation

The second allocation level extends candidate masses into subspace coefficients $\Gamma_{i,s}$, with $a_i = \sum_s \Gamma_{i,s}$:

$$Q_{i,s} = (1-\gamma) u_i \beta_s + \mathbf{1}\{i \in \mathcal{V}\} \Gamma_{i,s}, \qquad p_i = \sum_s Q_{i,s}, \qquad \alpha_{i,s} = \frac{Q_{i,s}}{p_i \beta_s}. \tag{18}$$

Physical coefficients satisfy $a_- \beta_s a_i \le \Gamma_{i,s} \le a_+ \beta_s a_i$; auxiliary coefficients equal $\beta_s a_i$. These constraints preserve bounded, normalized multipliers. Fixing $\Gamma_{i,s} = a_i \beta_s$ recovers chunk-only sampling. Joint optimization then adjusts both each row's training frequency and its internal supervision composition.

A shared batch-level objective. Let Z denote a modeled slot response, using $\widehat{D}_I(\xi)(\beta \odot \alpha_I)$ for a candidate chunk and the measured complete-loss moments for the complement. Write $\mu = \mathbb{E}[Z]$ and $\nu = \mathbb{E}\|Z\|^2$. For independent training slots,

$$\mathbb{E}\left[\frac{1}{2}\left\|r_t - \frac{1}{B}\sum_{b=1}^{B} Z_b\right\|^2\right] = \frac{1}{2}\|r_t - \mu\|^2 + \frac{1}{2B}(\nu - \|\mu\|^2). \tag{19}$$

For the mixed population used in allocation, the background is represented by its measured moments. With $\pi_{\mathcal{O}} = (1-\gamma)(1-U_{\mathcal{V}})$, the global moments are $\mu = \pi_{\mathcal{O}} m_{\mathcal{O}} + \sum_{i\in\mathcal{V}} \widehat{\widehat{D}}_i Q_i$ and $\nu = \pi_{\mathcal{O}}\nu_{\mathcal{O}} + \sum_{i\in\mathcal{V}} Q_i^\top \widehat{M}_i Q_i / p_i$. Thus both the candidate and baseline branches contribute to the same expected post-update reference error. The full-data regularizer is

$$D_{\mathrm{KL}}(Q\|u\beta^\top) = \sum_{i\in\mathcal{V},s} Q_{i,s}\log\frac{Q_{i,s}}{u_i\beta_s} + \pi_{\mathcal{O}}\log(1-\gamma). \tag{20}$$

The complement is aggregated exactly in this term, allowing full-data regularization with explicit storage only for the candidate rows. The expected training gradient is $\sum_{i,s} Q_{i,s}\mathbb{E}_\xi[g_{i,s}]$.

Optimization and local-model scope. For frozen response moments, μ is affine in Γ. Expanding Eq. equation 19 gives a nonnegative quadratic coefficient $(1-1/B)/2$, and each candidate second-moment term is a positive-semidefinite quadratic perspective. Together with the KL regularizer and linear constraints, these terms define a convex problem for $B \geq 1$. A constrained Frank–Wolfe solver starts from chunk-only allocation and updates the joint coefficients within the feasible set. The compact directions in Eq. equation 11 give the same objective because their coefficients satisfy $p_i\zeta_{i,s} = \Gamma_{i,s} - a_i\beta_s$.

This characterization applies to the frozen local response model. Under a uniform proxy-error bound Δ over the feasible family, an ϵ_{opt}-accurate proxy solution is within $2\Delta + \epsilon_{\mathrm{opt}}$ of the corresponding actual-update regularized optimum. Response fitting, finite-sample moments, the virtual optimizer, and snapshot age determine that approximation error. Periodic refresh updates the model as training progresses; rollout success and time-to-success evaluate the resulting policy improvements.

B Experimental Details

B.1 Benchmarks and Evaluation Scope

We evaluate adaptation on the Spatial, Object, Goal, and Long suites of LIBERO (Liu et al., 2023b). Each suite contains ten tasks, covering spatial relations, manipulated objects, task goals, and long-horizon behaviors, respectively. Each checkpoint uses 50 initial states per task and training seed, yielding 500 episodes per suite and 2,000 across all four suites. These states are provided by the benchmark. The initial states are shared across methods and training runs. LIBERO results and learning curves average three independently trained policies. The study evaluates adaptation to the task collection used for fine-tuning, with Cartesian-command simulation and joint-command real-world control providing two native action representations.

B.2 Real-World PiPER-X Evaluation

One instruction-conditioned policy is trained jointly on the four tasks in Table 4. It receives wrist-mounted and third-person camera observations, proprioceptive state, and a language instruction. The tasks span object transport, target selection, insertion, and multi-stage stacking. Each task is evaluated over 100 physical trials for each reported method–checkpoint pair, using the policy trained with seed 47. Success is recorded for each trial, and the real-world average assigns equal weight to the four tasks.

B.3 Compared Methods

All comparisons use the same $\pi_{0.5}$ initialization and LoRA parameterization within each setting. The methods share the training data available for adaptation, the native action representation, and the common optimization schedule. Each comparison changes how supervision is selected or weighted.

Table 4: Real-world manipulation tasks on PiPER-X.

Task	Manipulation objective
Placement	Pick up an orange and place it onto a plate.
Selection	Select the instruction-specified block among candidate blocks and place it onto a plate.
Insertion	Grasp one cup and insert it into another cup.
Stacking	Sequentially stack three blocks into a stable vertical structure.

LoRA. The reference uses baseline sampling u and the complete-chunk flow-matching objective with fixed subspace coefficients. Its adapters follow the common configuration in Section C.2 and the LoRA parameterization (Hu et al., 2022).

SIEVE and DataMIL. SIEVE (Wu et al., 2026) selects demonstrations using reusable behavioral structure. DataMIL (Dass et al., 2026) estimates data utility with datamodels and target-task supervision. In our comparisons, these approaches select fixed subsets for subsequent flow-based VLA fine-tuning. They assess the benefit of concentrating updates on a curated data pool. Their method-specific selection computation is included in the fine-tuning-time accounting.

InfoBatch. InfoBatch (Qin et al., 2024) dynamically changes sample usage using training losses and rescales retained supervision. We apply its selection and rescaling to complete action chunks while retaining their internal base loss proportions. This provides a dynamic sample-level comparison to hierarchical allocation.

FAMO. FAMO (Liu et al., 2023a) balances multiple training objectives. We use three physical action-subspace losses on LIBERO and seven on PiPER-X as the balancing objectives. Its shared objective weights provide a loss-balancing comparison to HERA's chunk-specific subspace allocation.

B.4 Evaluation Metrics and Timing Protocol

For a task with binary trial outcomes z_e, success rate is

$$\mathrm{SR}(\%) = \frac{100}{n} \sum_{e=1}^{n} z_e. \tag{21}$$

We first average tasks within each LIBERO suite and then average over the three training seeds. The LIBERO domain mean weights the four suites equally. Real-world rates use a single trained policy per method and average the four task rates separately.

Fixed-target efficiency. Let $s_m(k)$ be method m's mean LIBERO success rate at evaluated update count k, and let $t_m(k)$ denote the corresponding cumulative fine-tuning time used for the reported comparison. For target τ,

$$\begin{aligned} K_m(\tau) &= \min\{k \in \mathcal{E}_m : s_m(k) \geq \tau\}, \\ T_m(\tau) &= t_m(K_m(\tau)), \qquad \mathrm{Speedup}_m(\tau) = \frac{T_{\mathrm{LoRA}}(\tau)}{T_m(\tau)}. \end{aligned} \tag{22}$$

Here $\mathcal{E}_m$ is the evaluated checkpoint set. The reported targets are 70%, 80%, and 90%; NR denotes a target unreached within 50K updates. Attainment is determined from the first evaluated checkpoint on the three-seed mean curve. Normalized time is T_m/T_{LoRA}, and update reduction is $100(1 - K_m/K_{\mathrm{LoRA}})$ at the same target. Fine-tuning-time accounting includes method-specific preparation, selection, response estimation, and allocation. Physical task execution is a separate evaluation quantity.

Matching LoRA's peak mean success. The learning-curve annotation compares update counts at the highest mean success rate attained by LoRA within 50K updates:

$$\begin{aligned} s^\star &= \max_{k \in \mathcal{E}_{\mathrm{LoRA}},\, k \le 50000} s_{\mathrm{LoRA}}(k), \\ k^\star_{\mathrm{LoRA}} &= \min\{k \in \mathcal{E}_{\mathrm{LoRA}} : s_{\mathrm{LoRA}}(k) = s^\star\}, \\ k^\star_{\mathrm{HERA}} &= \min\{k \in \mathcal{E}_{\mathrm{HERA}} : s_{\mathrm{HERA}}(k) \ge s^\star\}. \end{aligned} \tag{23}$$

The $2.02\times$ annotation reports $k^\star_{\mathrm{LoRA}}/k^\star_{\mathrm{HERA}}$. The $1.93\times$ result reports the wall-clock ratio at the fixed 80% mean-success target. Section C.5 specifies the component and cumulative overhead accounting.

Controlled learning-dependency observations. The observation study compares the prediction errors and subsequent learning benefit of chunk B before and after training on a physically related chunk A. Comparisons use matched starting states and training budgets, with background training providing the reference for additional learning benefit. Separate subspace readouts identify how the benefits of further complete-chunk training differ within B. These measurements motivate the two allocation levels evaluated by the component ablations.

C Training and Implementation Details

C.1 Data Construction and Training Seeds

Table 5 summarizes the adaptation pools. Each valid frame defines a chunk start, so adjacent chunks can share future action targets. Chunks remain within their source demonstration; terminal segments follow the native data pipeline. Reference samples, calibration probes, and physical features use training-side information. Evaluation success rates are excluded from allocation.

Table 5: Adaptation data and native action horizons.

Setting	Tasks	Demonstrations	Chunk starts	Horizon
LIBERO	40	1,693	273,465	10
PiPER-X	4	200	83,625	16

All LIBERO methods use training seeds $\{47, 48, 49\}$, with each run initialized independently from the same pretrained checkpoint. PiPER-X uses seed 47 and one shared policy trained jointly on its four tasks, with 50 demonstrations per task. Evaluation initialization and across-run averaging follow Section B.1.

C.2 $\pi_{0.5}$ LoRA Fine-Tuning

All runs initialize from $\pi_{0.5}$-base (Physical Intelligence et al., 2025) and update only LoRA parameters. Adapters cover query, key/value, and output attention projections and the gate, up, and down feed-forward projections in all 18 layers of each Transformer stack. The VLM's language Transformer uses rank 16, and the action expert uses rank 32, with unit effective adapter scaling and zero LoRA dropout. The vision encoder, action projections, time-conditioning modules, and all other non-LoRA parameters remain frozen, giving 49,987,584 trainable parameters.

The two settings share the optimizer and schedule in Table 6. The learning rate warms up for 1K updates, decays to its floor at 30K, and stays at that floor through 50K. BF16 mixed precision is used for model computation; trainable LoRA parameters and optimizer states remain FP32. Each update processes 32 complete chunks with no gradient accumulation.

C.3 HERA Configuration

Table 7 gives the allocation budgets. After 250 baseline updates, HERA refreshes every 1,000 updates at a common frozen policy and optimizer state. This allocation warmup is separate from the 1,000-update learning-rate warmup. Reference instances and draws, probe IDs, and physical

Table 6: Common policy fine-tuning configuration.

Setting	Value
Initialization	$\pi_{0.5}$-base
VLM / action-expert LoRA ranks	16 / 32
VLM / action-expert LoRA scaling parameters	$\alpha_{\mathrm{LoRA}} = 16$ / 32
Trainable parameters	49,987,584 (49.99M)
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.95)$
Epsilon / weight decay	10^{-8} / 10^{-10}
Gradient clipping	Global norm 1.0
LR warmup / cosine endpoint	1K / 30K updates
Peak / minimum LR	2.5×10^{-5} / 2.5×10^{-6}
Maximum updates / reported SR budgets	50K / 20K and 50K
Physical / effective batch	32 / 32
Precision	BF16 mixed precision; FP32 LoRA parameters and optimizer states
Hardware	One NVIDIA RTX 5090

prototypes remain fixed. Source draws are renewed at each refresh and shared across a chunk's directions. Candidate draws and the anchor and background batches are sampled independently. Final probabilities and multipliers are then cached for the next training interval.

Table 7: Default HERA supervision-allocation configuration.

Parameter	LIBERO	PiPER-X
Candidate budget M	1,024	1,024
Candidate draws / scan batch	2 / 32	2 / 32
Reference chunks / fixed draws per chunk	80 / 1	64 / 1
Reference microbatch / anchor batch	32 / 32	32 / 32
Calibration-source chunks P	8	4
Physical subspaces C / calibration directions PC	3 / 24	7 / 28
Background batches $J \times b_{\mathrm{bg}}$	4×32	4×32
Allocation warmup W / refresh interval L	250 / 1,000	250 / 1,000
Maximum adaptive mass $\gamma_{\max}$	0.25	0.25
Pool concentration cap κ_{pool}	4	4
Candidate-branch multiplier bounds $[a_-, a_+]$	$[0.5, 1.5]$	$[0.5, 1.5]$
Maximum candidate carry-over fraction	25%	25%

C.4 Native Action-Subspace Instantiation

LIBERO. The `OSC_POSE` interface uses seven physical action coordinates: three translation commands, three rotation commands, and one gripper command. With one-based indexing, the physical subspaces are

$$\mathcal{C}_{\mathrm{tr}} = \{1, 2, 3\}, \qquad \mathcal{C}_{\mathrm{rot}} = \{4, 5, 6\}, \qquad \mathcal{C}_{\mathrm{gr}} = \{7\}. \tag{24}$$

Each loss uses the corresponding normalized flow-prediction residuals. The controller's conversion of commands into joint torques preserves this policy-side partition.

PiPER-X. Six joint-position targets and an absolute gripper command form the physical action. Before normalization and flow interpolation, future joint targets are expressed relative to the state observed at the chunk's start:

$$\Delta q_{i,h,j} = q_{i,h,j}^{\mathrm{target}} - q_{i,j}^{\mathrm{obs}}, \qquad j = 1, \ldots, 6, \quad h = 1, \ldots, H. \tag{25}$$

The same observation anchors all future targets, and the gripper retains its absolute encoding. The seven one-dimensional subspaces are

$$\begin{aligned} \mathcal{S}_{\text{PiPER-X}} &= \{J_1, J_2, J_3, J_4, J_5, J_6, \mathrm{gr}\}, \\ \mathcal{C}_{J_j} &= \{j\}, \qquad \mathcal{C}_{\mathrm{gr}} = \{7\}. \end{aligned} \tag{26}$$

For both interfaces, the coefficients β_s preserve the native loss reduction and any auxiliary supervision. HERA retains unit auxiliary multipliers and adjusts the physical subspace weights under the common normalization. The sampling and weighting formulation is shared across the two interfaces.

C.5 Runtime and Overhead Accounting

Periodic computation. The candidate budget gives $1{,}024 \times 2$ chunk–draw evaluations, or 64 forward batches of size 32, per refresh. Calibration, anchor, and background responses contribute $PC + 1 + J$ complete-reference directions: 29 on LIBERO and 33 on PiPER-X. Table 8 groups the supplied reference-response timings and candidate evaluation costs. Its subtotal covers the listed components; estimator fitting, allocation solving, and one-time setup are outside this component sum.

Table 8: **LIBERO refresh-component costs on one RTX 5090.** All component times are directly measured under the default HERA configuration.

Component	Time (s)
Current-reference and virtual-anchor readouts	4.814
Anchor gradient and reference response	7.336
Calibration-source computation	1.856
Calibration-reference responses	113.681
Background gradients and reference responses	29.415
Virtual optimizer and direction transfer	0.407
Candidate evaluation: 1,024 chunks, two draws	61.947
Component subtotal	219.456

The isolated steady-state update profiling measures the incremental cost of HERA's weighted update path relative to the baseline update. These measurements are used only for overhead attribution; end-to-end time-to-target is computed from cumulative training-run wall-clock records. The independently measured steady-state wall-clock time of one complete HERA refresh under the default $M = 1024$ configuration is 219.4 s, closely matching the sum of the individually measured components in Table 8.

Cumulative cost. Counting refreshes that supply the first k completed updates gives

$$n_{\mathrm{ref}}(k) = \begin{cases} 0, & k \le W, \\ \lceil (k - W)/L \rceil, & k > W. \end{cases} \tag{27}$$

Let $T_{\mathrm{train}}(k)$ denote cumulative ordinary-training wall-clock and $T_{\mathrm{refresh}}(k)$ the additional periodic refresh cost. Total fine-tuning time is

$$T_{\mathrm{total}}(k) = T_{\mathrm{train}}(k) + T_{\mathrm{refresh}}(k). \tag{28}$$

For overhead accounting, T_{extra} includes refresh computation and the incremental weighted-update cost already contained in the ordinary-training timer. The overhead fraction is

$$\mathrm{OH}(k) = \frac{T_{\mathrm{extra}}(k)}{T_{\mathrm{total}}(k)}. \tag{29}$$

At the 80% target, HERA reaches the target after 13K updates and executes 13 refreshes. The measured 219.4 s refresh time contributes 0.7923 h in total. The cumulative ordinary-training record is 12.0783 h, yielding 12.8706 h of total fine-tuning time. The measured incremental weighted-update cost contributes 0.0435 h within the ordinary-training record; together with refresh computation, this gives 0.8358 h of method-specific computation, corresponding to 6.49% overhead. LoRA requires 24.8487 h to reach the same target at 28K updates. HERA therefore requires 0.518 times the LoRA wall-clock time and achieves a $1.93\times$ fine-tuning speedup.

D Additional Ablation Studies

We further examine how candidate coverage, refresh frequency, and allocation granularity affect HERA's fine-tuning efficiency, and evaluate the response estimates guiding these decisions. Unless varied, experiments follow the common $\pi_{0.5}$/LIBERO configuration. Time is normalized to LoRA at the 80% success target, and overhead denotes the fraction of total fine-tuning time spent on method-specific computation up to the same target. Rows marked † indicate the default HERA configuration.

D.1 Candidate-Budget Sensitivity

We vary the candidate budget M while fixing the refresh interval at $L = 1000$, the calibration budget, and the allocation constraints. The retained fraction and coverage-queue rule are shared across configurations. This comparison examines how the scope of candidate reassessment affects learning progress and its computational cost.

Table 9: **Candidate-budget sensitivity on LIBERO.** The refresh interval, calibration budget, and allocation constraints are fixed across configurations.

M	SR@20K (%)	SR@50K (%)	Updates @80%	Time @80%	OH @80%	Refresh (s)
512	84.70	94.05	14K	0.55	5.7%	189.7
1,024†	85.43	94.43	**13K**	**0.52**	6.5%	219.4
2,048	85.55	**94.50**	**13K**	0.53	8.1%	282.9
4,096	**85.58**	94.48	**13K**	0.55	11.1%	405.8

Table 9 shows diminishing returns from expanding the candidate pool. Increasing M from 512 to 1,024 reduces the updates required to reach 80% success from 14K to 13K and lowers normalized time from 0.55 to 0.52. Further expansion to 2,048 or 4,096 candidates improves SR@50K by at most 0.07 percentage points over the default, with the target-attainment step remaining at 13K. Meanwhile, increasing M from 1,024 to 4,096 raises the measured per-refresh cost from 219.4 s to 405.8 s and overhead from 6.5% to 11.1%. The default budget therefore provides the lowest time to target among the tested configurations, capturing most of the learning benefit of a broader candidate search at substantially lower cost.

D.2 Refresh-Interval Sensitivity

With $M = 1024$ fixed, we vary the number of ordinary optimizer updates between consecutive allocation refreshes. Reference readouts, calibration responses, the estimator, response moments, and the final allocation are refreshed together. Each configuration retains the same 250-update initial warmup. Changing L therefore affects both the age of the allocation and the frequency of candidate rotation.

Table 10: **Refresh-interval sensitivity on LIBERO.** The candidate budget is fixed at $M = 1024$, and all response and allocation quantities share the specified refresh interval.

L	SR@20K (%)	SR@50K (%)	Updates @80%	Time @80%	OH @80%
500	**85.60**	**94.50**	**13K**	0.55	11.9%
1,000†	85.43	94.43	**13K**	**0.52**	6.5%
2,000	84.75	94.05	14K	0.54	3.5%

Table 10 identifies a balance between timely reassessment and recurring computation. Refreshing every 500 updates raises SR@50K from 94.43% to 94.50%, while retaining the same 13K updates to the target. Its 11.9% overhead increases normalized time to 0.55. Extending the interval to 2,000 updates reduces overhead to 3.5%, accompanied by lower success rates at both update budgets and an

additional 1K updates to reach 80% success. The default interval of 1,000 updates achieves the lowest normalized time, maintaining effective supervision reassessment with moderate computational overhead.

D.3 WEIGHTING-ONLY ALLOCATION

To isolate within-chunk supervision allocation, the weighting-only variant fixes the sampling distribution to $p = u$ and optimizes action-subspace weights within the current candidate pool:

$$Q_{i,s} = u_i \beta_s \alpha_{i,s}, \qquad \sum_s \beta_s \alpha_{i,s} = 1, \qquad \alpha_{i,s} = 1 \quad (i \notin \mathcal{V}). \tag{30}$$

Optimization is performed directly under the fixed row sums $\sum_s Q_{i,s} = u_i$. Physical multipliers use the same bounds as HERA, and auxiliary multipliers remain one. The reference objective, rotating candidate pool, and response-estimation protocol are retained. Weighted chunks are visited according to their baseline probability mass $U_{\mathcal{V}} = \sum_{i \in \mathcal{V}} u_i$, isolating subspace adaptation under baseline chunk visitation.

Table 11: **Single-level and joint supervision allocation on LIBERO.** Sampling only adapts chunk frequency; weighting only fixes $p = u$ and adapts action-subspace supervision within the candidate pool. All variants share the common policy-training configuration.

Variant	SR@20K (%)	SR@50K (%)	Updates @80%	Time @80%	OH @80%
LoRA	76.05	91.23	28K	1.00	0.0%
Sampling only	82.40	92.80	17K	0.63	4.3%
Weighting only	81.60	92.45	20K	0.77	5.9%
HERA	**85.43**	**94.43**	**13K**	**0.52**	6.5%

Table 11 demonstrates the complementary benefits of the two allocation levels. Weighting only improves SR@20K over LoRA by 5.55 percentage points and reduces the updates to 80% success from 28K to 20K. Sampling only reaches the target in 17K updates by changing which chunks receive training opportunities. Full HERA coordinates these opportunities with each chunk's supervision composition, reaching the target in 13K updates and 0.52 normalized time, compared with 0.63 for sampling only and 0.77 for weighting only. It also achieves the highest success rates at both update budgets. These gains demonstrate the value of jointly adapting chunk visitation and within-chunk supervision in the budgeted setting.

D.4 RESPONSE-ESTIMATION DIAGNOSTICS

We evaluate the shared response estimator at three frozen training snapshots: 1.25K, 10.25K, and 20.25K updates. Each snapshot uses 24 diagnostic source directions excluded from the corresponding ridge fit. For a measured stabilized-map response y_j and prediction $\widehat{y}_j$, relative error is $\|\widehat{y}_j - y_j\|/(\|y_j\| + \varepsilon)$; cosine agreement evaluates their directional alignment on nonzero responses.

A separate paired virtual-update check compares the signs of predicted reference-loss improvements with those produced by the actual policy optimizer. This check assesses the connection between response prediction and optimizer-induced improvement, including the stabilized-map approximation. All diagnostics use training-side data and leave the policy training state unchanged.

Table 12 shows consistent response prediction across the three examined stages. Relative error ranges from 0.14 to 0.18, while cosine agreement ranges from 0.90 to 0.94, indicating that the estimator captures the direction of reference corrections beyond its calibration directions. Improvement-sign agreement remains between 87.5% and 91.7%, linking these predictions to the direction of reference-loss change under the actual optimizer. Together, these diagnostics support response sharing across source supervision and its use as a common signal for dynamic chunk sampling and action-subspace weighting.

Table 12: **Response-estimation diagnostics across training stages.** Relative error and cosine agreement evaluate excluded diagnostic directions. Sign agreement compares predicted reference-loss improvements with paired virtual evaluations of the actual optimizer.

Snapshot	Update	Diagnostic directions	Relative error ↓	Cosine agreement ↑	Sign agreement ↑
Early	1.25K	24	0.18	0.90	87.5%
Middle	10.25K	24	**0.14**	**0.94**	**91.7%**
Late	20.25K	24	0.16	0.92	87.5%

E Limitations and Future Work

HERA's response estimates are local to the policy and optimizer state at a refresh. Their quality depends on the fixed reference, the physical correspondence features, and the calibration directions. Large changes between refreshes can reduce the accuracy of the predicted learning responses. The bounded candidate pool controls reassessment cost while restricting fine-grained reallocation to a subset of chunks at each snapshot. These dependencies motivate the reference, probe, and budget diagnostics described in the supplementary experiments.

The current evaluation covers $\pi_{0.5}$ with Cartesian-command LIBERO and joint-command PiPER-X adaptation. A natural extension is to test the same allocation principle across additional policy architectures, embodiments, and task distributions. Further directions include allocating candidate and probe budgets according to response uncertainty, adapting the refresh interval to changes in learning state, and learning physical correspondences that transfer across control interfaces. These extensions can broaden the operating range while preserving the central objective: shorter total fine-tuning time to a specified policy quality.