WHAT TO LEARN AND WHO TO ASK: ADAPTIVE CURRICULUM AND EXPERT ALLOCATION FOR INTERACTIVE IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive Imitation Learning (IIL) enables agents to master complex behaviors through real-time human interventions. However, its scalability is hindered by high supervision costs. Prior work has studied adaptive curricula and teacher selection, but not their coupled non-stationarity in intervention-based online IIL, where scenario value and supervision needs evolve with the learner. We propose COACH (Cost-Optimized Adaptive Curriculum & Human-supervision), which formulates this setting as a coupled non-stationary decision problem. COACH tracks scenario-specific learning dynamics to determine what to train next, while routing episode-level supervision according to the selected scenario, learner proficiency, and expert cost. COACH adopts a factorized but context-coupled architecture and uses Lazy Re-evaluation to selectively refresh non-stationary curriculum estimates. Extensive experiments on MetaDrive and MetaWorld demonstrate that COACH outperforms state-of-the-art baselines in task performance and cost reduction, results further confirmed through validation with human participants. Finally, we show COACH generalizes to skill-to-task matching, autonomously identifying human domain specialists without cost incentives. The corresponding source code is available at `https://anonymous.4open.science/r/coach-D250`.

1 INTRODUCTION

Interactive Imitation Learning (IIL) enables agents to master complex, safety-critical behaviors through real-time expert intervention. By integrating human guidance directly into the control loop, IIL avoids the need for complex reward engineering, enabling agents to learn intricate skills directly from human guidance. However, the scalability of this paradigm is strictly limited by the high cost of human supervision. To render IIL economically viable at scale, a training framework must optimize two distinct decision variables at every step: ***What*** *scenario should the agent learn from next? And* ***Who*** *should provide the supervision for that scenario?*

The first question–*what to learn*–has been extensively explored in curriculum learning, where algorithms prioritize scenarios based on uncertainty or difficulty Li et al. (2023); Bronstein et al. (2023); Yu et al. (2020). Similarly, the second question–*who to ask*–has also been studied in crowdsourcing and active learning, where algorithms select from diverse labelers to maximize cost-effectiveness Huang et al. (2017); Gao & Saar-Tsechansky (2020); Chakraborty (2020). However, these methods typically treat task difficulty or expert utility as relatively static. This assumption does not hold in online IIL, where the policy continuously evolves: the value of a training scenario changes as the agent learns, and the supervision required for that scenario changes with the agent's proficiency. *As the agent improves, tasks that once needed expensive senior experts can be handled by lower-cost junior experts.* Existing static methods fail to capture this temporal shift, resulting in systemic inefficiency by either wasting supervision budget on over-qualified experts or risking ineffective supervision with under-qualified ones. Thus, *what to learn* and *who to ask* must adapt jointly to the evolving learner.

Achieving this dynamic joint optimization in IIL presents a fundamental challenge. Naively modeling the decision space as a joint selection of scenarios and experts results in a combinatorial explosion, where the number of possible pairs scales multiplicatively. In Human-in-the-Loop learning, exploring

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

this vast joint space is prohibitively inefficient, as every evaluation requires a costly simulation rollout and potential human intervention. As the scenario and expert spaces grow, direct joint exploration therefore becomes increasingly sample-inefficient under a limited interaction budget.

To address this challenge, we propose COACH (**C**ost-**O**ptimized **A**daptive **C**urriculum & **H**uman-supervision), which factorizes the joint decision into two coupled stages that continuously adapt with the learner. 1) A Curriculum Bandit solves the What question by selecting scenarios according to the learner's scenario-specific learning dynamics, with a Lazy Re-evaluation Bandit efficiently tracking their changes. 2) A Supervision Bandit solves the Who question by selecting an expert conditioned on both the chosen scenario and the learner's current proficiency. This factorization reduces the exploration burden of the scenario–expert product space while retaining their dependency through context-conditioned expert allocation. With the feedback of learning policy to update both bandits, COACH ensures that high-cost expert capital is deployed just-in-time, only when the curriculum demands a task that exceeds the agent's current capabilities. In summary, our main contributions are:

- We formulate curriculum scheduling and heterogeneous supervision in online IIL as a coupled non-stationary decision problem, where both scenario training value and expert suitability evolve with the learner. COACH addresses this through a factorized but context-coupled architecture that preserves scenario–expert dependency while reducing product-space exploration.
- We propose a Lazy Re-evaluation Bandit to dynamically sample a subset of scenarios to refresh their curriculum estimates, enabling the system to accurately track the agent's non-stationary learning velocity with minimal computational overhead.
- Experiments on MetaDrive and MetaWorld show that COACH achieves superior task performance and lower supervision costs than baselines. We further validate COACH with human participants, demonstrating its autonomous matching of domain specialists to tasks without cost incentives.

2 Related Work

Interactive Imitation Learning. Interactive imitation learning methods such as DAgger Ross et al. (2011) and HG-DAgger Kelly et al. (2019) improve policy learning through expert feedback, while recent off-policy approaches further enhance data efficiency Li et al. (2022b); Peng et al. (2023); Ling et al. (2025). These methods primarily focus on how to learn from interventions, whereas COACH additionally adapts which scenarios to train on and which experts to query.

Automated Curriculum Learning. Curriculum learning prioritizes training tasks according to difficulty, uncertainty, or learning progress Li et al. (2023); Yu et al. (2020); Bronstein et al. (2023); Matiisen et al. (2019); Portelas et al. (2020). Bandit-based methods further adapt task selection online Graves et al. (2017); Feng et al. (2020), but repeated evaluation can be costly in human-in-the-loop settings. COACH addresses this with selective re-evaluation of non-stationary curriculum signals.

Multi-Expert Learning. Heterogeneous supervision has been studied in socially guided learning, crowdsourcing, and active learning. SGIM Nguyen & Oudeyer (2012); Nguyen et al. (2021) jointly selects goals, strategies, and teachers according to competence progress, while CEAL Huang et al. (2017) and related methods Gao & Saar-Tsechansky (2020) balance sample utility, annotator quality, and cost. Unlike these settings, COACH performs episode-level expert allocation for an evolving online control policy, conditioning supervision on both scenario context and learner proficiency.

3 Problem Formulation

We address Interactive Imitation Learning (IIL) with heterogeneous human supervision. The agent learns a policy π_θ by imitating expert interventions provided during training.

Curriculum and Scenario Space. Let $\mathcal{Z} = \{z_i\}_{i=1}^{N}$ denote a library of training scenarios. Each scenario (i.e., curriculum) z defines an environmental configuration (e.g., road topology or object geometry) that determines task difficulty and whose training value may change as the learner evolves.

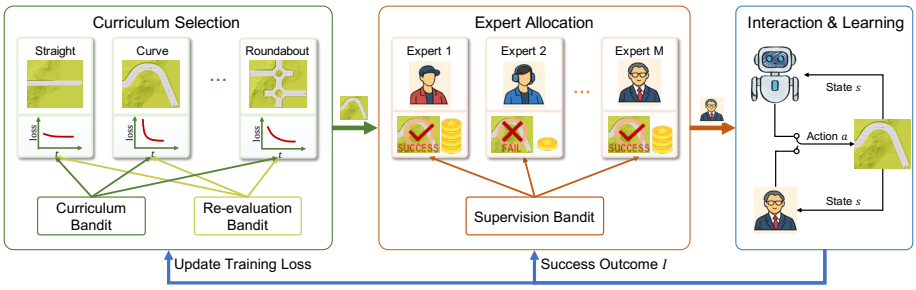


Figure 1: The COACH framework architecture. (Left) The Curriculum Bandit selects scenarios according to learning velocity, while Lazy Re-evaluation selectively refreshes scenario-level curriculum estimates. (Middle) The Supervision Bandit allocates an expert conditioned on scenario context and learner proficiency. (Right) The agent learns from expert intervention in the selected scenario, with training loss and success outcomes fed back to update the curriculum and supervision modules.

Heterogeneous Expert Pool. Let $\mathcal{E} = \{e_m\}_{m=1}^{M}$ denote a pool of experts, where each e_m has a specific per-unit-time supervision cost $C(e_m)$. This pool represents a spectrum of proficiency levels that requires a trade-off between supervision quality and supervision cost.

Optimization Objective. Our goal is to select the sequence of scenarios (z_k) and experts (e_k) from the product space $\Omega = \mathcal{Z} \times \mathcal{E}$ to maximize the cumulative Net Utility $\mathcal{U}$ over a training budget K:

$$J = \mathbb{E}\left[\sum_{k=1}^{K} \mathcal{U}(z_k, e_k)\right]. \tag{1}$$

The utility $\mathcal{U}(z, e)$ balances two factors: 1) *Task Performance*: Failed allocations incur a severe penalty, regardless of the cost savings. 2) *Supervision Cost*: Among successful pairs, the utility is maximized by minimizing the supervision cost $C(e)$.

Directly optimizing Eq. (1) over the full product space is challenging because the utility of each scenario–expert pair is unknown a priori, and both can evolve with the learner. Evaluating a pair requires environment interaction and expert supervision. As the number of scenarios and experts grows, sufficiently exploring the resulting product space becomes increasingly sample-inefficient under a limited interactive-training budget. This motivates a more scalable decision structure that reduces exploration cost while retaining the dependency between scenario selection and expert allocation.

4 Methodology

To address this challenge, COACH adopts a *factorized but context-coupled* decision structure (Fig. 1). The Curriculum Bandit (Left) first selects a scenario according to the learner's evolving training dynamics. Conditioned on the selected scenario and the learner's current proficiency, the Supervision Bandit (Middle) then allocates an expert using a contextual utility model. Thus, factorization reduces the exploration burden of the scenario–expert product space while preserving the dependency between scenario selection and expert allocation, at the cost of potentially sacrificing some joint-space optimality under a limited interactive-training budget.

4.1 Curriculum Bandit

We formulate scenario selection as an adversarial multi-armed bandit problem, where the curriculum prioritizes scenarios exhibiting the most significant current learning dynamics. We next address two questions: *how to quantify such dynamics* and *how to track them efficiently as the policy evolves*.

4.1.1 Curriculum Reward: Learning Velocity

A useful curriculum signal should reflect not the absolute difficulty of a scenario, but how informative additional training on that scenario is for the current learner. Absolute imitation loss alone is

insufficient for this purpose: a high loss may correspond either to a learnable scenario with substantial room for improvement or to a scenario on which learning has already plateaued. Conversely, a previously mastered scenario may later deteriorate because of interference from training on other tasks. We therefore characterize each scenario by the temporal evolution of its imitation loss, i.e., *learning velocity* Matiisen et al. (2019), yielding a dynamic and scenario-specific measure of the learner's current training dynamics.

For any given scenario z at the current training episode k, we examine the imitation loss $\mathcal{L}$ over a sliding window of the most recent τ evaluations. Specifically, the imitation loss $\mathcal{L}_{k,z}$ is defined by

$$\mathcal{L}_{k,z} = \mathbb{E}_{(s,a)\in\mathcal{D}_z}[-\log \pi_{\theta_k}(a|s)], \tag{2}$$

where $\mathcal{D}_z$ denotes the set of expert intervention demonstrations in scenario z, and π_{θ_k} denotes the learning policy in episode k. This loss is a generic form; in practice, we use policy-specific curriculum losses for MetaDrive and MetaWorld (Appendix E.3). We quantify the velocity $\nu_{k,z}$ as the temporal slope of the loss, estimated by Ordinary Least Squares Ruppert (2004):

$$\nu_{k,z} = \frac{\sum_{t=0}^{\tau-1}(t-\bar{t})(\mathcal{L}_{k-t,z}-\bar{\mathcal{L}}_z)}{\sum_{t=0}^{\tau-1}(t-\bar{t})^2}, \tag{3}$$

where $\bar{t}$ and $\bar{\mathcal{L}}_z$ represent the mean time index and mean loss of training on z over the history τ, respectively. This windowed slope reduces sensitivity to short-term stochastic fluctuations. We then define the curriculum reward $r_{k,z}$ as the magnitude of this slope:

$$r_{k,z} = |\nu_{k,z}|. \tag{4}$$

Importantly, $r_{k,z}$ should be interpreted as a proxy for the magnitude of scenario-level learning dynamics rather than a signed estimate of policy improvement or net utility. A large $|\nu_{k,z}|$ can indicate either rapid learning, where imitation loss decreases quickly, or skill degradation, where the loss increases because of forgetting or interference. Both cases warrant additional curriculum attention. In contrast, a near-zero slope indicates that the scenario is currently stable, whether because the skill has been mastered or because learning has temporarily plateaued, and therefore provides less immediate training value.

Loss velocity also provides dense feedback directly from the imitation-learning process, without requiring additional task-level evaluations. Appendix C.2 compares it with policy-entropy change and success-rate improvement.

4.1.2 Adversarial Curriculum Selection

With the curriculum reward $r_{k,z}$ defined as the learning velocity (Eq. (4)), our **Curriculum Bandit** is instantiated with the Exp3 algorithm to maintain a weight vector $w_k \in \mathbb{R}^N$ on the scenario library $\mathcal{Z}$. At each episode k, we sample a training scenario z_k^* from the probability distribution p_k:

$$p_k^z = (1-\gamma)\frac{w_k^z}{\sum_{z'\in\mathcal{Z}} w_k^{z'}} + \frac{\gamma}{|\mathcal{Z}|} \tag{5}$$

where $\gamma \in (0,1]$ is the exploration factor. To account for the bias introduced by sampling, we update the weights w_k using the Inverse Propensity Scoring (IPS) estimator Auer et al. (2002). Specifically, for the selected scenario z^*, the observed reward r_{k,z^*} is scaled by $1/p_k^{z^*}$ to update the corresponding weight exponentially, driving the policy towards scenarios with the largest estimated learning dynamics.

4.1.3 Lazy Curriculum Re-evaluation

Because the policy is continuously updated, the learning velocity of every scenario is inherently non-stationary. Ideally, $L_{k,z}$ and $r_{k,z}$ would be recomputed for all $z \in \mathcal{Z}$ after every policy update. However, exhaustive re-evaluation is computationally expensive, particularly when policy inference and loss evaluation involve large models. We therefore maintain cached curriculum-reward estimates and selectively refresh only a small subset of N_{eval} scenarios at each episode, where $N_{\text{eval}} \ll |\mathcal{Z}|$.

We formulate this refresh process as a second Exp3 bandit with weights u_k. Its objective is to allocate more evaluation effort to scenarios whose curriculum estimates are likely to become stale quickly.

Scenarios with large $|\nu_{k,z}|$ are undergoing rapid learning or degradation and therefore require more frequent refreshes, whereas scenarios with near-zero velocity can typically tolerate longer evaluation intervals. Accordingly, the re-evaluation bandit uses the same velocity magnitude in Eq. (4) as its feedback signal. At the end of episode k, it samples N_{eval} distinct scenarios $\{z^i_{\text{eval}}\}_{i=1}^{N_{\text{eval}}}$ according to the Exp3 distribution induced by u_k, recomputes their current imitation losses and learning velocities, and updates both the cached curriculum estimates and the re-evaluation weights through IPS.

Importantly, Exp3 retains explicit exploration over all scenarios, so even low-priority scenarios continue to have a non-zero probability of being refreshed. If a previously mastered scenario is later re-evaluated after skill degradation, the resulting increase in $|\nu_{k,z}|$ raises its subsequent re-evaluation priority and allows the Curriculum Bandit to redirect training toward it. Thus, detection of forgetting is not instantaneous for an unevaluated scenario; rather, N_{eval} controls the trade-off between estimate freshness and re-evaluation overhead; see Appendix C.2.

4.2 Supervision Bandit

Once scenario z^*_k is selected, the framework must determine who provides the intervention. This resource allocation problem seeks to balance task success and supervision cost when selecting an expert e_m. We formulate this as a Neural Contextual Bandit to capture the non-linear interactions of control tasks—A junior expert might handle high speeds or sharp curves individually but fail when both occur simultaneously—which standard linear models often miss.

4.2.1 Context Representation

For each episode k, we construct a context vector x_k that couples task difficulty with the learner's evolving capability. This vector consists of: 1) *Environmental Complexity*: Semantic features from z^*_k, such as road curvature and traffic density in driving, or object geometry in manipulation. 2) *Agent Proficiency*: The agent's empirical success rate over the most recent T training episodes across all selected scenarios. This time-dependent context allows the bandit to recognize when a scenario previously requiring a senior expert can now be handled by a junior expert as the agent improves.

4.2.2 Cost-Sensitive Utility Modeling

We define a utility function to balance task success against supervision cost. Let $I_k \in \{0, 1\}$ denote the binary success outcome at episode k, and $C_k(e)$ denote supervision cost of expert e at episode k, computed from its per-unit-time cost and the actual episode duration. The utility is defined as:

$$U_{k,e} = I_k \cdot \exp(-\kappa \cdot C_k(e)) - (1 - I_k) \cdot \psi, \tag{6}$$

where $\kappa > 0$ is a cost-sensitivity coefficient and $\psi > 0$ is a severe failure penalty. This formulation enforces that: the failure penalty ψ discourages unsuccessful allocations, while the exponential term favors lower-cost supervision when the task succeeds.

4.2.3 NeuralUCB with Weighted Replay

To estimate the utility $U_{k,e}$, we instantiate the Supervision Bandit with NeuralUCB Zhou et al. (2020) that approximates rewards using a shared neural network $f_\phi(x, e)$. We select the expert e that maximizes the upper confidence bound:

$$e^* = \arg\max_{e \in \mathcal{E}} \left(f_\phi(\mathbf{x}_k; e) + \alpha \sqrt{g_{k,e}^\top \Lambda_k^{-1} g_{k,e}} \right) \tag{7}$$

where $g_{k,e} = \nabla_\phi f_\phi(\mathbf{x}_k; e)$ denotes the gradient of the network output with respect to the weights, and α is a hyperparameter governing the exploration intensity. We initialize the Hessian matrix as $\Lambda_1 = \delta I$, where $\delta > 0$ is a regularization parameter ensuring invertibility. In practice, we employ a diagonal approximation of Λ_k and update it recursively as $\Lambda_{k+1} = \lambda \Lambda_k + g_{k,e^*} g_{k,e^*}^\top$.

To address the non-stationarity of the agent's proficiency, we implement *Weighted Experience Replay* Schaul et al. (2015). Upon observing the realized utility U_{k,e^*}, we update the network parameters ϕ by minimizing a time-weighted loss function:

$$\mathcal{L}_k(\phi) = \sum_{(j, \mathbf{x}_j, e, U_{j,e}) \in \mathcal{B}} \lambda^{k-j} \left(f_\phi(\mathbf{x}_j; e) - U_{j,e} \right)^2 + \mu \|\phi\|^2, \tag{8}$$

Algorithm 1 COACH Algorithm

Input: Scenario library $\mathcal{Z}$, Expert pool $\mathcal{E}$, Exploration rates γ, γ_{eval}, Forgetting factor λ, Lazy Sampling Size N_{eval}
Initialize: Curriculum bandit weights $\mathbf{w}_1 \leftarrow \mathbf{1}_{|\mathcal{Z}|}$, Re-evaluation bandit weights $\mathbf{u}_1 \leftarrow \mathbf{1}_{|\mathcal{Z}|}$, Hessian $\Lambda_1 \leftarrow \delta I$, Randomize Expert Utility Network f_ϕ, Buffer $\mathcal{B} = \emptyset$
for episode $k = 1$ to K **do**
  Compute scenario selection prob p_k^z via Eq. (5) *// Curriculum Selection (Exp3)*
  Sample training scenario $z_k^* \sim p_k$
  Extract context $\mathbf{x}_k$ from z_k^* and agent proficiency *// Expert Allocation (NeuralUCB)*
  Estimate uncertainty $\sigma_{k,e} = \sqrt{g_{k,e}^\top \Lambda_k^{-1} g_{k,e}}$ for all $e \in \mathcal{E}$
  Select expert e^* for scenario z_k^* via Eq. (7)
  Rollout π_{θ_k} on z_k^* with expert e^*, observe outcome I_k and cost $C(e^*)$ *// Interaction & Learning*
  Compute Utility U_{k,e^*} via Eq. (6)
  Update agent policy $\pi_{\theta_{k+1}}$ via HG-Dagger
  Store experience $(k, \mathbf{x}_k, e^*, U_{k,e^*})$ in $\mathcal{B}$
  Update ϕ by minimizing weighted loss via Eq. (8) *// Feedback & Bandits Update: 1. Update Supervision Bandit*
  Update Hessian $\Lambda_{k+1} = \lambda \Lambda_k + g_{k,e^*} g_{k,e^*}^\top$
  Compute cached pre-update velocity reward r^* for z_k^* via Eq. (4) *// 2. Update Curriculum Bandit*
  Update weights $\mathbf{w}_{k+1}$ via IPS using r^* and $p_k(z_k^*)$
  Compute evaluation probability $q_k^z \propto u_k^z$ via Exp3 *// 3. Update Lazy Re-Evaluation Bandit (Exp3)*
  Sample evaluation subset $\mathcal{Z}_{eval} \sim q_k$
  for each $z \in \mathcal{Z}_{eval}$ **do**
    Compute imitation loss $\mathcal{L}_{k,z}$, learning velocity $\nu_{k,z}$, and velocity reward r_z via Eqs. (2), (3), and (4), respectively
    Update weights $\mathbf{u}_{k+1}$ via IPS using r_z and q_k^z
  end for
end for

where $\mathcal{B}$ denotes the replay buffer in which each item $(j, \mathbf{x}_j, e, U_{j,e})$ denotes the context of select arm $\mathbf{x}_j$, assigned expert e and corresponding $U_{j,e}$ utility reward of j-th episode ($1 \leq j \leq k$). μ is the regularization term, and $\lambda \in (0, 1)$ serves as a forgetting factor. By exponentially decaying the influence of historical observations, this objective forces the bandit to prioritize recent interactions, allowing it to adaptively forget outdated expert assessments as the agent's proficiency improves.

4.3 The COACH Algorithm

Algorithm 1 summarizes COACH. At each episode, the Curriculum Bandit selects a scenario, after which the Supervision Bandit allocates an expert conditioned on scenario context and learner proficiency. The resulting intervention data update the policy via HG-DAgger, while outcome, learning-velocity, and selective re-evaluation feedback update the supervision, curriculum, and re-evaluation bandits, respectively.

Theoretical Performance: We analyze a composite surrogate regret combining curriculum regret with supervision regret against the best expert at each realized context. Under the stated assumptions, including normalized pre-decision curriculum feedback and calibrated supervision confidence widths, appropriate parameter scaling gives an upper bound consisting of a sublinear learning term and an explicitly characterized additive residual term accounting for lazy evaluation and online adaptation. Detailed definitions, assumptions, and proofs are provided in Appendix A. The bound characterizes the surrogate bandit decisions rather than final policy performance.

5 Experiments

5.1 Settings

Environments. We evaluate on MetaDrive and MetaWorld (Appendix B), training one shared policy over 10 scenarios and testing on 50 unseen variants. MetaDrive uses 10 road topologies with randomized geometry and traffic under a 30k-step budget (∼60 min), while MetaWorld uses 10 Sawyer manipulation tasks with randomized initial states under a 15k-step budget (∼30 min). Experts intervene upon collision risk in MetaDrive and intent misalignment in MetaWorld.

Expert Configuration. We employ synthetic and human expert pools, both categorized into Junior, Intermediate, and Senior levels with per-unit-time supervision costs of 1, 5, and 10, respectively.

Table 1: Comparison results on MetaDrive and MetaWorld. Results report mean ± std over 3 random seeds, each generating a distinct test set of 50 scenarios.

Method	MetaDrive					MetaWorld			
	Training		Testing			Training		Testing	
	Expert Data	Expert Cost	Episodic Return	Safety Cost	Success Rate	Expert Data	Expert Cost	Episodic Return	Success Rate
Random	4056	336	296.59 ± 11.0	3.50 ± 1.8	0.60 ± 0.06	4321	229	1727.82 ± 134.0	0.63 ± 0.02
Heuristic	4358	441	312.94 ± 15.9	18.61 ± 15.8	0.64 ± 0.05	5042	241	1887.45 ± 75.4	0.58 ± 0.04
Geometric	3645	342	319.20 ± 14.1	4.84 ± 1.4	0.70 ± 0.06	4163	231	1928.30 ± 246.2	0.68 ± 0.08
DQM	3505	303	313.44 ± 8.3	3.30 ± 1.6	0.72 ± 0.02	3461	198	1974.08 ± 179.9	0.71 ± 0.03
RgC	3429	302	331.22 ± 12.6	3.48 ± 0.7	0.78 ± 0.06	3145	201	2130.54 ± 165.8	0.82 ± 0.06
SMT	3677	311	340.53 ± 9.30	1.08 ± 0.51	0.807 ± 0.05	2821	197	1953.68 ± 72.4	0.740 ± 0.03
SGIM	3219	270	334.39 ± 7.2	3.60 ± 0.9	0.81 ± 0.01	3111	228	2100.86 ± 87.9	0.85 ± 0.02
CEAL	3087	311	341.15 ± 14.6	2.33 ± 2.3	0.81 ± 0.02	2815	244	2459.74 ± 140.9	0.88 ± 0.02
Joint MAB	3067	293	343.88 ± 10.4	2.35 ± 1.5	0.85 ± 0.03	2968	219	2314.58 ± 149.3	0.88 ± 0.03
SMT+COACH	2876	256	358.00 ± 1.10	0.96 ± 0.27	0.887 ± 0.01	2832	185	2310.98 ± 158.49	0.865 ± 0.02
COACH (Ours)	**2783**	**242**	$\mathbf{360.50 \pm 2.5}$	$\mathbf{0.43 \pm 0.04}$	$\mathbf{0.92 \pm 0.01}$	**2665**	**173**	$\mathbf{2592.98 \pm 183.3}$	$\mathbf{0.92 \pm 0.04}$

1) *Synthetic Expert Pool* (Main Benchmark). We trained three Reinforcement Learning (RL) agents as synthetic experts to isolate algorithmic performance from human variance (e.g., fatigue, attention drift). To ensure a clear proficiency gradient, the RL agents were trained in scenarios of varying difficulty, confirming distinct test-set success rates of $12.0\%/44.0\%/94.0\%$ in MetaDrive and $11.0\%/45.0\%/97.0\%$ in MetaWorld.

2) *Human Participants* (Real-World Validation). We recruited nine college participants (age: 24.67 ± 1.15 years) divided into three groups based on varying training duration (5, 20, and 60 minutes). Their validation scores align with the synthetic tiers: $6.0\%/62.0\%/88.0\%$ in MetaDrive and $23.0\%/53.0\%/95.0\%$ in MetaWorld.

Metrics. We primarily report *episodic return*, *safety cost* (in MetaDrive), *success rate*, *Expert Data* (counts of intervention state–action pairs), and *Expert Cost* (accumulating the assigned expert's per-unit-time supervision cost over all training steps).

Baselines. We compare COACH with three groups of baselines. (1) *Passive curriculum*: Random uniformly samples scenarios; Heuristic Li et al. (2023) follows a predefined easy-to-hard schedule; and Geometric Bronstein et al. (2023) samples according to a fixed difficulty-based distribution. (2) *Active curriculum*: DQM Feng et al. (2020) prioritizes scenarios with high success-rate variation; RgC Mysore et al. (2019) uses historical performance to guide sampling; and SMT Cho et al. (2024) adaptively schedules tasks in multi-task learning. (3) *Joint curriculum–supervision*: SGIM Nguyen et al. (2021) selects goals and teachers based on competence progress; CEAL Huang et al. (2017) jointly considers sample utility, annotator quality, and query cost; and Joint MAB treats each scenario–expert pair as an independent Exp3 arm. We additionally evaluate *SMT + COACH*, which combines SMT with COACH's expert allocation to isolate the contribution of adaptive supervision.

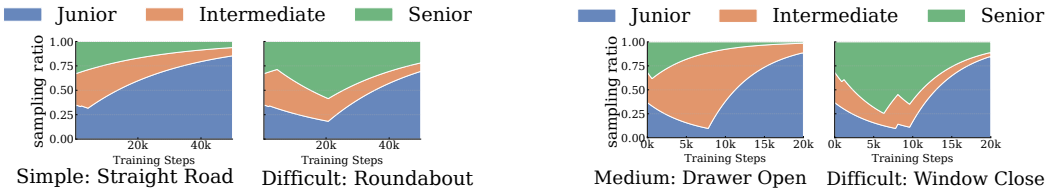
Implementation Details. We adopt HG-DAgger Kelly et al. (2019) as the base IIL algorithm. For the policy network, we utilize a Multi-Layer Perceptron (MLP) in MetaDrive and a Diffusion Policy with a 1D Conditional U-Net backbone Chi et al. (2025) in MetaWorld. All experiments were conducted on a workstation equipped with an NVIDIA GeForce RTX 4060 Laptop GPU. See Appendix E for hyperparameters, hyperparameter sensitivity analyses, and architectural details.

5.2 Comparison Results

As shown in Table 1, active curriculum methods generally improve over passive scheduling, while methods that jointly adapt curriculum and supervision achieve further gains. COACH obtains the strongest overall task performance and supervision efficiency in both domains, suggesting that efficient online IIL requires adapting both *what to learn* and *who to ask* as the learner evolves. Among joint-selection baselines, SGIM and CEAL do not explicitly condition expert utility on the evolving proficiency of an online control policy. In contrast, COACH conditions expert allocation on both the selected scenario and learner proficiency, enabling supervision to shift as the policy improves. The controlled SMT comparison further isolates the benefit of adaptive supervision. Replacing random

Table 2: Ablation study on MetaDrive and MetaWorld. Results report mean ± std over 3 random seeds, each generating a distinct test set of 50 scenarios.

Method	MetaDrive					MetaWorld			
	Training		Testing			Training		Testing	
	Expert Data	Expert Cost	Episodic Return	Safety Cost	Success Rate	Expert Data	Expert Cost	Episodic Return	Success Rate
Junior-Only	4068	**94**	192.40 ± 24.6	9.70 ± 10.2	0.22 ± 0.05	4063	**38**	1345.64 ± 109.9	0.50 ± 0.02
Inter-Only	3578	355	264.55 ± 5.6	1.38 ± 0.6	0.50 ± 0.01	3619	188	2171.22 ± 64.3	0.68 ± 0.03
Senior-Only	3001	466	**366.14 ± 4.4**	**0.48 ± 0.1**	**0.94 ± 0.01**	2703	375	**2760.90 ± 189.4**	**0.98 ± 0.01**
Rand-Expert	3123	296	328.63 ± 5.8	3.86 ± 1.1	0.74 ± 0.05	3477	196	1933.10 ± 75.1	0.61 ± 0.02
Rand-Curri	3388	313	331.69 ± 5.9	4.54 ± 1.2	0.76 ± 0.02	3415	237	2174.37 ± 109.8	0.74 ± 0.02
No-Context	3230	281	351.62 ± 12.2	5.05 ± 2.0	0.84 ± 0.07	3247	262	2225.17 ± 195.1	0.86 ± 0.02
No-LazyUpdate	**2636**	**221**	**362.46 ± 7.7**	**0.24 ± 0.12**	**0.93 ± 0.01**	**2568**	**175**	**2595.97 ± 211.6**	**0.93 ± 0.03**
COACH (Ours)	**2783**	**242**	**360.50 ± 2.5**	**0.43 ± 0.0**	**0.92 ± 0.01**	**2665**	**173**	**2592.98 ± 183.3**	**0.92 ± 0.04**



(a) MetaDrive (b) MetaWorld

Figure 2: Dynamic Expert Allocation. Stacked plots illustrate the evolution of expert sampling ratios (calculated over a 300-step sliding window) in (a) MetaDrive and (b) MetaWorld. As the agent gains proficiency in complex tasks, the framework autonomously shifts from high-cost Senior experts (green) to cost-effective Junior experts (blue).

expert assignment with COACH allocation improves success from 0.807 to 0.887 while reducing expert cost from 311 to 256 in MetaDrive; in MetaWorld, success improves from 0.740 to 0.865 and cost decreases from 197 to 185.

5.3 Ablation Study

We compare COACH against five variants: 1) Single Expert (Junior/Inter/Senior-Only); 2) Random-Expert (uniform); 3) Random-Curriculum (uniform); 4) No-LazyUpdate (full re-evaluation in curriculum bandit); and 5) No-Context (removes context features in supervision bandit).

As shown in Table 2, COACH approaches Senior-Only performance (92% vs. 94%) while reducing expert cost from 466 to 242. Randomizing either curriculum or expert allocation degrades performance, confirming the need to adapt both decisions. Removing context substantially increases safety cost, while full re-evaluation achieves comparable performance but incurs higher computational overhead; in MetaWorld, Lazy Re-evaluation reduces training time by 44.2% (53.65 to 29.91 min).

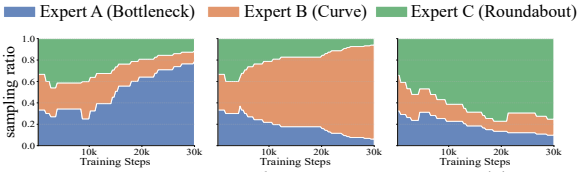
5.4 Validation with Human Experts

We validate COACH with nine human participants, obtaining results consistent with the synthetic-expert benchmarks. In MetaDrive, human-guided agents achieve an episodic return of 357.85 ± 2.57, a safety cost of 0.66 ± 0.26, and a success rate of $88.1\% \pm 3.6\%$, with a total expert cost of 219.7 ± 9.6. Similarly, in MetaWorld, agents achieve a success rate of $92.3\% \pm 1.3\%$ and a return of 2563.62 ± 91.2, incurring a human cost of 196.7 ± 6.5.

Fig. 2 illustrates the adaptive allocation across varying difficulties. As agent proficiency improves, both domains autonomously shift to cost-effective Junior experts. Specifically, MetaDrive's simple tasks transition quickly, while complex Roundabouts require Senior supervision for 20k steps. Similarly, MetaWorld's medium (Drawer Open) and difficult (Window Close) tasks initially rely on Intermediate and Senior experts ($<$10k steps) before transitioning.

Table 3: Joint MAB vs. COACH in MetaDrive.

Setting	$\lvert\mathcal{Z}\rvert \times \lvert\mathcal{E}\rvert$	Return		Success	
		Joint	COACH	Joint	COACH
Small-Hard	3×2	**423.85**	414.38	**0.92**	0.79
Base	10×3	343.88	**360.50**	0.85	**0.92**
Expert-Exp.	10×10	309.20	**332.71**	0.64	**0.84**
Large	50×10	301.57	**339.71**	0.56	**0.84**



(a) Bottleneck *(b)* Curve *(c)* Roundabout

Figure 3: Skill matching with equal expert costs.

5.5 Joint versus Factorized Optimization

To isolate the effect of factorization, we compare COACH with flat Joint MAB across scenario–expert spaces of different sizes under a fixed interaction budget (Table 3). For the 3×2 setting, we use harder, non-saturated tasks so that the six scenario–expert pairs can be sufficiently explored without both methods reaching ceiling performance. Joint MAB performs better in this small space, indicating some loss of joint-space optimality from factorization. As the space grows under the same budget, however, Joint MAB must explore many more independent pairs, leaving fewer interactions per pair. COACH reduces this exploration burden while preserving scenario–expert dependency through contextual expert allocation. The results therefore reveal a budget-dependent optimality–scalability trade-off: flat joint optimization is advantageous when the product space is sufficiently explored, whereas COACH is more effective when exploration is budget-limited.

5.6 Generalization to Skill Matching

To test whether COACH captures expert expertise rather than merely favoring lower-cost supervision, we assign equal costs to three specialists trained on Bottleneck, Curve, and Roundabout, respectively. As shown in Fig. 3, COACH progressively converges to the corresponding specialist for each scenario: Expert A accounts for over 75% of Bottleneck allocations, while Experts B and C dominate Curve and Roundabout, respectively, demonstrating context-dependent skill matching without cost incentives.

6 Discussion

Expert Quality and Late-Stage Supervision. COACH mitigates heterogeneous demonstration quality through proficiency-conditioned routing. In MetaDrive, Junior interventions selected by COACH are highly consistent with Senior actions (mean MSE 0.001438), while mature learners are frequently assigned Junior experts for monitoring or routine corrections and retain Senior supervision for harder residual errors. These diagnostics reduce, but do not eliminate, the risk of lower-tier supervision; detailed results are provided in Appendix C.1.

Modularity of Bandit Backends. Exp3 and NeuralUCB are concrete backends rather than essential restrictions of COACH. Other algorithms that support exploration under evolving partial feedback for curriculum selection and context-dependent expert utility estimation can replace them, although the corresponding theoretical guarantees may change.

Practical Limitations and Future Work. Our fixed per-unit-time supervision costs abstract away human factors such as fatigue, workload, latency, and changing availability. Weighted replay and contextual features can accommodate gradual or observable changes, while abrupt shifts may still cause adaptation delay. Future work will extend COACH to model time-varying expert quality and availability, as well as when supervision can be omitted entirely.

7 Conclusion

In this paper, we proposed COACH, a framework to jointly adapt curriculum scheduling and heterogeneous expert allocation in online IIL as the learner policy evolves. By factorizing the decision process into a hierarchical bandit architecture, COACH effectively tracks the agent's non-stationary learning velocity and ensures cost-effective supervision. Our experiments in MetaDrive and MetaWorld demonstrate that COACH improves success rates while reducing human supervision costs compared to state-of-the-art baselines.

8 AI USE STATEMENT

During the preparation of this manuscript, we used generative AI tools for language refinement and literature discovery. The AI tools were used to improve writing clarity and assist in organizing related information. All technical ideas, methodology, experiments, analyses, and conclusions were developed and verified by the authors. The authors take full responsibility for the final content of this paper.

REFERENCES

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24, 2011.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Eli Bronstein, Sirish Srinivasan, Supratik Paul, Aman Sinha, Matthew O'Kelly, Payam Nikdel, and Shimon Whiteson. Embedding synthetic off-policy experience for autonomous driving via zero-shot curricula. In *Conference on Robot Learning*, pp. 188–198. PMLR, 2023.

Shayok Chakraborty. Asking the right questions to the right users: Active learning with imperfect oracles. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3365–3372, 2020.

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

Myungsik Cho, Jongeui Park, Suyoung Lee, and Youngchul Sung. Hard tasks first: Multi-task reinforcement learning through task scheduling. In *ICML*, pp. 8556–8577, 2024.

Dieqiao Feng, Carla P Gomes, and Bart Selman. A novel automated curriculum strategy to solve hard sokoban planning instances. *Advances in Neural Information Processing Systems*, 33:3141–3152, 2020.

Ruijiang Gao and Maytal Saar-Tsechansky. Cost-accuracy aware adaptive labeling for active learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 2569–2576, 2020.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. PMLR, 2017.

Sheng-Jun Huang, Jia-Lve Chen, Xin Mu, and Zhi-Hua Zhou. Cost-effective active learning from diverse labelers. In *IJCAI*, pp. 1879–1885, 2017.

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083. IEEE, 2019.

Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(3):3461–3475, 2022a.

Quanyi Li, Zhenghao Peng, and Bolei Zhou. Efficient learning of safe driving policy via human-ai copilot optimization. *arXiv preprint arXiv:2202.10341*, 2022b.

Quanyi Li, Zhenghao Mark Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *Advances in neural information processing systems*, 36:3894–3920, 2023.

Bo Ling, Zhengyu Gan, Wanyuan Wang, Guanyu Gao, Weiwei Wu, and Yan Lyu. Faithful dynamic imitation learning from human intervention with dynamic regret minimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

Siddharth Mysore, Robert Platt, and Kate Saenko. Reward-guided curriculum for robust reinforcement learning. In *Workshop on Multi-task and Lifelong Reinforcement Learning at ICML*, 2019.

Sao Mai Nguyen and Pierre-Yves Oudeyer. Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. *Paladyn*, 3(3):136–146, 2012.

Sao Mai Nguyen, Nicolas Duminy, Alexandre Manoury, Dominique Duhaut, and Cedric Buche. Robots learn increasingly complex tasks with intrinsic motivation and automatic curriculum learning: Domain knowledge by emergence of affordances, hierarchical reinforcement and active imitation learning. *KI-Künstliche Intelligenz*, 35(1):81–90, 2021.

Zhenghao Mark Peng, Wenjie Mo, Chenda Duan, Quanyi Li, and Bolei Zhou. Learning from active human involvement through proxy value propagation. *Advances in neural information processing systems*, 36:77969–77992, 2023.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

David Ruppert. The elements of statistical learning: data mining, inference, and prediction, 2004.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International conference on machine learning*, pp. 11492–11502. PMLR, 2020.

A Theoretical Derivations

A.1 Scope and Regret Definition

We give a conditional regret bound for the two bandit decisions in COACH under the feedback timing and calibration conditions specified below. The curriculum uses learning velocity as a proxy for training value, whereas supervision uses cost-sensitive task utility. Their sum defines a *composite surrogate regret*; it is not regret for the joint objective in Eq. (1), nor a guarantee on the final imitation policy. Let $N = |\mathcal{Z}| \geq 2$, $M = |\mathcal{E}|$, and K be the number of episodes. Write z_k and m_k for the selected scenario and expert index.

Timing and normalization. Let $\mathcal{F}_{k-1}$ contain the history available before scenario selection in episode k. The curriculum reward vector r_k and the feedback vector $\hat{r}_k$ used by Exp3 are determined before this selection, although only $\hat{r}_{k,z_k}$ is supplied to its update. They may depend on the entire past, including earlier policy updates. Here $r_{k,z}$ denotes the value from a fresh evaluation of the current policy, and $\hat{r}_{k,z}$ its cached value; a fresh feedback implementation has $\hat{r}_{k,z} = r_{k,z}$. Assume the velocity magnitude in Eq. (4) has a known bound $R_v > 0$, and divide it by R_v for this analysis and the associated Exp3 update, so $r_{k,z} \in [0, 1]$. Clipping instead would define a different proxy objective.

The timing condition concerns the policy snapshot used to define the reward, not merely when its numerical value is computed. A loss evaluated after the rollout may still satisfy the condition if it uses the frozen pre-episode policy and demonstration data. However, a velocity recomputed using the updated policy $\pi_{\theta_{k+1}}$ or newly collected demonstrations generally depends on z_k and cannot be identified with the pre-decision vector r_k. Algorithm 1 places the curriculum update after policy training; the result below therefore applies to that update only if it uses the pre-decision cache or an evaluation of the frozen pre-decision state. Using post-training feedback instead requires a separate analysis of its conditional mean and its discrepancy from the benchmark reward.

For supervision, assume $C(e) \geq 0$, so the utility in Eq. (6) lies in $[-\psi, 1]$. Define $Y_{k,m} = (U_{k,m} + \psi)/(1 + \psi) \in [0, 1]$ and

$$\mu_{k,m} = \mathbb{E}[Y_{k,m} \mid \mathcal{G}_k], \tag{9}$$

where $\mathcal{G}_k$ includes the past, the selected scenario, and all expert contexts and scores, but precedes the current expert choice and outcome. This conditioning allows contexts to depend on previous supervision outcomes.

Benchmarks. We use expected external regret against a fixed scenario and dynamic pseudo-regret against the best expert at the realized context:

$$R_{\mathrm{cur}}(K) = \max_{z \in \mathcal{Z}} \mathbb{E}\left[\sum_{k=1}^{K} (r_{k,z} - r_{k,z_k})\right], \tag{10}$$

$$R_{\mathrm{sup}}(K) = \mathbb{E}\left[\sum_{k=1}^{K} (\max_{m} \mu_{k,m} - \mu_{k,m_k})\right], \tag{11}$$

$$R_{\mathrm{COACH}}(K) = R_{\mathrm{cur}}(K) + R_{\mathrm{sup}}(K). \tag{12}$$

The maximum in Eq. (10) is outside the expectation: the fixed comparator is chosen independently of the realized random history. This distinction matters because policy learning makes the reward sequence adaptive. The comparison uses rewards along COACH's actual history, not the counterfactual history produced by always training on another scenario. Likewise, Eq. (11) evaluates experts at COACH's realized contexts without conditioning on the entire future scenario sequence. In original units, the two component regrets are multiplied by R_v and $1 + \psi$, respectively.

A.2 Assumptions

Assumption A.1 (Predictable bounded feedback and controlled drift)**.** The curriculum vectors $r_k, \hat{r}_k \in [0, 1]^N$ are $\mathcal{F}_{k-1}$-measurable. There are deterministic numbers $\rho_s \geq 0$ such that, almost

surely,

$$\max_z |r_{s+1,z} - r_{s,z}| \le \rho_s, \qquad V_r := \sum_{s=1}^{K-1} \rho_s. \tag{13}$$

The normalized supervision utilities belong to $[0, 1]$.

Assumption A.2 (Lazy feedback and conditional refresh coverage)**.** An initial evaluation supplies $r_{1,z}$ for every scenario, before the K episodes being analyzed. Let $t_k(z)$ be the most recent refresh index strictly before episode k, with the initial index 1 used if no later refresh is available. In this convention, a refresh at the end of episode j stores $r_{j,z}$, evaluated using the frozen pre-episode state; this value becomes available in episode $j + 1$. If the update instead uses the current pre-decision reward as fresh feedback, set $t_k(z) = k$. Then $\hat{r}_{k,z} = r_{t_k(z),z}$ and $\hat{r}_{1,z} = r_{1,z}$. If S_j is the subset refreshed at the end of episode j, then, conditional on all information before drawing that subset,

$$\Pr(z \in S_j \mid \text{history before refresh sampling}) \ge p_{\min} > 0 \quad \text{for every } z, j. \tag{14}$$

A refresh reveals the corresponding reward exactly, and $0 < p_{\min} \le 1$.

Here $p_{\min}$ is a *subset inclusion probability*, not a per-draw sampling probability. For example, mixing uniform sampling of an N_{eval}-element subset with probability γ_{eval} guarantees $p_{\min} \ge \gamma_{\text{eval}} N_{\text{eval}}/N$. For other sampling-without-replacement schemes, the inclusion probability must be established for that scheme; it cannot be obtained by multiplying an already defined inclusion probability by N_{eval}. The same inclusion probabilities are needed for unbiased IPS updates of the re-evaluation bandit. The analysis uses its refresh coverage, not a regret guarantee for that auxiliary bandit. Exact refreshes idealize loss evaluation and velocity estimation; errors from noisy evaluations or a different time convention for irregularly sampled OLS slopes require an additional feedback-error term.

Assumption A.3 (Supervision confidence calibration)**.** Write $f_{k,m} = f_{\phi_k}(\mathbf{x}_k; m)$ for the prediction before the current outcome, $g_{k,m} \in \mathbb{R}^d$ for its gradient, and assume $\|g_{k,m}\|_2 \le L$ and $\alpha \ge 0$. Here d is the dimension of the gradient representation used by the uncertainty matrix; for gradients with respect to all neural parameters, $d = \dim(\phi)$, not the input-context dimension. Let $\Lambda_1 = \delta I_d$ with $\delta > 0$, and use either the full update in the main text or its diagonal approximation:

$$\Lambda_{k+1} = \lambda \Lambda_k + H_k, \qquad H_k = g_{k,m_k} g_{k,m_k}^\top \text{ or } H_k = \operatorname{diag}(g_{k,m_k} g_{k,m_k}^\top). \tag{15}$$

Here $0 < \lambda \le 1$; $\lambda = 1$ is the no-forgetting special case. Define $s_{k,m} = \sqrt{g_{k,m}^\top \Lambda_k^{-1} g_{k,m}}$, $F_{k,m} = (f_{k,m} + \psi)/(1 + \psi)$, and $\bar{\alpha} = \alpha/(1 + \psi)$. For a chosen failure probability $\varepsilon \in (0, 1)$, assume there are deterministic $b_k \ge 0$ such that, with probability at least $1 - \varepsilon$, simultaneously for every k, m,

$$|\mu_{k,m} - F_{k,m}| \le \bar{\alpha} s_{k,m} + b_k. \tag{16}$$

Let $B_K = \sum_{k=1}^K b_k$.

This is an explicit additional assumption on prediction and uncertainty calibration, not a consequence of bounded rewards or smooth drift alone. The residual b_k allows utility drift, neural approximation, optimization, replay, and covariance-approximation errors beyond the nominal confidence width. The theorem below applies to the implemented neural scores only when Eq. (16) holds for those scores and widths. The original NeuralUCB result Zhou et al. (2020) does not by itself establish this condition for discounted replay with a diagonal matrix. The affine normalization preserves expert selection: maximizing $F_{k,m} + \bar{\alpha} s_{k,m}$ is equivalent to Eq. (7). The dimension-dependent bound below can consequently be vacuous for a large network; no reduction to a neural effective dimension is established here.

A.3 Lazy-Feedback Error

Define a uniform feedback error, which controls both the chosen scenario and any comparator,

$$\Delta_{\text{lazy}}(K) = \sum_{k=1}^K \max_z |r_{k,z} - \hat{r}_{k,z}|. \tag{17}$$

Lemma A.4 (Cumulative lazy error)**.** *Under Assumptions A.1 and A.2,*

$$\mathbb{E}[\Delta_{\text{lazy}}(K)] \le D_K := \min\left\{K, \frac{(\log N + 2)V_r}{p_{\min}}\right\}. \tag{18}$$

In particular, if $\rho_s \le \rho$, then $D_K \le \min\{K, (\log N + 2)\rho(K-1)/p_{\min}\}$.

Proof. Telescoping gives

$$|r_{k,z} - \hat{r}_{k,z}| \le \sum_{s=1}^{k-1} \rho_s \mathbf{1}\{t_k(z) \le s\}.$$

For $s < k$, the event $t_k(z) \le s$ requires no refresh of z in episodes $s+1, \dots, k-1$. Iterating the conditional bound in Eq. (14) yields a probability at most $(1-p_{\min})^{k-1-s}$; independence across episodes is unnecessary. A union bound over scenarios therefore gives

$$\mathbb{E}\Big[\max_z |r_{k,z} - \hat{r}_{k,z}|\Big] \le \sum_{s=1}^{k-1} \rho_s \min\{1, N(1-p_{\min})^{k-1-s}\}.$$

With the zeroth power interpreted as one, for $0 < p \le 1$,

$$\sum_{\ell=0}^{\infty} \min\{1, N(1-p)^\ell\} \le 1 + \int_0^\infty \min\{1, Ne^{-px}\}\, dx = 1 + \frac{\log N + 1}{p} \le \frac{\log N + 2}{p}.$$

Summing over k and interchanging the finite sums proves the variation-dependent bound. The bound K follows from $r_{k,z}, \hat{r}_{k,z} \in [0,1]$. Using a maximum over scenarios avoids applying a fixed-scenario age bound to the adaptively chosen z_k. □

A.4 Curriculum Regret

Specify the Exp3 update on normalized feedback as

$$\begin{aligned} q_{k,z} &= \frac{w_{k,z}}{\sum_{z'} w_{k,z'}}, \qquad p_{k,z} = (1-\gamma)q_{k,z} + \frac{\gamma}{N}, \\ \widetilde{r}_{k,z} &= \frac{\hat{r}_{k,z}\mathbf{1}\{z_k = z\}}{p_{k,z}}, \qquad w_{k+1,z} = w_{k,z}\exp(\eta \widetilde{r}_{k,z}), \end{aligned} \tag{19}$$

where $w_{1,z} = 1$, $0 < \gamma \le 1/2$, and $0 < \eta \le \gamma/N$.

Lemma A.5 (Exp3 with lazy feedback)**.** *Under Assumptions A.1 and A.2,*

$$R_{\text{cur}}(K) \le A_K + 2D_K, \qquad A_K := \min\left\{K, \frac{\log N}{\eta} + \frac{(\mathrm{e}-2)\eta N K}{1-\gamma} + \gamma K\right\}. \tag{20}$$

Choosing $\gamma = \min\{1/2, \sqrt{N\log N/K}\}$ and $\eta = \gamma/N$ gives $A_K = \mathcal{O}(\sqrt{KN\log N})$.

Proof. We give the standard exponential-weights argument Auer et al. (2002) with consistent scaling. Let $W_k = \sum_z w_{k,z}$. Since $0 \le \eta\widetilde{r}_{k,z} \le \eta N/\gamma \le 1$, the inequality $e^x \le 1 + x + (\mathrm{e}-2)x^2$ for $x \in [0,1]$ implies

$$\log\frac{W_{k+1}}{W_k} \le \eta \sum_z q_{k,z}\widetilde{r}_{k,z} + (\mathrm{e}-2)\eta^2 \sum_z q_{k,z}\widetilde{r}_{k,z}^2.$$

For any fixed scenario z, the lower bound $\log(W_{K+1}/W_1) \ge \eta\sum_k \widetilde{r}_{k,z} - \log N$ then gives

$$\sum_k \widetilde{r}_{k,z} - \sum_k \sum_{z'} q_{k,z'}\widetilde{r}_{k,z'} \le \frac{\log N}{\eta} + (\mathrm{e}-2)\eta \sum_k \sum_{z'} q_{k,z'}\widetilde{r}_{k,z'}^2.$$

Predictability ensures conditional unbiasedness for the cached pre-decision feedback, and

$$\mathbb{E}[\widetilde{r}_{k,z} \mid \mathcal{F}_{k-1}] = \hat{r}_{k,z}, \qquad \mathbb{E}\left[\sum_z q_{k,z}\widetilde{r}_{k,z}^2 \mid \mathcal{F}_{k-1}\right] = \sum_z \frac{q_{k,z}\hat{r}_{k,z}^2}{p_{k,z}} \le \frac{N}{1-\gamma}.$$

This identity would not identify a post-training reward with $r_{k,z}$: for an actual normalized feedback X_k supplied after choosing z_k,

$$\mathbb{E}\left[\frac{X_k \mathbf{1}\{z_k = z\}}{p_{k,z}} \,\middle|\, \mathcal{F}_{k-1}\right] = \mathbb{E}[X_k \mid \mathcal{F}_{k-1}, z_k = z],$$

which need not equal either $r_{k,z}$ or the cache $\hat{r}_{k,z}$. Changing from q_k to p_k costs at most γ per episode. Taking expectations and then maximizing over fixed z bounds the expected external regret on $\hat{r}$ by the nontrivial expression in A_K; bounded rewards also give the bound K. For every fixed z, pathwise,

$$\sum_k (r_{k,z} - r_{k,z_k}) \le \sum_k (\hat{r}_{k,z} - \hat{r}_{k,z_k}) + 2\Delta_{\text{lazy}}(K).$$

Taking expectations and the maximum proves Eq. (20). The stated parameter choice gives the square-root rate when $K \ge 4N \log N$; otherwise the trivial bound K gives the same order. □

A.5 Supervision Regret with Discounting

The next bound uses the actual matrix recursion in Eq. (15); in particular, its initial regularizer is discounted over time. Define

$$h_K = \begin{cases} \min\{K, (1-\lambda)^{-1}\}, & \lambda < 1, \\ K, & \lambda = 1, \end{cases} \qquad Q_K = d \log\left(1 + \frac{L^2 h_K}{\delta d}\right) + Kd \log \frac{1}{\lambda}. \tag{21}$$

Lemma A.6 (Discounted sum of confidence widths)**.** *Under the matrix and feature conditions in Assumption A.3, for either the full or diagonal update,*

$$\sum_{k=1}^{K} \min\{1, s_{k,m_k}\} \le \sqrt{2KQ_K}. \tag{22}$$

Proof. Write $s_k = s_{k,m_k}$. For a full matrix, the determinant identity gives

$$\frac{\det \Lambda_{k+1}}{\lambda^d \det \Lambda_k} = 1 + \frac{s_k^2}{\lambda} \ge 1 + s_k^2.$$

For a diagonal matrix, the same ratio is $\prod_{i=1}^{d}(1 + g_{k,m_k,i}^2/(\lambda \Lambda_{k,ii})) \ge 1 + s_k^2/\lambda$, so the same inequality holds. Since $\min\{1, x\} \le 2\log(1+x)$ for $x \ge 0$,

$$\begin{aligned} \sum_{k=1}^{K} \min\{1, s_k^2\} &\le 2\left(\log \frac{\det \Lambda_{K+1}}{\det \Lambda_1} + Kd \log \frac{1}{\lambda}\right) \\ &\le 2\left(d \log\left(1 + \frac{L^2 h_K}{\delta d}\right) + Kd \log \frac{1}{\lambda}\right) = 2Q_K. \end{aligned}$$

The second inequality follows from $\operatorname{tr} \Lambda_{K+1} \le d\delta + L^2 \sum_{j=0}^{K-1} \lambda^j \le d\delta + L^2 h_K$ and the arithmetic–geometric mean inequality for eigenvalues. Cauchy–Schwarz proves the result. The term $Kd \log(1/\lambda)$ accounts for forgetting and cannot be omitted. □

Lemma A.7 (Conditional supervision bound)**.** *Under Assumption A.3, the UCB rule in Eq.* (7) *satisfies*

$$R_{\text{sup}}(K) \le \min\left\{K, c_\alpha \sqrt{2KQ_K} + 2B_K + \varepsilon K\right\}, \qquad c_\alpha = \max\{1, 2\bar{\alpha}\}. \tag{23}$$

Proof. On the simultaneous confidence event, let $m_k^\star \in \arg\max_m \mu_{k,m}$. The UCB selection rule implies

$$\begin{aligned} \mu_{k,m_k^\star} &\le F_{k,m_k^\star} + \bar{\alpha} s_{k,m_k^\star} + b_k \le F_{k,m_k} + \bar{\alpha} s_{k,m_k} + b_k, \\ \mu_{k,m_k} &\ge F_{k,m_k} - \bar{\alpha} s_{k,m_k} - b_k. \end{aligned}$$

Thus the instantaneous pseudo-regret is at most $2\bar{\alpha} s_{k,m_k} + 2b_k$, and also at most one. Consequently it is bounded by $c_\alpha \min\{1, s_{k,m_k}\} + 2b_k$. Summing and applying Lemma A.6 proves the bound on the confidence event. On its complement the sum is at most K, contributing at most εK in expectation. No conditioning on future contexts or scenarios is needed. □

A sufficient special case for confidence calibration. The confidence assumption has a familiar sufficient condition in an idealized stationary linear model. Suppose $\mathbb{E}[U_{k,m} \mid \mathcal{G}_k] = g_{k,m}^\top \theta^\star$, $\|\theta^\star\|_2 \le S$, and the features use a fixed representation. The selected-action noise $\xi_k = U_{k,m_k} - g_{k,m_k}^\top \theta^\star$ must be conditionally centered and satisfy

$$\mathbb{E}[\exp(t\xi_k) \mid \mathcal{G}_k, m_k] \le \exp(\sigma^2 t^2/2), \qquad t \in \mathbb{R}.$$

With $\lambda = 1$, a full matrix, and the exact ridge predictor

$$f_{k,m} = g_{k,m}^\top \hat{\theta}_k, \qquad \hat{\theta}_k = \Lambda_k^{-1} \sum_{j<k} g_{j,m_j} U_{j,m_j},$$

the self-normalized confidence inequality Abbasi-Yadkori et al. (2011) gives, simultaneously over all episodes and available experts,

$$|g_{k,m}^\top (\hat{\theta}_k - \theta^\star)| \le \left[\sigma \sqrt{2\log(1/\varepsilon) + d \log\left(1 + \frac{KL^2}{\delta d}\right)} + \sqrt{\delta} S \right] s_{k,m}$$

with probability at least $1 - \varepsilon$. Choosing α at least as large as the bracketed term establishes Assumption A.3 with $b_k = 0$. This example permits adaptive contexts and shows a setting where the assumption follows from a statistical model. It does not identify the trained neural network with a ridge estimator or establish calibration for diagonal, discounted replay; those cases retain the explicit assumption and residual B_K.

A.6 Combined Bound and Its Interpretation

Theorem A.8 (Conditional composite regret bound for COACH)**.** *Under Assumptions A.1–A.3, with the normalized, pre-decision-feedback Exp3 update in Eq.* (19) *and the UCB rule in Eq.* (7)*,*

$$\boxed{R_{\mathrm{COACH}}(K) \le A_K + 2D_K + c_\alpha \sqrt{2KQ_K} + 2B_K + \varepsilon K.} \tag{24}$$

More explicitly, the bound can be capped componentwise as

$$R_{\mathrm{COACH}}(K) \le \min\{K, A_K + 2D_K\} + \min\{K, c_\alpha \sqrt{2KQ_K} + 2B_K + \varepsilon K\} \le 2K.$$

Here A_K, D_K, and Q_K are defined in Eqs. (20)*,* (18)*, and* (21)*.*

Proof. Add Lemmas A.5 and A.7 using the definition in Eq. (12). All expectations are over the same adaptive interaction history; independence between the modules is not required. □

Sublinear learning and cumulative residuals. Equation (24) separates learning from the cumulative residuals associated with lazy evaluation and online adaptation. Using $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$ and the definition of Q_K, we can write

$$R_{\mathrm{COACH}}(K) \le A_K + c_\alpha \sqrt{2Kd \log\left(1 + \frac{L^2 h_K}{\delta d}\right)} + \mathcal{E}_K, \tag{25}$$

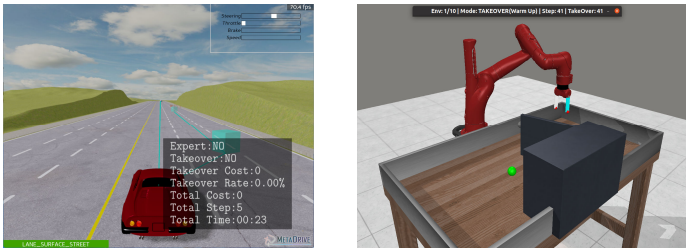
$$\mathcal{E}_K := 2D_K + c_\alpha K \sqrt{2d \log(1/\lambda)} + 2B_K + \varepsilon K. \tag{26}$$

With the Exp3 tuning in Lemma A.5, fixed N, d, L, δ, and $c_\alpha = \mathcal{O}(\sqrt{\log K})$, the first two terms in Eq. (25) are sublinear. The term $\mathcal{E}_K$ collects the residual costs of lazy evaluation and online adaptation. For fixed c_α, λ, and ε, bounded per-episode drift and calibration residuals give $\mathcal{E}_K = \mathcal{O}(K)$, so the bound consists of a sublinear learning term plus a potentially linear residual term. Dividing by K gives a vanishing learning contribution plus the average residual $\mathcal{E}_K/K$. If instead c_α grows with K, its dependence must also be retained in the residual: for example, the uncapped forgetting term can scale as $K\sqrt{\log K}$ when $\lambda < 1$ is fixed and c_α scales as $\sqrt{\log K}$. The componentwise caps keep the overall upper bound at most $2K$, but do not make it sublinear.

Conditions for an overall sublinear bound. With the Exp3 tuning in Lemma A.5, Eq. (24) yields a sublinear upper bound if

$$N \log N = o(K), \qquad D_K = o(K), \qquad B_K = o(K), \qquad c_\alpha^2 Q_K = o(K), \qquad \varepsilon = o(1). \tag{27}$$

Parameters may depend on K, with Assumption A.3 required to hold for each choice.



(a) MetaDrive *(b)* MetaWorld

Figure 4: MetaDrive and MetaWorld

B Experimental Setup and Protocols

Environment Details. We conduct our experiments on two challenging simulation environments: MetaDrive Li et al. (2022a) and MetaWorld Yu et al. (2020).

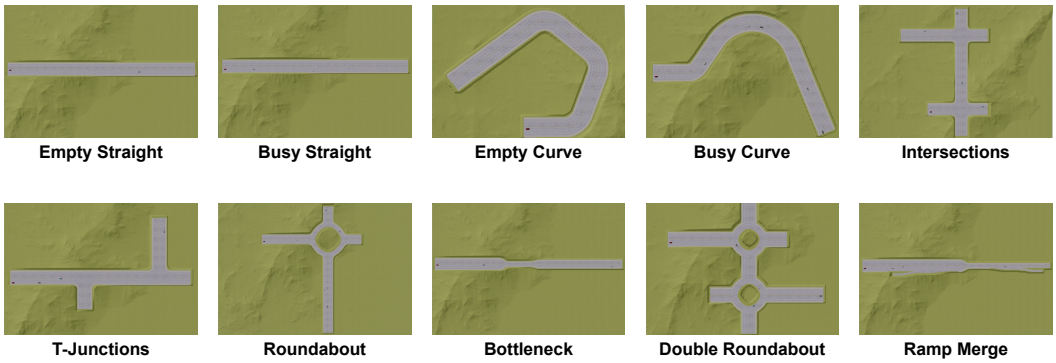


Figure 5: Visualizations of 10 driving scenarios in MetaDrive

MetaDrive offers a diverse array of simulation environments for autonomous driving. For the training phase, we curated a suite of 10 distinct scenarios designed to encompass the fundamental characteristics of the MetaDrive simulator. Specifically, these scenarios vary significantly in terms of road topologies—ranging from simple straight roads and continuous curves to complex intersections, roundabouts, bottlenecks, and highway ramps—as well as varying levels of traffic density, as illustrated in Fig. 5. During the Human-in-the-Loop (HIL) training sessions within MetaDrive, human participants supervise the learning agent and intervene via a keyboard interface whenever they detect potential traffic violations, safety breaches, or actions misaligned with human intent. Each HIL session lasts approximately 60 minutes, consisting of roughly 30,000 agent-environment interaction steps. In the testing phase, we test on a separate set of 50 scenarios based on these primitives, featuring varied origin-destination settings and traffic loads, and without any expert intervention. performance is primarily evaluated using episodic return, safety cost, and success rate. The episodic return is calculated following the default configuration detailed in PVP Peng et al. (2023). Similarly, the safety cost adheres to the default settings, where a penalty of +1 is added to the episodic cost for every timestep that a collision (e.g., with other vehicles, obstacles, or sidewalks) persists.

MetaWorld serves as a comprehensive benchmark for robotic manipulation. In this study, we utilize 10 representative environments that involve interactions with diverse mechanical components, including doors, drawers, windows, buttons, handles, and levers, as shown in Fig. 6. From a control perspective, these tasks are constructed around combinations of three atomic action primitives: reaching, pushing, and grasping. During the Human-in-the-Loop (HIL) training sessions, human participants continuously supervise the learning agent and intervene via a keyboard interface whenever they detect actions misaligned with human intent. Each HIL session consists of approximately 15,000 agent-environment interaction steps, lasting around 30 minutes. In testing, we evaluate on a separate

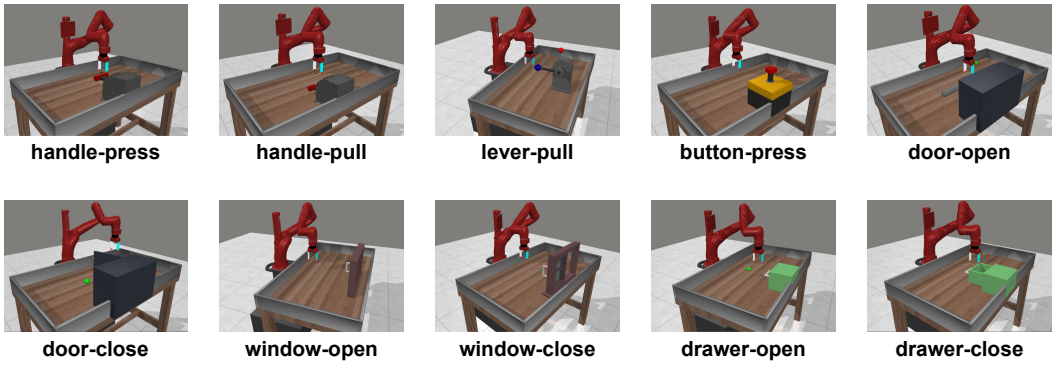


Figure 6: Visualizations of 10 tasks in the MetaWorld

set of 50 tasks based on these primitives but initialized with different arm states and object locations, and without any expert intervention. We evaluate the performance using cumulative episodic return and success rate, where the success rate indicates the proportion of episodes in which the agent successfully manipulates the object to the target state.

Expert Pool. 1) **Synthetic Expert Pool**. We trained three Reinforcement Learning (RL) agents as synthetic experts to isolate algorithmic performance from human variance (e.g., fatigue, attention drift) and guarantee fair comparison under standardized conditions. To ensure a clear proficiency gradient, the RL agents were trained on scenarios of varying difficulty. We validate the capability of the RL agents on a testing set, and confirmed distinct capability levels: In MetaDrive, the agents achieved average success rates of 12.0% ($\pm$4.6, Junior), 44.0% ($\pm$7.0, Intermediate), and 94.0% ($\pm$3.3, Senior). Similarly, in MetaWorld, the success rates were 11.0% ($\pm$4.4), 45.0% ($\pm$7.1), and 97.0% ($\pm$2.4), respectively. The costs of experts are set to 1, 5, and 10 for junior, intermediate and senior experts, respectively. 2) **Human Participants**. We recruited nine college students (volunteers) to participate in our human-in-the-loop (HIL) experiments. To ensure a strictly controlled baseline for skill acquisition, we selected participants with no prior experience in motor vehicle driving or robotic arm manipulation, aligning with the requirements of our MetaDrive and MetaWorld environments. All subjects were fully briefed on the research objectives, the collaborative nature of the tasks, and data usage policies. Written informed consent was obtained from all participants prior to the study. The protocol was reviewed and approved by the university Institutional Review Board (IRB), ensuring compliance with ethical standards for human-subject research.

HIL Experiment Details. The experiments were designed to evaluate task allocation among experts of varying proficiency. Prior to formal testing, participants underwent targeted training to establish three distinct tiers—junior, intermediate, and senior—with training durations of 5, 20, and 60 minutes, respectively. Post-training validation confirmed distinct capability levels: In MetaDrive, the groups achieved average success rates of 6.0% ($\pm$3.4, Junior), 62.0% ($\pm$6.9, Intermediate), and 88.0% ($\pm$4.6, Senior). Similarly, in MetaWorld, the success rates were 23.0% ($\pm$5.9), 53.0% ($\pm$7.1), and 95.0% ($\pm$3.1), respectively. Subsequently, participants were organized into three-member experimental groups, each containing one expert from each proficiency level. For each episode, the evaluated algorithm selected one expert from the pool; only the selected expert monitored the learner and intervened when necessary, while the other two did not participate in that episode. To prevent order bias, the presentation sequence of the nine algorithms and task scenarios was randomized. For MetaDrive, each group dedicated a total of 9 hours to the experiments (covering 9 different algorithms with approximately 1 hour per session). For MetaWorld, the total commitment was 4.5 hours (covering 9 algorithms with approximately 30 minutes per session). This setup allows for a comprehensive analysis of how different algorithms manage expert assignment.

C ADDITIONAL EXPERIMENTAL ANALYSES

C.1 EXPERT ALLOCATION ANALYSIS

To validate the effectiveness of the Supervision Bandit module in dynamically allocating appropriate supervision, we visualize the evolution of expert sampling ratios throughout the training process. Fig. 7 and Fig. 8 illustrate the selection probabilities of Junior (Blue), Intermediate (Orange), and Senior (Green) experts across 10 distinct scenarios in MetaDrive and MetaWorld, respectively. For visualization of allocation dynamics, we additionally extend training to 40k steps in MetaDrive and 20k steps in MetaWorld; the quantitative comparisons in Tables 1–2 use the fixed 30k/15k budgets.

Adaptive Expert Transition. A consistent trend is observed across both domains: in the initial training phase, the agent heavily relies on Senior and Intermediate experts (indicated by the large green and orange areas). This behavior is expected, as the agent's policy is initially untrained, necessitating high-quality interventions to ensure safety and effective learning. As the number of training steps increases and the agent's proficiency improves, the Supervision Bandit adaptively shifts the workload towards the Junior expert. By the end of the training , the Junior expert dominates the selection in most tasks. This transition demonstrates COACH's ability to minimize supervision costs by avoiding over-qualified assistance once the agent becomes competent enough to learn from less skilled (and cheaper) supervisors.

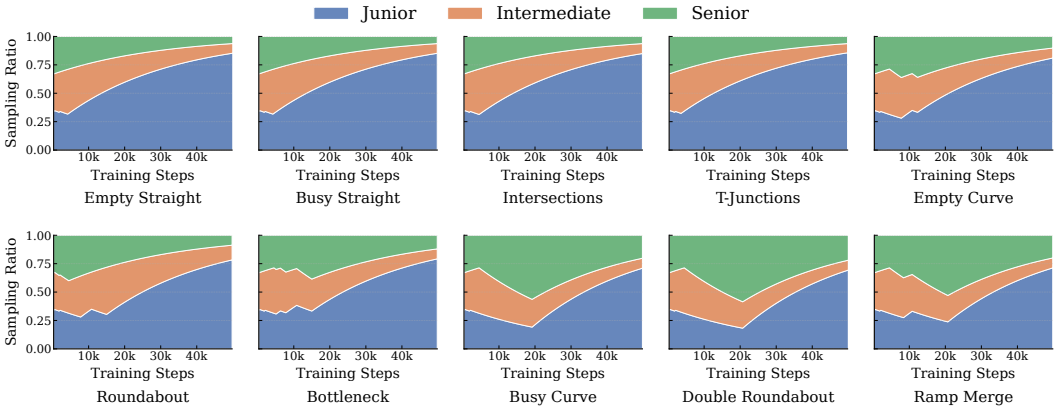


Figure 7: Dynamic Expert Allocation. Stacked plots illustrate the evolution of expert sampling ratios (calculated over a 300-step sliding window) across 10 Scenarios in MetaDrive.

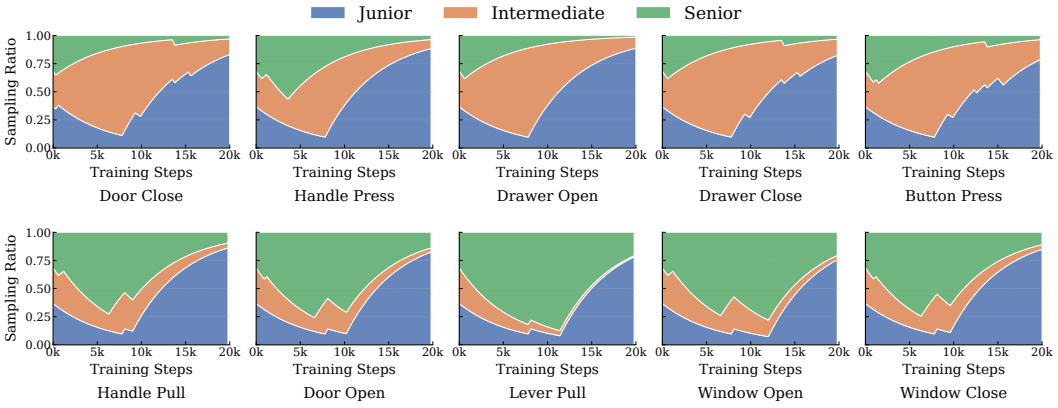


Figure 8: Dynamic Expert Allocation. Stacked plots illustrate the evolution of expert sampling ratios (calculated over a 300-step sliding window) across 10 Scenarios in MetaWorld.

Table 4: Expert allocation and intervention behavior with a mature learner in MetaDrive.

Expert	Assigned	Intervened	Int. Rate	Int./Episode	Success
Junior	38	16	42.11%	1.24 ± 2.16	81.25%
Senior	12	6	50.00%	4.00 ± 6.34	100.00%

Table 5: Comparison of curriculum signals in MetaDrive.

Curriculum Signal	Episodic Return	Success Rate
Loss Velocity	**360.5**	**92.0%**
Policy Entropy Change	348.5	86.0%
Success-Rate Improvement	355.4	82.0%

Task-Aware Allocation. The results also highlight the context-awareness of our method. In simpler scenarios, the transition to Junior experts occurs rapidly. For instance, in the *Empty Straight* task of MetaDrive (Fig. 7) and the *Door Close* task of MetaWorld (Fig. 8), the Junior expert's ratio rises sharply early in training. Conversely, in highly complex environments requiring precise manipulation or navigation—such as *Double Roundabout* and *Ramp Merge* in MetaDrive, or *Window Open/Close* in MetaWorld—the system retains Senior and Intermediate experts for a significantly longer duration. This delayed transition confirms that COACH does not follow a fixed schedule but intelligently assesses the difficulty of the context $\mathbf{x}_k$ relative to the experts' capabilities, ensuring that cost reduction does not come at the expense of safety or learning stability.

Action Consistency of Junior Interventions. To examine the quality of lower-tier demonstrations routed by COACH, we compare Junior and Senior expert actions at the same intervention states in MetaDrive. For each state s at which the Junior expert intervenes, we query the Senior expert offline to obtain a reference action a_{Sen} and compute the mean squared error between a_{Jun} and a_{Sen}. Across the collected Junior intervention states, the action MSE has a mean of 0.001438, a standard deviation of 0.004733, and a maximum of 0.078763. Given the continuous action range of $[-1, 1]$, these results indicate that the Junior interventions actually selected by COACH are highly consistent with the corresponding Senior actions.

Supervision Behavior with a Mature Learner. We further analyze why lower-cost experts remain useful after the learner becomes proficient. We initialize COACH from a mature learner checkpoint and consider an expert pool containing only Junior and Senior experts. Over 50 MetaDrive episodes, we record expert assignments, intervention frequency, task outcomes, and intervention contexts. The results are summarized in Table 4.

Among the 38 episodes assigned to the Junior expert, 22 require monitoring only, without intervention. When intervention is needed, the Junior expert also provides substantially fewer corrections than the Senior expert. Their intervention contexts further differ: Junior interventions are concentrated on routine local corrections, particularly curve deviation (48.9%), whereas Senior interventions are dominated by collision avoidance (66.7%). The corresponding safety costs of intervened episodes are 5.38 ± 9.49 for the Junior expert and 1.67 ± 3.73 for the Senior expert. These results suggest that, once the learner becomes mature, Junior experts mainly provide low-cost monitoring and occasional routine corrections, while Senior supervision remains important for more difficult or high-risk residual errors.

C.2 Curriculum Signal and Lazy Re-evaluation

Alternative Curriculum Signals. We compare loss velocity with two alternative progress signals, policy-entropy change and success-rate improvement, in MetaDrive. As shown in Table 5, loss velocity achieves the best overall performance among the evaluated signals. Compared with policy entropy, it more directly reflects changes in the imitation objective, while success-rate improvement provides a sparser progress signal that requires additional task-level evaluation.

Table 6: Sensitivity to the Lazy Re-evaluation budget N_{eval}.

Environment	N_{eval}	Return	Success Rate
MetaDrive	1	350.8	82.0%
	4	361.2	88.0%
	5	360.5	92.0%
	6	363.6	92.0%
MetaWorld	1	2029.79	62.0%
	4	2053.70	78.0%
	5	2592.85	92.0%
	6	2576.64	92.0%

Lazy Re-evaluation Budget. We further analyze the re-evaluation budget N_{eval}, which controls how many scenario-level curriculum estimates are refreshed at each update. As shown in Table 6, a very small evaluation budget can leave stale curriculum estimates and degrade performance. Increasing N_{eval} generally improves performance, while the gains largely saturate around $N_{\text{eval}} = 5$ in both domains. These results demonstrate the expected trade-off between estimate freshness and re-evaluation overhead.

D Ethics Statement

The experimental protocols and data collection procedures were formally approved by the university's Institutional Review Board. Prior to the study, all participants provided written informed consent following a comprehensive briefing on the research objectives, interactive tasks, and data protocols, with the explicit understanding that participation was voluntary and they could withdraw at any time without penalty. To ensure confidentiality and security, all collected data were fully anonymized to exclude personally identifiable information and are stored on secure, encrypted university servers accessible only to the core research team, in strict adherence to institutional data governance policies.

E Implementation Details and Hyperparameters

This section provides implementation details, network architectures, curriculum-loss definitions, and hyperparameter configurations.

E.1 Hyperparameter Sensitivity

We evaluate the sensitivity of the key temporal and cost-sensitive hyperparameters: the learning-velocity window τ, cost coefficient κ, proficiency window T, forgetting factor λ, and failure penalty ψ. The selected configurations correspond to the settings used in the main experiments. Across both domains, performance is generally stable in a neighborhood of the selected values, while overly short/long temporal windows or excessive retention of stale supervision feedback can reduce return or success. The cost-aware parameters also exhibit the intended trade-off: settings with comparable success may differ in expert expenditure, so we select values that balance task performance and supervision cost rather than maximizing return alone.

E.2 Network Architecture.

For MetaDrive, the policy network is a Multi-Layer Perceptron (MLP). For MetaWorld, we utilize a Diffusion Policy based on a 1D Conditional U-Net to model the complex multi-modal distribution of robotic manipulation tasks. The specific architectural details are compared in Table 9 and Table 10.

E.3 Implementation of the Curriculum Loss

Both environments evaluate $\mathcal{L}_{k,z}$ as the action-space imitation error with the current policy π_{θ_k} in a gradient-free forward pass.

Table 7: Hyperparameter sensitivity on MetaDrive.

Hyperparameter	Value	Episodic Return	Success Rate
τ	3	345.9	86.0%
	5	360.5	92.0%
	8	358.6	92.0%
κ	0.04	354.7	88.0%
	0.05	360.5	92.0%
	0.06	366.4	92.0%
T	10	356.8	88.0%
	20	360.5	92.0%
	40	350.4	84.0%
λ	0.85	358.7	92.0%
	0.90	360.5	92.0%
	0.95	343.4	90.0%
ψ	0.5	358.3	90.0%
	1	360.5	92.0%
	2	365.4	92.0%

Table 8: Hyperparameter sensitivity on MetaWorld.

Hyperparameter	Value	Episodic Return	Success Rate
τ	3	1792.91	64.0%
	5	2592.98	92.0%
	8	2640.11	92.0%
κ	0.04	2348.10	76.0%
	0.05	2592.98	92.0%
	0.06	2387.33	88.0%
T	5	2390.19	90.0%
	10	2592.98	92.0%
	20	2199.94	88.0%
λ	0.80	2483.42	92.0%
	0.85	2592.98	92.0%
	0.90	2340.17	90.0%
ψ	0.5	2524.87	92.0%
	1	2592.98	92.0%
	2	2537.43	92.0%

Table 9: Network Architecture for MetaDrive (BC Policy)

Component	**Specification**
Policy Type	MLP
Hidden Layers	2
Hidden Units	512
Activation Function	ReLU
Optimizer	Adam
Loss Function	MSE Loss

Table 10: Network Architecture for MetaWorld (Diffusion Policy)

Component	**Specification**
Policy Type	1D Conditional U-Net
Noise Scheduler	DDPM
Diffusion Steps	30
Input Horizons	$T_{obs} = 4, T_{pred} = 4$
Down-sampling Channels	$[64, 128, 256]$
Kernel Size	5
Group Norm	8
Optimizer	AdamW

MetaDrive. The MLP policy outputs a deterministic action $\mu_{\theta_k}(s)$. Under its fixed-variance Gaussian surrogate $\pi(a|s) = \mathcal{N}(\mu_{\theta_k}(s), \sigma^2 I)$, the NLL in Eq. (2) reduces to $\frac{1}{2\sigma^2}\|a - \mu_{\theta_k}(s)\|_2^2 + \text{const.}$, so the implemented loss

$$\mathcal{L}_{k,z}^{\text{MD}} = \frac{1}{B_z} \sum_{(s,a)\in\mathcal{B}_z} \|a - \mu_{\theta_k}(s)\|_2^2 \tag{28}$$

equals the NLL(Negative Log-Likelihood) up to a scenario-independent affine map: the OLS slope of Eq. (3) cancels the offset, and the constant scale is absorbed by the Exp3 reward normalization (Appendix A), leaving the induced curriculum unchanged. $\mathcal{B}_z$ contains the $B_z = 256$ most recent expert intervention pairs of scenario z, aggregated by the mean over batch and action dimensions.

MetaWorld. The NLL of the Diffusion Policy Chi et al. (2025) is intractable. The policy is trained with the standard denoising objective

$$\mathcal{L}_{\text{denoise}} = \mathbb{E}_{(s,\mathbf{a})\sim\mathcal{D},\, t\sim\mathcal{U}\{1,\dots,T\},\, \boldsymbol{\epsilon}\sim\mathcal{N}(0,I)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\sqrt{\bar{\alpha}_t}\,\mathbf{a} + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon},\, s,\, t\big) \right\|_2^2, \tag{29}$$

where $\boldsymbol{\epsilon}_\theta$ is the U-Net noise predictor, $T = 30$, and $\bar{\alpha}_t$ follows the linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. The curriculum, however, does not reuse Eq. (29): it evaluates the reconstruction error of full diffusion inference (T reverse steps from fresh Gaussian noise, conditioned on the $T_{\text{obs}} = 4$ observation history, clamped to $[-1, 1]$),

$$\mathcal{L}_{k,z}^{\text{MW}} = \frac{1}{B_z} \sum_{(s,\mathbf{a})\in\mathcal{B}_z} \|\hat{\mathbf{a}}_{\theta_k}(s) - \mathbf{a}\|_2^2, \tag{30}$$

where $\hat{\mathbf{a}}_{\theta_k}(s) \in \mathbb{R}^{T_{\text{pred}}\times d_a}$ is the generated action chunk and $\mathbf{a}$ the expert chunk. $\mathcal{B}_z$ contains $B_z = 64$ (observation, action-chunk) pairs sampled uniformly from the intervention episodes of scenario z (start indices restricted to expert-action positions; boundary windows padded by repeating the first observation and the last action), averaged over batch, horizon, and action dimensions; scenarios without data default to 1. The sampling noise of $\hat{\mathbf{a}}_{\theta_k}(s)$ is dampened by the minibatch mean and the OLS slope over the τ cached evaluations; losses are recomputed only for scenarios refreshed by the re-evaluation mechanism.

With these instantiations, $\mathcal{L}_{k,z}$, $\nu_{k,z}$ (Eq. (3)), and $r_{k,z}$ (Eq. (4)) are fully determined for both environments.

E.4 Full Hyperparameter Configurations

We list Hyperparameter Configurations for compared methods.

Table 11: COACH (MetaDrive)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	2
Gradient Steps per Iteration	1
Curriculum Exploration γ	0.2
Re-eval Exploration γ_{eval}	0.2
Velocity Window Size τ	5
UCB Confidence α	0.5
Cost Sensitivity κ	0.05
Bandit Hidden Units	32
Bandit Latent Units	16
Bandit Learning Rate	0.001
Proficiency Window T	20
Forgetting Factor λ	0.9
Failure penalty ψ	1.0

Table 12: COACH (MetaWorld)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	128
Steps per Iteration	2
Gradient Steps per Iteration	1
Curriculum Exploration γ	0.2
Re-eval Exploration γ_{eval}	0.2
Velocity Window Size τ	5
UCB Confidence α	0.3
Cost Sensitivity κ	0.05
Bandit Hidden Units	32
Bandit Latent Units	16
Bandit Learning Rate	0.001
Proficiency Window T	10
Forgetting Factor λ	0.85
Failure penalty ψ	1.0

Table 13: DQM (MetaDrive)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	2
Gradient Steps per Iteration	1
MAB Scaling Factor γ	0.01
Success Smoothing α	0.2
Initial Belief h_i	0.1

Table 14: DQM (MetaWorld)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	128
Steps per Iteration	2
Gradient Steps per Iteration	1
MAB Scaling Factor γ	0.01
Success Smoothing α	0.2
Initial Belief h_i	0.1

Table 15: RgC (MetaDrive)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	2
Gradient Steps per Iteration	1
MAB Weight Scaling γ	0.01
History Rolling Window (M)	50
Lower Percentile Threshold	0.2
Upper Percentile Threshold	0.8

Table 16: RgC (MetaWorld)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	128
Steps per Iteration	2
Gradient Steps per Iteration	1
MAB Weight Scaling γ	0.01
History Rolling Window (M)	50
Lower Percentile Threshold	0.2
Upper Percentile Threshold	0.8

Table 17: SGIM (MetaDrive)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	2
Gradient Steps per Iteration	1
Exploration Rate ϵ	0.1
Progress Window Size	30
Min Samples for Progress	5

Table 18: SGIM (MetaWorld)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	128
Steps per Iteration	2
Gradient Steps per Iteration	1
Exploration Rate ϵ	0.1
Progress Window Size	30
Min Samples for Progress	5

Table 19: CEAL (MetaDrive)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	2
Gradient Steps per Iteration	1
Exploration Rate ϵ	0.1
EMA Smoothing α_{ema}	0.2
Softmax Temperature τ	0.2
Cost Exponent α_{cost}	0.1
Uncertainty Window	30

Table 20: CEAL (MetaWorld)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	128
Steps per Iteration	2
Gradient Steps per Iteration	1
Exploration Rate ϵ	0.1
EMA Smoothing α_{ema}	0.2
Softmax Temperature τ	0.2
Cost Exponent α_{cost}	0.1
Uncertainty Window	30

Table 21: Joint MAB (MetaDrive)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	2
Gradient Steps per Iteration	1
Exp3 Exploration γ	0.1
Progress Window Size	30

Table 22: Joint MAB (MetaWorld)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	128
Steps per Iteration	2
Gradient Steps per Iteration	1
Exp3 Exploration γ	0.1
Progress Window Size	30

Table 23: SMT(MetaDrive)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	200
Steps per Iteration	2
Gradient Steps per Iteration	1
Difficulty Amplification β	2.0
Performance Window T	20
Solved Threshold M	0.8
Evaluation Interval	10
UCB Confidence α	0.5
Forgetting Factor λ	0.9
Cost Sensitivity κ	0.05
Failure penalty ψ	1.0

Table 24: SMT(MetaWorld)

Hyper-parameter	Value
Train Batch Size	128
Learning Rate	0.0001
Steps before Learning Start	128
Steps per Iteration	2
Gradient Steps per Iteration	1
Difficulty Amplification β	2.0
Performance Window T	10
Solved Threshold M	0.8
Evaluation Interval	10
UCB Confidence α	0.5
Forgetting Factor λ	0.85
Cost Sensitivity κ	0.05
Failure penalty ψ	1.0