FASTER POLICY LEARNING FROM MULTIPLE EXPERTS: STITCH, IMITATE, OPTIMIZE

Anonymous authors
Paper under double-blind review

ABSTRACT

How can we train a new policy from a set of expert policies, when none of the experts fully solve the task and none are available at deployment? We propose algorithms to learn to stitch the best experts across states, and to further improve upon the stitched experts. We jointly train two policies: a *student*, which will ultimately be deployed, and a *selector*, which at each state selects an expert or the student to act in the environment. The selector is updated with standard reinforcement learning (RL) to maximize returns. The student is updated with standard RL on its own actions, and with behavior cloning (BC) on actions taken by selected experts. Because the selector tends to select experts only if they outperform the student, the BC can automatically stop as the student outperforms the experts. Overall, this simple mechanism allows the student to combine the strengths of multiple experts, learn quickly from them in the early phase, and continue improving beyond them in the later phase, with the selector automatically guiding the phase transition. Empirically, we measure returns on all combinations of eight environments (four classic control environments, four higher-dimensional MuJoCo environments) with up to twelve expert sets. On all expert sets, averaging over the environments, our most robust algorithm's return CIs overlap or exceed both max-aggregation's (Cheng et al., 2020) and max$^+$-aggregation's (Liu et al., 2024), exceeding them by up to $5\times$.

1 INTRODUCTION

In many reinforcement learning settings, existing policies provide useful but incomplete solutions to a task. We refer to these policies as *experts*, even when none can solve the task on its own. Their strengths may be complementary across states. For example, in robotic manipulation, one expert may reliably grasp an object, while another can place it accurately once it is held. Similarly, a locomotion controller may perform well on level ground but struggle on stairs, where a different controller is more effective. These examples motivate learning to *stitch* useful behaviors from different experts, rather than committing to a single expert throughout an episode. Our goal is to train a single student policy that combines these strengths and can further improve from rewards.

Several prior works study imitation learning from multiple experts. Max-following (Cheng et al., 2020; Hussing et al., 2024) follows the expert with the highest estimated value at each state, while max-aggregation (Cheng et al., 2020) performs one-step policy improvement over the highest estimated value. Both approaches admit guarantees of learning a policy at least as good as the best *single* expert. However, this does not ensure *optimal stitching*. Specifically, the state value function used in these methods measures the return from continuing with one expert, rather than with switching experts. It can therefore underestimate the usefulness of actions whose benefit relies on subsequent switches to other experts, causing the methods to miss useful combinations of experts (Section B.1).

Extensions of max-aggregation incorporate reinforcement learning (Cheng et al., 2020; Liu et al., 2024), allowing the student to improve beyond the fixed experts. However, the ability to eventually outperform the experts does not mean that the experts' strengths are effectively exploited: when the student is still weak, these methods still suffer the stitching limitation described above.

A separate consideration is how expert knowledge is transferred to the student. The primary methods in Cheng et al. (2020) and Liu et al. (2024) rely on *value supervision*, in which the student searches

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.

over actions with the guidance of expert-derived values. This allows the student to discover better actions than those provided by the experts, making it particularly useful when the experts' actions are suboptimal. However, when the experts already provide useful actions, learning solely through value supervision may require unnecessary exploration to rediscover them. In contrast, *action supervision*, such as behavior cloning, directly transfers expert actions to the student, though the student might lack exploration over actions beyond experts' supervision. These complementary strengths motivate combining both forms of supervision: directly imitate useful expert behaviors while retaining value-guided exploration and RL for further improvement.

We propose Stitch-Imitate-Optimize (SIO), a method that unifies expert stitching, action supervision, value supervision, and reinforcement learning (RL). SIO trains a high-level selector over the experts and the student. At each state, the selector chooses either an expert or the current student to act, and is updated to maximize long-term rewards. The selector learns to stitch expert behaviors, and we show that the selector policy competes with the optimal statewise stitching of the experts. The student is updated in two complementary ways: behavior cloning (BC) on expert actions (when an expert is selected by the selector), and RL on the actions it takes itself (when the student is selected by the selector). BC provides action supervision, while the RL update incorporates value supervision through a critic shared with the selector. Through RL training, the selector uses return feedback to automatically adapt which experts the student imitates and the balance between BC and RL. These adaptations are statewise and evolve throughout training.

Empirically, we evaluate on four classic control environments, four higher-dimensional MuJoCo environments, and up to twelve expert-set types. Our expert-set types include random experts, good experts, good experts mixed with random experts, and several complementary-expert constructions. Across these combinations, our methods are strongest when the experts contain useful but incomplete behavior: the selector can exploit complementary expertise, while the student can continue learning from reward. We compare algorithms based on their returns normalized per environment. Averaging over the classic control environments or the higher-dimensional environments, our most robust algorithm's return CIs overlap or exceed both max-aggregation's (Cheng et al., 2020) and max$^+$-aggregation's (Liu et al., 2024) on all expert sets, exceeding them by up to $5\times$.

2 PRELIMINARIES

Notation. For a positive integer $N \in \mathbb{N}$, define $[N] = \{1, 2, \ldots, N\}$. We use Δ_X to denote the probability simplex over a set X, and write $\Delta_N = \Delta_{[N]}$.

Environment. We consider a discounted MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$, initial state distribution $d_0 \in \Delta_{\mathcal{S}}$, and discount factor $\gamma \in [0, 1)$. A policy π maps each state $s \in \mathcal{S}$ to a distribution $\pi(\cdot|s) \in \Delta_{\mathcal{A}}$. When π is deterministic, we write $\pi(s) \in \mathcal{A}$ for the action selected at state s. For a policy π, define its state-value and state-action-value functions by $V^\pi(s) = \mathbb{E}^{\pi,P}\left[\sum_{i=1}^{\infty} \gamma^{i-1} R(s_i, a_i) \,\middle|\, s_1 = s\right]$, $Q^\pi(s, a) = \mathbb{E}^{\pi,P}\left[\sum_{i=1}^{\infty} \gamma^{i-1} R(s_i, a_i) \,\middle|\, (s_1, a_1) = (s, a)\right]$. The advantage function is $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. The value functions for the optimal policy are denoted as $Q^\star$ and $V^\star$. We use the normalized discounted occupancy measure $d^\pi_\varrho(s) = (1-\gamma)\mathbb{E}^{\pi,P}\left[\sum_{i=1}^{\infty} \gamma^{i-1} \mathbf{1}[s_i = s] \,\middle|\, s_1 \sim \varrho\right]$, and write $d^\pi = d^\pi_{d_0}$. The normalized return is $J(\pi) = (1-\gamma)\mathbb{E}_{s\sim d_0}[V^\pi(s)]$.

Experts. We are given K expert policies $\pi^{(1)}, \ldots, \pi^{(K)}$. During training, the student can query expert k at state s to obtain an action sample from $\pi^{(k)}(\cdot|s)$, but it does not observe the full distribution $\pi^{(k)}(\cdot|s)$. The experts are not available at deployment time; only the student policy is deployed.

2.1 MAXFOLLOWING

The remaining subsections review existing approaches for this setting, starting with max-following, which we refer to as MaxFollowing from here on. MaxFollowing estimates the value function $V^{(k)} := V^{\pi^{(k)}}$ of each expert $k \in [K]$. It then follows the expert with the largest value at each state: let $k^\star(s) = \operatorname{argmax}_{k\in[K]} V^{(k)}(s)$ and set $\pi_{\mathrm{MF}}(s) = \pi^{(k^\star(s))}(s)$. This idea has been applied in autonomous driving (Li et al., 2018) and theoretically investigated in later work (Cheng et al., 2020;

Hussing et al., 2024; Sengupta et al., 2025). When the $V^{(k)}$ estimates are accurate, MaxFollowing ensures $V^{\pi_{\text{MF}}}(s) \geq V^{(k)}(s)$ for all s and $k \in [K]$.

A strength of MaxFollowing is that it provides direct *action supervision*: the student can directly imitate expert $k^\star(s)$'s action on state s. However, MaxFollowing requires estimating all expert value functions, and its policy is limited to following one of the experts at each state.

2.2 MaxAgg-NoSI

While MaxFollowing can uniformly improve over the experts, its policy is limited to following one of the experts at each state. This can be restrictive when the experts are poor. For example, when $K = 1$, MaxFollowing essentially reduces to imitating the only expert, and can perform poorly if that expert is poor. Max-aggregation (Cheng et al., 2020) is proposed to address this limitation. We refer to this algorithm as MaxAgg-NoSI from here on, which will be clarified in the next subsection.

Like MaxFollowing, MaxAgg-NoSI also estimates the value functions of all experts on each state. Instead of directly following the best expert, it performs one-step policy improvement over the maximum expert value. Let $f^{\max}(s) = \max_{k\in[K]} V^{(k)}(s)$. The target policy of MaxAgg-NoSI is

$$\pi_{\text{MA}}(s) = \operatorname{argmax}_{a\in\mathcal{A}} A^{\max}(s,a) := \operatorname{argmax}_{a\in\mathcal{A}} R(s,a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[f^{\max}(s')] - f^{\max}(s).$$

With accurate value estimates, the resulting policy is no worse than any single expert. It gains additional benefits due to the one-step improvement. Unlike MaxFollowing, MaxAgg-NoSI uses *value supervision* rather than action supervision: The learner explores primitive actions (i.e., actions in the environment) and updates toward actions with larger $A^{\max}(s,a)$. This makes MaxAgg-NoSI more robust than directly imitating expert actions, but it requires more exploration from the student.

Cheng et al. (2020) use a PPO-style update with a $\text{GAE}_{\lambda_{\text{MA}}}$ estimator built from $A^{\max}$. The parameter λ_{MA} controls the interpolation between bootstrapping from $f^{\max}$ when $\lambda_{\text{MA}} \approx 0$ and using longer reward information from the student's own rollouts when $\lambda_{\text{MA}} \approx 1$. Thus, λ_{MA} balances RL and imitation learning (IL). (We use the MA subscript, abbreviating **M**ax**A**gg, to emphasize the distinction from the typical λ for GAE.) This PPO-like algorithm is called MAMBA in Cheng et al. (2020), but we call it just MaxAgg-NoSI again for simplicity.

2.3 MaxAgg-SI

A limitation of MaxAgg-NoSI is that the bootstrap value $f^{\max}$ is static and does not move toward the optimal value $V^\star$ even when the student improves. This limits the ability of the student to approach the true optimal policy. To address this, Liu et al. (2024) propose a *self-inclusive* version of MaxAgg-NoSI, which they call $\max^+$-aggregation, while we call it MaxAgg-SI from here on. Section A has the full pseudocode for MaxAgg-NoSI and MaxAgg-SI.

The key idea is to treat the current student as an additional, $(K+1)$-th expert. The aggregation value becomes $f_t^{\max,+}(s) = \max\left\{V^{(1)}(s), \ldots, V^{(K)}(s), V^{\pi_t}(s)\right\}$, where π_t is the student's current policy. As the student improves, its own value can dominate all fixed experts' values. In that regime, the update becomes closer to ordinary RL, allowing the learner to move beyond the fixed experts. MaxAgg-SI remains a value-supervision method: it still relies on value estimation and primitive-action exploration, rather than directly cloning expert actions.

Action Supervision vs. Value Supervision. The above methods differ in how they use the experts. MaxFollowing uses action supervision, which reduces the need for exploration, but limits the student to the selected experts' actions. In contrast, MaxAgg-NoSI and MaxAgg-SI use value supervision: the learner searches over primitive actions using a value target derived from the experts. This can improve beyond the experts, especially with self-inclusion, but it requires more exploration from the student and may underuse the experts during the early stages of training. Our approach combines the benefits of both supervisions, as we illustrate in the next section.

3 Proposed Algorithm: Stitch-Imitate-Optimize (SIO)

We improve prior methods by (i) enabling the student to exploit the experts more effectively early in training, while (ii) preserving its ability to improve (explore) beyond them. For (i), we propose

Algorithm 1: SIO (Stitch-Imitate-Optimize).

Inputs: K experts $\pi^{(1)}, \dots, \pi^{(K)}$.
Hyperparameters: learning rates (LRs) α_{selector}, α_{student}, α_{critic}, BC weight β (we use $\beta = 1$), GAE parameter λ, clipping factor ϵ_{clip}, batch size N, number of updates per batch M, minibatch size B, student probability minimum ξ.
Definitions: $\text{clip}(x) := \max\{\min\{x, 1+\epsilon_{\text{clip}}\}, 1-\epsilon_{\text{clip}}\}$, selector with minimum student probability ξ: $q_\omega(k \mid s) := (1-\xi)\, p_\omega(k \mid s) + \xi\, \mathbf{1}[k = K+1]$.
Initialization: selector network $p_\omega : \mathcal{S} \to \Delta_{K+1}$, student network $\pi_\theta : \mathcal{S} \to \Delta_{\mathcal{A}}$, critic $V_\phi : \mathcal{S} \to \mathbb{R}$.
for $t = 1, 2, \dots$ **do**
  Observe state s_1.
  for $i = 1, \dots, N$ **do**
    Sample expert or student $k_i \sim q_\omega(\cdot \mid s_i)$.
    Sample action $a_i \sim \begin{cases} \pi^{(k_i)}(\cdot \mid s_i) & \text{if } k_i \le K, \\ \pi_\theta(\cdot \mid s_i) & \text{if } k_i = K+1. \end{cases}$
    Observe reward and next state (r_i, s_i').
    Draw s_{i+1} from initial distribution if s_i' is a terminal state; otherwise, $s_{i+1} = s_i'$.
  Compute advantage estimators: $(\hat{A}_i)_{i=1}^N = \text{GAE}_\lambda((s_i, r_i, s_i')_{i=1}^N, V_\phi)$.
  Define $\hat{V}_i = V_\phi(s_i) + \hat{A}_i$ for $i \in [N]$.
  Define $q_{\text{old}} = q_\omega$ and $\pi_{\text{old}} = \pi_\theta$.
  for $j = 1, \dots, M$ **do**
    Sample a minibatch $\mathcal{B} \subset [N]$ with size $|\mathcal{B}| = B$.
    Define $\rho_{\omega,i} = \frac{q_\omega(k_i|s_i)}{q_{\text{old}}(k_i|s_i)}$, $\rho_{\omega,i}^{\text{clip}} = \text{clip}(\rho_{\omega,i})$, $\rho_{\theta,i} = \frac{\pi_\theta(a_i|s_i)}{\pi_{\text{old}}(a_i|s_i)}$, $\rho_{\theta,i}^{\text{clip}} = \text{clip}(\rho_{\theta,i})$ for $i \in \mathcal{B}$.
    Compute minibatch-normalized advantage $\widetilde{A}_i$ from $\hat{A}_i$.
    $\omega \leftarrow \omega + \alpha_{\text{selector}} \nabla_\omega \frac{1}{B} \sum_{i \in \mathcal{B}} \min\left\{\rho_{\omega,i} \widetilde{A}_i, \rho_{\omega,i}^{\text{clip}} \widetilde{A}_i\right\}$.
    $\theta \leftarrow \theta + \alpha_{\text{student}} \nabla_\theta \frac{1}{B} \sum_{i \in \mathcal{B}} \left(\mathbf{1}[k_i = K+1] \min\left\{\rho_{\theta,i} \widetilde{A}_i, \rho_{\theta,i}^{\text{clip}} \widetilde{A}_i\right\} + \beta \mathbf{1}[k_i \le K] \log \pi_\theta(a_i|s_i)\right)$.
    $\phi \leftarrow \phi - \alpha_{\text{critic}} \nabla_\phi \frac{1}{B} \sum_{i \in \mathcal{B}} \left(V_\phi(s_i) - \hat{V}_i\right)^2$.
Deployment: π_θ.

to *stitch the experts* and use action supervision from the stitched expert policy to directly guide the student. For (ii), we incorporate *self-inclusion*, which lets the algorithm smoothly shift from imitation learning to RL as the student approaches the best stitching of the experts.

Stitching the Experts. By *stitching*, we mean allowing different experts to act at different states, combining their strengths. MaxFollowing and MaxAgg-NoSI can somewhat *stitch* over experts in that they learn from the best expert value baseline $\max_k V^{(k)}(s)$ at every state s, and the maximizers differ among states. However, their theoretical guarantee only competes with the best single policy. We instead aim to compete with the *optimal stitching* of the experts, which could improve over MaxFollowing and MaxAgg-NoSI when the experts are *complementary*, i.e. good at different states. Section B.2 has a toy example where SIO's selector learns optimal stitching (under some assumptions), but MaxFollowing and MaxAgg-NoSI do not. We achieve this stitching via a high-level *selector* that selects which expert should act at each state, with the goal of maximizing the return induced by this selection. Equivalently, this selector optimizes its policy in a meta-MDP where experts are actions. As the experts are not available at deployment, we further let the student behavior-clone (BC) the selected expert's action at each state, by maximizing the log-likelihood $\log \pi_\theta(a_i|s_i)$. We find this simple BC-based algorithm, **Stitch-Imitate-Optimize** with No Self-Inclusion (**SIO-NoSI**), empirically effective (Section 5.1).

Exploring Beyond the Experts. However, ideally we would like to explore beyond the expert actions. This theoretically allows the student to find the optimal policy of the environment rather than just the best stitching of the experts. Inspired by MaxAgg-SI (Liu et al., 2024), we propose a self-inclusive (SI) version of SIO-NoSI, which we call just **SIO**. Algorithm 1 shows our practical implementation (we explain its q_ω later in this section). SIO modifies the BC variant of SIO-NoSI by:

(1) adding an RL term for the student training, alongside the BC term; and (2) allowing the selector to select the student ($k_i = K + 1$) as well, as opposed to only an expert ($k_i \leq K$). The selector is still trained to maximize returns from its selection. The student is trained to maximize return from its own actions (when it is selected by the selector), and to behavior-clone actions from experts (when an expert is selected by the selector). Concretely, SIO adds a PPO (Schulman et al., 2017) term, $\mathbf{1}[k_i = K+1] \min\{\rho_{\theta,i}\widetilde{A}_i,\ \rho^{\text{clip}}_{\theta,i}\widetilde{A}_i\}$, alongside the BC term, $\beta\mathbf{1}[k_i \leq K] \log \pi_\theta(a_i|s_i)$. We add β to balance the PPO and BC terms, though we use $\beta = 1$ except in our ablations. This is the student update at Line 23. The student is deployed at test time. We find SIO both theoretically (Section 4) and empirically (Section 5) effective.

Intuition. Unlike approaches that pretrain with BC and then post-train with RL, and unlike approaches that naively train with BC and RL together, SIO can automatically use BC only when it is increasing returns: this can vary over states, experts, and training steps. A simplified view is two phases. Phase one: the student is worse than the experts, so the selector mostly stitches expert actions, and the student clones. Phase two: once the selector has learned to stitch the experts and the student has cloned this behavior, the student will be selected with higher probability, $p_\omega(K+1|s)$, due to its improved performance. Every time the student is selected, it trains with RL, so can eventually surpass the experts, which would cause the selector to further reduce the weight on the experts and hence the probability of BC. However, to emphasize, the algorithm might do even better than a simple two-phase approach over all states, experts, and training steps. Section B further contrasts with related work, especially MaxAgg.

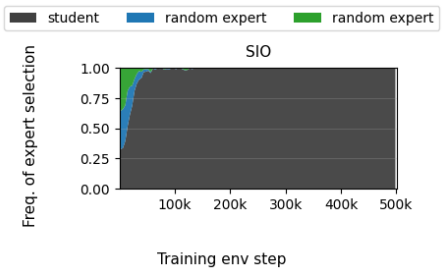

Figure 1: SIO quickly avoids BC of random experts. `CartPole-v1`, $\xi = 0$.

Prioritizing the Deployment Policy. The naive implementation of SIO initializes its selector policy with similar probabilities over all of the experts, including the student policy as if it were an expert. This is likely often undesirable because we ultimately will deploy the student policy. Even though the naive implementation of SIO can automatically learn to favor the student once the student has surpassed the experts (by its RL term, or by learning to stitch the best experts through BC, or both), this mechanism alone might not prioritize the student enough. To handle this, we add a student probability minimum hyperparameter, ξ, which we find empirically increases robustness to bad experts, especially if there are many of them (Section 5.1). This hyperparameter is analogous to λ_{MA} in MaxAgg, which balances RL and IL. Specifically, when selecting an expert k_i to act in the environment, instead of using the original selector $p_\omega(\cdot \mid s)$, we select

$$k_i \sim q_\omega(k \mid s_i) \;=\; (1-\xi)\, p_\omega(k \mid s_i) \;+\; \xi\, \mathbf{1}[k = K+1]\,,$$

so the student policy is selected with probability at least ξ (Line 6, Line 11).

4 Theoretical Analysis

In this section, we analyze a population version of our main algorithm. The goal is to formalize the two mechanisms behind SIO. First, the high-level selector policy learns to stitch the best experts across states. Second, the low-level student policy learns from this stitched behavior policy while also improving beyond the experts through self-inclusion. Because this idealized population model differs from the implemented algorithms, we view the analysis as a verification of these mechanisms, rather than a practical performance guarantee.

Algorithm 2 is a full-information idealization of Algorithm 1. The behavior policy μ_t defined in Line 7 corresponds to the policy used to collect data in Algorithm 1. Additionally, it replaces PPO clipping with KL regularization, which lets us view the updates as mirror-ascent steps. Lastly, the BC weight $\beta_t(s)$ is made state-dependent, as needed by our theoretical analysis.

Regret of the Behavior Policy. We first show that the behavior policy μ_t can compete with any policy μ that can be represented as a statewise mixture of the experts. If μ cannot be represented this way, the regret pays only for the residual part of μ that is not covered by the experts.

Algorithm 2: Population version of SIO for theoretical analysis.

1 **Hyperparameters**: KL-regularization parameter η, BC weights $\beta_t(s)$.

2 **Initialization**: Set $p_1(\cdot|s)$ and $\pi_1(\cdot|s)$ to be uniform for all states s.

3 **for** $t = 1, 2, \ldots, T$ **do**

4 $\quad p_{t+1} = \operatorname{argmax}_p \mathbb{E}_{s\sim d^{\mu_t}} \left[\mathbb{E}_{k\sim p(\cdot|s)} \left[A_t^{\mu_t}(s,k)\right] - \eta^{-1} D_{\mathrm{KL}}(p(\cdot|s) \,\|\, p_t(\cdot|s))\right],$

5 $\quad \pi_{t+1} = \operatorname{argmax}_\pi \mathbb{E}_{s\sim d^{\mu_t}} \big[p_t(K+1|s) \left(\mathbb{E}_{a\sim\pi(\cdot|s)} \left[A^{\mu_t}(s,a)\right] - \eta^{-1} D_{\mathrm{KL}}(\pi(\cdot|s) \,\|\, \pi_t(\cdot|s))\right)$
$\qquad\qquad + \beta_t(s) \sum_{k=1}^K p_t(k|s) \mathbb{E}_{a\sim\pi^{(k)}(\cdot|s)}[\log \pi(a|s)]\big]$

6 $\quad$ where μ_t is the policy induced by the expert distribution p_t and student policy π_t:

7 $\quad \mu_t(a|s) = \sum_{k=1}^K p_t(k|s)\pi^{(k)}(a|s) + p_t(K+1|s)\pi_t(a|s)$

8 $\quad$ and $A_t^{\mu_t}$ is the expert-level advantage function:

9 $\quad A_t^{\mu_t}(s,k) = \begin{cases} \langle \pi^{(k)}(\cdot|s), A^{\mu_t}(s,\cdot)\rangle & \text{if } k \in [K], \\ \langle \pi_t(\cdot|s), A^{\mu_t}(s,\cdot)\rangle & \text{if } k = K+1. \end{cases}$

Theorem 1. *For any policy $\mu : \mathcal{S} \to \Delta_{\mathcal{A}}$, assume that $\mu(\cdot|s)$ can be decomposed as*

$$\mu(\cdot|s) = \textstyle\sum_{k=1}^K p_\mu(k|s)\pi^{(k)}(\cdot|s) + p_\mu(K+1|s)\mu_\perp(\cdot|s) \tag{1}$$

for some $p_\mu(\cdot|s) \in \Delta_{K+1}$ and $\mu_\perp(\cdot|s) \in \Delta_{\mathcal{A}}$. Then the update in Line 4 with $p_1(\cdot|s) = \frac{\mathbf{1}}{K+1}$ and optimal choice of η ensures

$$\textstyle\sum_{t=1}^T \left(J(\mu) - J(\mu_t)\right) \le O\big(L\sqrt{T\log(K+1)}\big) + \mathbb{E}_{s\sim d^\mu}\left[p_\mu(K+1|s)\mathrm{Reg}_{\mathrm{stu}}(\mu_\perp; s)\right], \tag{2}$$

where $\mathrm{Reg}_{\mathrm{stu}}(\pi; s) := \sum_{t=1}^T \langle \pi(\cdot|s) - \pi_t(\cdot|s), A^{\mu_t}(s,\cdot)\rangle$ and $L = \max_{\pi,s,a} |A^\pi(s,a)|$.

Theorem 1 formalizes the statewise stitching effect of the selector policy. The first term in (2), $L\sqrt{T\log(K+1)}$, is the standard mirror-ascent regret for learning among $K+1$ experts. The last term in (2) is the price of comparing against the residual of μ that is not represented by the experts combination. By the decomposition (1), if $\mu(\cdot|s)$ lies in the convex hull of $\{\pi^{(k)}(\cdot|s)\}_{k=1}^K$, then we can choose $p_\mu(K+1|s) = 0$ and the residual cost at state s vanishes.

Note that this guarantee depends only on the selector-policy update Line 4, not on the student-policy update Line 5. Intuitively, if the student policy is still poor, the selector can place little mass on the student branch and focus on stitching among the fixed experts. Thus, the behavior policy can already compete with the best expert stitching before the student becomes useful.

Regret of the Student Policy. Now we turn to the student policy π_t. We show that, under certain assumptions, the student can inherit the stitched behavior and eventually improve beyond the experts.

Theorem 2. *Let $\mu^\star$ be the optimal policy. Assume the state density ratio between any two policies is bounded by C (Assumption 1), and $\beta_t(s) = \lambda p_t(K+1|s)$ for some $\lambda > 0$. Then with optimal choices of η and λ, the regret of the student policy satisfies*

$$\textstyle\sum_{t=1}^T \left(J(\mu^\star) - J(\pi_t)\right) \le O\big(C^2 L\sqrt{T\log((K+1)|\mathcal{A}|)}\big),$$

where $L = \max_{\pi,s,a} |A^\pi(s,a)|$.

A key step in the proof is to relate the student's suboptimality directly to that of the behavior policy: $J(\mu^\star) - J(\pi_t) \le C\mathbb{E}_{s\sim d^{\mu^\star}}\left[\langle \mu^\star(\cdot|s) - \pi_t(\cdot|s), A^{\mu_t}(s,\cdot)\rangle\right] + C(C-1)\big(J(\mu^\star) - J(\mu_t)\big)$. Intuitively, if the selector policy is already near-optimal but the student is not, then the behavior policy's advantages must provide a useful improvement direction for the student. The student's RL update controls the cumulative advantage term, with BC contributing at most λT to the bound. Together with the selector's regret guarantee, this also controls the behavior-policy regret.

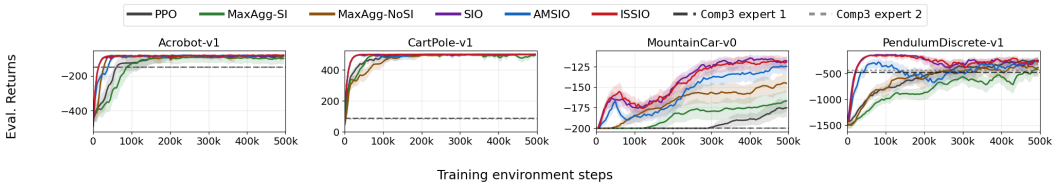

Figure 2: `Comp3` learning curves, with all learning rates (LRs) at 10^{-3}. MaxAgg $\lambda_{\text{MA}} = 0.95$, and SIO $\xi = 0$. All algorithms learn quickly on `Acrobot-v1` and `CartPole-v1`, though our algorithms are all among the fastest. On `MountainCar-v0` and `PendulumDiscrete-v1`, our algorithms learn faster than both MaxAgg (regardless of SI) and PPO. Thirty seeds, 95% bootstrap CIs.

5 Experiments

First, we test on four discrete-action, classic control tasks (Brockman et al., 2016; Towers et al., 2024): `CartPole-v1`, `Acrobot-v1`, `MountainCar-v0`, and a discrete-action version of `Pendulum-v1`. We consider the following expert sets:

1. `R`: A single random expert: a policy that acts uniformly randomly.
2. `2R`: Two random experts.
3. `8R`: Eight random experts.
4. `G`: A single great expert: a DQN net trained on the environment, which had its LR tuned to maximize the expert's returns.
5. `G2R`: The great expert and two random experts.
6. `G7R`: The great expert and seven random experts.
7. `Comp1`: The great expert modified into two new, complementary experts. One position value of the state space is split into two halves, and each of the two experts acts uniformly randomly on opposite halves (remaining great on its other half). For example, for `CartPole-v1`, the split is the pole angled left vs. right.
8. `Comp2`: The same as `Comp1`, but splitting on the velocity value of the state space. For example, for `CartPole-v1`, the split is the pole moving left vs. right.
9. `Comp3`: Roughly the same as `Comp1`, but a checkerboard pattern over the state space instead of a left-right split. For example, for `CartPole-v1`, each expert is great on only half of twenty bins over cart position and twenty bins over pole angle.
10. `Comp4`: Roughly the same as `Comp3`, but eight experts instead of two, where each expert is great on only one-eighth of the checkerboard.

To measure learning speed, we compare average evaluation returns during all of training. In our experiments, on all non-random expert sets, our basic algorithm SIO with $\xi = 0.5$ attains higher means than MaxAgg-SI (self-inclusion) and MaxAgg-NoSI (no self-inclusion), with only one overlapping CI (Table 1). We also include two additional variants, Advantage-Masked SIO (**AMSIO**) and Importance-Sampling SIO (**ISSIO**), that differ from SIO only in the BC term of the student-actor update (Line 23): they replace $\beta \mathbf{1}[k_i \leq K] \log \pi_\theta(a_i \mid s_i)$ with (new terms in orange):

$$\underbrace{\mathbf{1}\left[\widetilde{A}_i > 0\right] \beta \mathbf{1}[k_i \leq K] \log \pi_\theta(a_i \mid s_i)}_{\text{AMSIO}} \quad \text{or} \quad \underbrace{\beta \mathbf{1}[k_i \leq K] \min\{\rho_{\theta,i},\, 1 + \epsilon_{\text{clip}}\}}_{\text{ISSIO}}.$$

We find both even more robust than SIO. All but one of AMSIO's mean returns in Table 1 are higher than both MaxAgg-SI and MaxAgg-NoSI. Further, on that exception, `R`, AMSIO's returns are smaller by less than 1% (and the CIs overlap). Out of all of our algorithms and baselines, ISSIO has the best mean rank over all environment–expert combinations (Table 6). With Adam's (Kingma & Ba, 2014) default LR for all algorithms, SIO, AMSIO, and ISSIO all beat both MaxAgg-SI and MaxAgg-NoSI in mean returns on at least 77% of the environment–expert combinations, while ISSIO exceeds 80% against both (Table 9). As expected, we also find that the better the experts, the better MaxAgg-NoSI performs compared to MaxAgg-SI.

For our main classic control results on all algorithms, we sweep the actor and critic LRs. We show the per-LR results in Section K, and the frequency of each expert or student being selected in Section L.

Table 1: For each environment, evaluation returns are averaged over all of training and over the learning-rate grid (Section J.1), then normalized so a uniformly random policy's returns are 0, and DQN's returns are 100 under the same averaging. The per-environment normalized returns are then averaged over environments. Plain PPO under the same procedure gets 49.5 normalized returns. All tables and figures use means and 95% bootstrap CIs over ten seeds unless otherwise specified. For concerns about outlier environments, and for default LR (10^{-3}) results, see Section C. The best mean per expert set is **bold**, and 95% bootstrap CIs overlapping the best mean's CI are highlighted.

Experts	MaxAgg-SI $\lambda_{\text{MA}} = 0.95$	MaxAgg-NoSI $\lambda_{\text{MA}} = 0.95$	SIO $\xi = 0.5$	AMSIO $\xi = 0.5$	ISSIO $\xi = 0.5$
R	45.5 [43.2, 47.7]	**55.1 [53.6, 56.7]**	38.1 [36.1, 40.0]	54.8 [52.7, 56.9]	46.7 [44.4, 49.2]
2R	42.7 [40.6, 44.8]	52.5 [50.5, 54.3]	36.3 [34.0, 38.6]	**53.2 [50.9, 55.6]**	43.4 [40.9, 45.9]
8R	42.2 [40.0, 44.3]	48.9 [46.9, 51.0]	33.3 [31.3, 35.5]	**50.5 [48.7, 52.3]**	38.6 [36.7, 40.6]
G	52.7 [49.7, 55.7]	76.9 [74.7, 79.0]	**134.3 [129.7, 139.0]**	104.1 [101.7, 106.5]	130.9 [125.9, 135.7]
G2R	52.5 [49.8, 55.3]	73.5 [71.9, 75.1]	96.7 [93.4, 100.1]	87.4 [84.6, 90.4]	**100.5 [97.5, 103.6]**
G7R	51.2 [48.6, 53.8]	64.9 [63.1, 66.7]	72.5 [69.7, 75.4]	72.8 [70.7, 75.0]	**76.9 [74.2, 79.7]**
Comp1	50.0 [47.5, 52.6]	73.7 [70.7, 76.7]	96.0 [94.5, 97.5]	96.0 [94.1, 97.8]	**97.9 [96.3, 99.7]**
Comp2	49.2 [46.8, 51.4]	68.8 [66.3, 71.5]	99.9 [98.0, 102.0]	87.8 [85.7, 90.0]	**100.2 [98.4, 102.2]**
Comp3	50.8 [48.3, 53.2]	81.0 [78.1, 83.7]	104.7 [102.4, 106.9]	102.1 [100.0, 104.3]	**111.9 [108.5, 115.2]**
Comp4	44.1 [41.9, 46.3]	58.9 [57.3, 60.5]	62.4 [60.4, 64.6]	**68.8 [67.0, 70.8]**	67.9 [66.0, 70.0]

Higher-Dimensional, Continuous-Action Environments. We also test on MuJoCo (Todorov et al., 2012; Brockman et al., 2016). The results align with our existing findings. For example, AMSIO's CIs overlap or exceed MaxAgg-SI's and MaxAgg-NoSI's CIs in every pooling we examine: per expert set, or rank-wise over all environments and expert sets, or per environment (Table 2 and Section F). Further, AMSIO's (PPO-normalized) returns are often far higher than MaxAgg's: e.g., *over* $4\times$ *higher* on G, G2R, and G7R. Section F also discusses MaxAgg's relatively low returns in this setting.

Table 2: Table 1 for continuous-action environments instead of discrete-action environments, with all LRs set to 10^{-3}, and with two additional expert sets (M and 2M, described in Section J.2.2). Normalized so that PPO's returns are 100 (and, again, a uniformly random policy's returns are 0).

Experts	MaxAgg-SI $\lambda_{\text{MA}} = 0.95$	MaxAgg-NoSI $\lambda_{\text{MA}} = 0.95$	SIO $\xi = 0.5$	AMSIO $\xi = 0.5$	ISSIO $\xi = 0.5$
R	82.6 [73.4, 91.8]	35.2 [32.9, 37.5]	78.3 [75.9, 80.6]	**89.5 [83.8, 95.1]**	62.5 [58.8, 66.3]
2R	**92.2 [79.8, 105.0]**	42.7 [40.5, 45.0]	74.3 [71.3, 77.1]	80.2 [75.1, 84.6]	51.0 [49.2, 52.9]
8R	76.2 [71.3, 81.3]	44.2 [40.6, 47.5]	69.3 [66.0, 72.7]	**80.3 [76.7, 84.5]**	45.3 [44.3, 46.5]
M	59.9 [53.1, 69.1]	55.1 [52.2, 58.2]	196.8 [193.9, 199.9]	**197.9 [194.7, 201.1]**	152.2 [150.4, 154.0]
2M	59.4 [54.6, 65.0]	60.2 [54.4, 66.8]	146.3 [141.7, 151.4]	151.6 [146.4, 156.8]	**157.2 [149.5, 165.3]**
G	52.7 [49.9, 55.3]	55.9 [52.4, 59.1]	**250.4 [248.5, 252.6]**	248.7 [246.4, 251.0]	232.6 [229.6, 235.6]
G2R	56.5 [52.4, 61.0]	57.4 [53.3, 62.6]	242.9 [239.6, 246.3]	**250.8 [247.2, 254.4]**	247.3 [244.4, 250.0]
G7R	50.0 [47.0, 52.9]	47.5 [44.7, 50.4]	213.8 [210.6, 216.7]	219.3 [213.0, 224.7]	**232.1 [225.7, 238.1]**
Comp1	41.3 [38.4, 45.0]	40.7 [38.6, 42.8]	241.5 [239.1, 244.0]	246.3 [242.8, 249.6]	**247.9 [244.2, 251.5]**
Comp2	67.6 [58.5, 75.7]	42.8 [39.2, 46.2]	241.2 [238.5, 243.8]	**253.4 [250.2, 256.4]**	248.0 [243.9, 252.3]
Comp3	71.3 [66.3, 75.9]	52.4 [49.2, 55.9]	208.6 [203.3, 214.1]	**229.5 [225.0, 233.9]**	188.8 [184.1, 193.8]
Comp4	69.9 [63.5, 77.2]	49.3 [47.1, 51.3]	68.0 [64.9, 70.8]	**90.7 [87.4, 94.1]**	47.3 [46.5, 48.2]

5.1 Additional Experiments

We ablate the BC coefficient and self-inclusion in Table 3, and the student-selection floor in Table 4.

Varying the BC Coefficient. All results in this paper use $\beta = 1$ except where labeled otherwise. $\beta = 1$ works well, perhaps because SIO indeed automatically tends to avoid training data from bad

Table 3: SIO ablations ($\xi = 0$). Again normalized by DQN (100) and a random policy (0). *Left*: our default BC coefficient $\beta = 1$ vs. smaller coefficients. *Right*: our default self-inclusive (SI) vs. NoSI.

Experts	$\beta = 1$	$\beta = 0$	$\beta = 0.5$	SI	NoSI
R	34.0 [31.7, 36.4]	**52.0 [49.4, 54.4]**	38.1 [35.7, 40.6]	**34.0 [31.7, 36.4]**	-10.3 [-11.5, -9.1]
2R	26.2 [23.8, 28.5]	**44.7 [41.9, 47.5]**	29.2 [26.7, 31.8]	**26.2 [23.8, 28.5]**	-11.8 [-12.7, -10.8]
8R	9.7 [7.4, 12.1]	**27.5 [25.1, 29.9]**	13.6 [11.3, 15.8]	**9.7 [7.4, 12.1]**	-9.5 [-10.8, -8.1]
G	**141.5 [137.5, 145.5]**	29.5 [27.4, 31.7]	133.1 [129.7, 136.5]	141.5 [137.5, 145.5]	**160.8 [159.9, 161.6]**
G2R	**106.7 [103.0, 110.6]**	33.4 [31.4, 35.5]	98.2 [94.9, 101.8]	106.7 [103.0, 110.6]	**107.1 [103.4, 111.2]**
G7R	77.7 [74.6, 81.1]	27.5 [24.6, 30.3]	**80.5 [76.6, 84.4]**	77.7 [74.6, 81.1]	**81.6 [77.7, 85.7]**
Comp1	**111.4 [108.3, 114.8]**	44.1 [41.3, 46.8]	106.0 [102.6, 109.4]	111.4 [108.3, 114.8]	**116.6 [113.1, 120.4]**
Comp2	**107.2 [104.7, 109.7]**	34.2 [31.5, 36.7]	102.2 [100.3, 104.1]	107.2 [104.7, 109.7]	**114.9 [112.5, 117.4]**
Comp3	**133.2 [129.0, 137.4]**	47.8 [45.6, 50.1]	128.0 [124.5, 131.4]	133.2 [129.0, 137.4]	**158.3 [157.5, 159.1]**
Comp4	**55.9 [54.2, 57.8]**	39.0 [36.8, 41.2]	53.1 [51.7, 54.5]	**55.9 [54.2, 57.8]**	54.8 [53.6, 55.9]

experts, as we have designed it. However, the left side of Table 3 shows that, depending on the expert set, SIO can learn even better with a tuned β.

No Self-Inclusion (NoSI). The right side of Table 3 compares self-inclusion (SI) to NoSI. As expected, NoSI seems to work better when the highest-performing individual expert is stronger, and also when each of the experts is individually stronger. The negative values for some NoSI cells reflect NoSI performing worse than a uniformly random policy, which is common for many algorithms and LRs on Pendulum, where often the agent must counterintuitively rotate away from the goal angle to gain momentum.

Table 4: SIO with a student-selection floor ($\xi = 0.5$) vs. no floor ($\xi = 0$).

Experts	$\xi = 0$	$\xi = 0.5$
R	34.0 [31.7, 36.4]	**38.1 [36.1, 40.0]**
2R	26.2 [23.8, 28.5]	**36.3 [34.0, 38.6]**
8R	9.7 [7.4, 12.1]	**33.3 [31.3, 35.5]**
G	**141.5 [137.5, 145.5]**	134.3 [129.7, 139.0]
G2R	**106.7 [103.0, 110.6]**	96.7 [93.4, 100.1]
G7R	**77.7 [74.6, 81.1]**	72.5 [69.7, 75.4]
Comp1	**111.4 [108.3, 114.8]**	96.0 [94.5, 97.5]
Comp2	**107.2 [104.7, 109.7]**	99.9 [98.0, 102.0]
Comp3	**133.2 [129.0, 137.4]**	104.7 [102.4, 106.9]
Comp4	55.9 [54.2, 57.8]	**62.4 [60.4, 64.6]**

No Probability Minimum for Selecting the Student, $\xi = 0$. Table 4 demonstrates the benefits and downsides of the ξ hyperparameter: the student probability minimum helps on the random-expert sets, while $\xi = 0$ is better for most great, mixed, and complementary expert sets.

Frozen Selector. Section E and Section G test the benefit of learning a selector policy. They freeze SIO's selector policy at its initialization. We find that learning the selector often attains higher returns (e.g., it does so on ten out of twelve of the higher-dimensional expert sets, and by up to $4.71\times$), and always attains nearly as high returns as not learning the selector (the worst case over all twelve expert sets still attains $0.86\times$ the returns).

6 Limitations and Discussion

Limitations. We assume expert actions are observable in training, though this is common (e.g., with large language models). Like the main prior works (Cheng et al., 2020; Liu et al., 2024), we leave more practical experiments for important future research. See Section 4 for our theory limitations.

Future Algorithms. Combining the advantage-masking of AMSIO with the importance-sampling BC term of ISSIO might improve over both algorithms. Alternatively, using off-policy learning for the student might more efficiently use expert data than our initial implementation. TD3+BC (Fujimoto & Gu, 2021) is framed as an offline RL algorithm, but might be a natural fit for the student given how beneficial we find our BC term. Uncertainty-based approaches (Jin et al., 2021) for the student similarly might be important.

AI USE STATEMENT

In this work, we used generative AI tools: to aid and polish writing (e.g., to help come up with names for algorithms and concepts, and to improve clarity); to help retrieval and discovery (e.g., to help find related works); to draft sections of the paper (e.g., the expert constructions appendices); for generating synthetic datasets (the exact numbers and features to split on for the `Comp` experts); to aid in proving mathematical claims; to help write our code; to get additional feedback on our hypotheses and methodology; and to help find mistakes in our paper and code. We have reviewed all AI-assisted work. For example, we manually checked our theory, manually searched for related works, and verified and tested LLM-generated code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We include anonymized code in the supplementary materials to reproduce our main experiments (for both the low-dimensional, discrete-action environments, and the higher-dimensional, continuous-action environments). We include proofs in Section I, and dataset details and further experiment details in Section J.

REFERENCES

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Jonathan D. Chang, Andrew Drozdov, Shubham Toshniwal, Owen Oertell, Alexander Trott, Jacob Portes, Abhay Gupta, Pallavi Koppol, Ashutosh Baheti, Sean Kulinski, Ivan Zhou, Irene Dea, Krista Opsahl-Ong, Simon Favreau-Lessard, Sean Owen, Jose Javier Gonzalez Ortiz, Arnav Singhvi, Xabi Andrade, Cindy Wang, Kartik Sreenivasan, Sam Havens, Jialu Liu, Peyton DeNiro, Wen Sun, Michael Bendersky, and Jonathan Frankle. KARL: Knowledge agents via reinforcement learning. *arXiv preprint arXiv:2603.05218*, 2026.

C. Cheng, X. Yan, N. Wagener, and B. Boots. Fast policy learning through imitation and reinforcement. In *Uncertainty in Artificial Intelligence*, 2018.

Ching-An Cheng, Andrey Kolobov, and Alekh Agarwal. Policy improvement via imitation of multiple oracles. *Advances in Neural Information Processing Systems*, 33:5587–5598, 2020.

CleanRL contributors. CleanRL DQN implementation, 2026a. URL `https://github.com/vwxyzjn/cleanrl/blob/fe8d8a03c41a7ef5b523e2e354bd01c363e786bb/cleanrl/dqn.py`. GitHub commit fe8d8a03c41a7ef5b523e2e354bd01c363e786bb.

CleanRL contributors. CleanRL PPO implementation, 2026b. URL `https://github.com/vwxyzjn/cleanrl/blob/fe8d8a03c41a7ef5b523e2e354bd01c363e786bb/cleanrl/ppo.py`. GitHub commit fe8d8a03c41a7ef5b523e2e354bd01c363e786bb.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Lakshita Dodeja, Ondrej Biza, Shivam Vats, Stephen Hart, Stefanie Tellex, Robin Walters, Karl Schmeckpeper, and Thomas Weng. When life gives you BC, make Q-functions: Extracting Q-values from behavior cloning for on-robot reinforcement learning. *arXiv preprint arXiv:2605.05172*, 2026.

Scott Fujimoto and Shixiang (Shane) Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:20132–20145, December 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/a8166da05c5a094f7dc03724b41886e5-Abstract.html`.

GLM-4.5 Team. GLM-4.5: Agentic, reasoning, and coding (ARC) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

GLM-5 Team. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. CleanRL: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Marcel Hussing, Michael Kearns, Aaron Roth, Sikata Bela Sengupta, and Jessica Sorrell. Oracle-efficient reinforcement learning for max value ensembles. *arXiv preprint arXiv:2405.16739*, 2024.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline RL? In *International Conference on Machine Learning*, pp. 5084–5096. PMLR, July 2021. URL https://proceedings.mlr.press/v139/jin21e.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Andrey Kurenkov, Ajay Mandlekar, Roberto Martin-Martin, Silvio Savarese, and Animesh Garg. AC-Teach: A Bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. In *Proceedings of the Conference on Robot Learning*, 2020.

Guohao Li, Matthias Mueller, Vincent Casser, Neil Smith, Dominik L. Michels, and Bernard Ghanem. OIL: Observational imitation learning. *arXiv preprint arXiv:1803.01129*, 2018.

Xuefeng Liu, Takuma Yoneda, Rick Stevens, Matthew Walter, and Yuxin Chen. Blending imitation and reinforcement learning for robust policy improvement. In *The Twelfth International Conference on Learning Representations*, 2024.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, Jinhao Dong, Zhifang Sui, and Fuli Luo. MOPD: Multi-teacher on-policy distillation for capability integration in LLM post-training. *arXiv preprint arXiv:2606.30406*, 2026.

Chiara Mignacco. *A mathematical study of policy orchestration for reinforcement learning*. PhD thesis, Université Paris-Saclay, Paris, France, December 2025. URL https://theses.hal.science/tel-05467153.

NVIDIA. Nemotron 3 Ultra: Open, efficient mixture-of-experts hybrid Mamba-Transformer model for agentic reasoning. *arXiv preprint arXiv:2606.15007*, 2026.

Simon Schmitt, Jonathan J. Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M. Czarnecki, Joel Z. Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, and S. M. Ali Eslami. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sikata Bela Sengupta, Zakaria Mhammedi, and Teodor Vanislavov Marinov. Oracle-efficient adversarial reinforcement learning via Max-Following. In *The Exploration in AI Today Workshop at ICML 2025*, 2025. URL https://openreview.net/forum?id=KdYtu1HiE9.

Idan Shenfeld, Zhang-Wei Hong, Aviv Tamar, and Pulkit Agrawal. TGRL: An algorithm for teacher guided reinforcement learning. In *International Conference on Machine Learning*, pp. 31077–31093. PMLR, 2023.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Xiaomi LLM-Core Team. MiMo-V2-Flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

A FULL MAXAGG PSEUDOCODE

Algorithm 3: MaxAgg (SI-only lines in **red**)

Inputs: K experts $\pi^{(1)}, \dots, \pi^{(K)}$.
Hyperparameters: LRs α_{actor}, α_{experts}, α_{critic}, PPO clipping factor ϵ_{clip}, batch size N, number of updates per batch M, minibatch size B, pretrain episodes per expert N_{pretrain}, environment horizon H, GAE parameter λ_{MA}.
Initialization: expert critic $V_\psi : \mathcal{S} \to \mathbb{R}^K$ with heads $V_\psi^{(k)}$ estimating $V^{(k)}$ for $k \in [K]$, student $\pi_\theta : \mathcal{S} \to \Delta_{\mathcal{A}}$, empty per-expert suffix replay buffers $D_1, \dots, D_K$ (each capped at $N_{\text{buf}} := \max\{N, HN_{\text{pretrain}}\}$ transitions, dropping oldest whole suffixes); $t \leftarrow 0$, cycling expert index $i \leftarrow 0$, next PPO main-env step $t_{\text{next}} \leftarrow 0$.
Initialization: student-critic $V_\phi : \mathcal{S} \to \mathbb{R}$.
// pretrain
for $k = 1, \dots, K$ **do**
  Run N_{pretrain} full episodes under expert $\pi^{(k)}$, appending the resulting transitions to D_k.
  Update $V_\psi^{(k)}$ with λ_{MA}-returns on D_k.
while *True* **do**
  // roll-in, roll-out (RIRO) training
  Pick expert $k \leftarrow 1 + (i \bmod K)$; $i \leftarrow i + 1$.
  Sample RIRO horizon split point $h \sim \text{Uniform}(0, H)$; reset env; $\tau \leftarrow$ empty suffix buffer.
  for $j = 0, 1, 2, \dots$ *until episode end* **do**
    if $j < h$ **then**
      // roll-in: student acts, transition not pushed to replay
      $a_j \sim \pi_\theta(\cdot \mid s_j)$.
    else
      // roll-out: fixed expert k
      $a_j \sim \pi^{(k)}(\cdot \mid s_j)$.
    Step env, observing (r_j, s_{j+1}); $t \leftarrow t + 1$.
    if $j \geq h$ **then**
      append observed transition to τ.
    if $t \geq t_{\text{next}}$ **then**
      Run PPOPHASE; $t_{\text{next}} \leftarrow t + N$.
  // critic update
  Append τ to D_k; drop oldest whole suffixes until D_k contains at most N_{buf} transitions.
  For each suffix in D_k, update $V_\psi^{(k)}$ with λ_{MA}-returns.

Procedure PPOPHASE**:**
  Collect $(s_i, a_i, r_i, s_i')_{i=1}^N$ under the student π_θ.
  Define $f^{\max}(s) = \max_{k \in [K]} V_\psi^{(k)}(s)$ and $f_t^{\max,+}(s) = \max\big(f^{\max}(s),\, V_\phi(s)\big)$.
  Compute advantage estimators: $(\hat{A}_i)_{i=1}^N = \text{GAE}_{\lambda_{\text{MA}}}((s_i, r_i, s_i')_{i=1}^N, f^{\max})$ (non-SI) or $\text{GAE}_{\lambda_{\text{MA}}}((s_i, r_i, s_i')_{i=1}^N, f_t^{\max,+})$ (SI).
  Also compute $(\hat{A}_i^\phi)_{i=1}^N = \text{GAE}_{\lambda_{\text{MA}}}((s_i, r_i, s_i')_{i=1}^N, V_\phi)$, and define $\hat{V}_i = V_\phi(s_i) + \hat{A}_i^\phi$ for $i \in [N]$.
  Define $\pi_{\text{old}} = \pi_\theta$.
  for $j = 1, \dots, M$ **do**
    Sample a minibatch $\mathcal{B} \subset [N]$ with size $|\mathcal{B}| = B$.
    Define $\rho_{\theta,i} = \frac{\pi_\theta(a_i|s_i)}{\pi_{\text{old}}(a_i|s_i)}$, $\rho_{\theta,i}^{\text{clip}} = \text{clip}(\rho_{\theta,i})$ for $i \in \mathcal{B}$.
    Compute minibatch-normalized advantage $\widetilde{A}_i$ from $\hat{A}_i$.
    // student network update
    $\theta \leftarrow \theta + \alpha_{\text{actor}} \nabla_\theta \frac{1}{B} \sum_{i \in \mathcal{B}} \min\left\{\rho_{\theta,i} \widetilde{A}_i, \rho_{\theta,i}^{\text{clip}} \widetilde{A}_i\right\}$.
    // student-critic update
    $\phi \leftarrow \phi - \alpha_{\text{critic}} \nabla_\phi \frac{1}{B} \sum_{i \in \mathcal{B}} \left(V_\phi(s_i) - \hat{V}_i\right)^2$.
Deployment: π_θ.

B Related Work

B.1 Further Comparisons with MaxFollowing and MaxAgg

Table 5: Comparisons with prior algorithms.

	MaxFol.	MaxAgg-NoSI	MaxAgg-SI	SIO-NoSI	SIO
Approach/Surpass Optimal Stitching?	N	N	Y	Y	Y
Action/Value Supervision?	Action	Value	Value	Action	Both
Need to Train $V^{(k)}(s)$ for Every k?	Y	Y	Y	N	N
Data Collection?	–	RIRO	RIRO	Selector	Selector

Approach/Surpass Optimal Stitching? MaxFollowing and MaxAgg-NoSI do not always approach/surpass optimal stitching — we show an example in Section B.2. These methods do "stitch", but not in an optimal way.

In contrast, due to self-inclusion, MaxAgg-SI and SIO can approach the globally optimal policy, and hence can surpass the optimal stitching. However, roughly speaking, before the learner policy surpasses the best expert: MaxAgg-SI is similar to MaxAgg-NoSI, and so only *imitates a sub-optimal stitching*; SIO imitates via the selector network, competing with optimal stitching of the experts.

Here, we have considered only the $\lambda_{\mathrm{MA}} = 0$ versions of MaxAgg-NoSI and MaxAgg-SI. $\lambda_{\mathrm{MA}} > 0$ interpolates MaxAgg with standard RL (PPO in particular). Cheng et al. (2020) does not give a guarantee that this interpolation can approach/surpass optimal stitching (though the $\lambda_{\mathrm{MA}} = 1$ case can in the sense that it is purely standard RL). Further, similar to adding self-inclusion, the interpolation still incorporates imitation of a sub-optimal stitching.

Action/Value Supervision? Under action supervision, the teacher directly tells the student what action should be taken (BC). Under value supervision, the teacher only affects the advantage function used to train the student. Action supervision can allow faster learning from the experts, since it directly clones them. However, unlike value supervision, the student cannot go beyond what the experts do. MaxFollowing, MaxAgg-NoSI, MaxAgg-SI, and SIO-NoSI each use only one of these supervisions, while SIO uses both and can automatically balance them. Early in training, SIO can rely more on action supervision, for faster learning from the experts. Later in training, SIO can rely more on value supervision, for more exploration beyond the experts.

Need to Train $V^{(k)}(s)$ for Every k? In MaxFollowing and MaxAgg, every expert has its own value function $V^{(k)}(s)$. Training each value function uses separate episodes for every expert. SIO instead always acts in the environment with the selector, which more directly learns to stitch experts.

Data Collection? The practical implementations of MaxAgg-NoSI and MaxAgg-SI, both in prior works (Cheng et al., 2020; Liu et al., 2024) and in our own work, use roll-in, roll-out (RIRO) data collection. That is, they roll in with the student over a random horizon length, then roll out the remainder of the episode with an expert. SIO instead always acts in the environment with a policy that is itself training to maximize its returns (the selector). A downside of RIRO training is that, for infinite-horizon tasks, it requires estimating the empirical horizon length periodically throughout training. A benefit of RIRO training is that it might help with long-horizon exploration. In contrast, SIO may take a long time to get to parts of the state space where certain experts are best. In particular, this may happen if an expert happens to be bad near the initial state distribution and near other policies, but great in some far off states. (To our knowledge, prior works do not consider a practical implementation of MaxFollowing, but RIRO would be one natural choice for its data collection.)

B.2 Optimal Stitching Counterexample

The Figure 5 caption of Cheng et al. (2020) describes a task (a modification of Figure 5 itself) where neither MaxFollowing nor MaxAgg-NoSI gives optimal stitching. In particular, it describes swapping the rewards of s_7 and s_{11} from their Figure 5. Fig. 3 shows the resulting task.

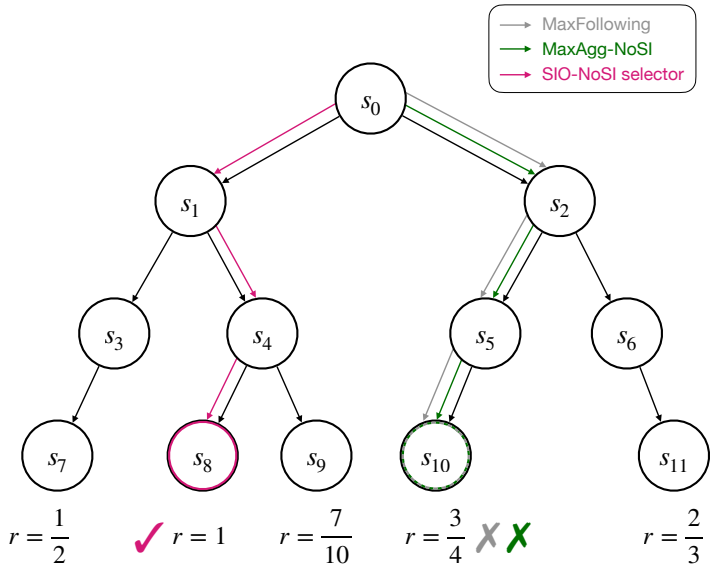

Figure 3: Task from Cheng et al. (2020). Zero rewards except the terminal nodes. Starts at s_0. Two experts: one always goes left (if there is a leftward action at the current state), and the other always goes right (if there is a rightward action at the current state). MaxFollowing and MaxAgg-NoSI get $\frac{3}{4}$ reward, but optimal stitching (as SIO's selector can achieve, even without self-inclusion) gets 1 reward. *MaxFollowing*: at s_0, compares a. always going left ($\frac{1}{2}$) vs b. always going right ($\frac{2}{3}$) — and so goes right, which is already suboptimal. *MaxAgg-NoSI*: at s_0, compares a. going left once, to s_1, then the max of always going left or always going right (max is $\frac{7}{10}$) vs b. going right once, to s_2, then the max of always going left or always going right (max is $\frac{3}{4}$) — and so goes right ($\frac{3}{4}$), like MaxFollowing does, which is already suboptimal.

We consider only the non-self-inclusive (NoSI) algorithms here, but see Section B.1 for an argument that this affects the SI algorithms as well.

This task does not satisfy all of the assumptions of Theorem 2, but Theorem 1 remains applicable. Thus, for this task, our theory only proves (under our Theorem 1 assumptions) that SIO's *selector* policy will learn the optimal stitching. So this task result is an unfair comparison against MaxFollowing and MaxAgg-NoSI because SIO's selector policy is only usable when experts are available (such as during training). However, we think this task still provides helpful intuition.

B.3 Additional Related Work

AC-Teach (Kurenkov et al., 2020) uses a Bayesian critic with Thompson sampling and a commitment rule to choose among expert and student actions. No BC is used, though the DDPG critic target may use expert actions.

Kickstarting (Schmitt et al., 2018) chooses which experts to use per task, not per state. This might not make the best use of experts that are good at different states of one task. Further, Kickstarting has its student always act in the environment, and distills the expert policies at the states the student visits. This might be inefficient if exploration is difficult. We leave testing similar variants of SIO for future work.

Among many other contributions, Mignacco (2025) introduces and theoretically analyzes an algorithm similar to SIO-NoSI, with no exploration in primitive actions.

Q2RL (Dodeja et al., 2026) studies offline-to-online improvement from one pretrained BC policy. It estimates a static Q-function for the BC policy first, and then gates between BC and RL actions by comparing the static BC Q-value to an online RL Q-value. In contrast, SIO dynamically learns

where to use BC throughout training, learning to use BC even where an expert would only be useful if stitched with other policies.

Single-expert algorithms most similar to ours include LOKI (Cheng et al., 2018), which does a phase of IL then a phase of RL, and TGRL (Shenfeld et al., 2023), which uses a Lagrangian approach to adapt how much to learn from the expert at states the student encounters (somewhat similar to Kickstarting). In contrast, throughout training, SIO learns at every state which experts (or the student) should act in the environment, which also controls which experts (if any) the student should learn from. For additional related work, see Table 2 in Liu et al. (2024).

Hussing et al. (2024) introduces MaxIteration, competing with MaxFollowing, under weaker assumptions than MaxAgg-NoSI requires. Sengupta et al. (2025) introduces AdversarialMaxIteration (AMI), handling adversarial initial-state distributions.

SIO can be viewed as hierarchical RL: learning with one trainable option (Sutton et al., 1999) and K static options, though each option runs for only one step, and the K static options are unavailable at deployment time. Additionally, SIO uses the K static options to help teach the trainable option.

Multiple Experts and Large Language Models (LLMs). Recently, many LLMs have been trained with *multi-expert distillation* and related approaches. That is, specialist models (often called "teachers" or "experts") are first trained on different datasets, then used in some form to train a single, generalist model (DeepSeek-AI, 2024; 2025; GLM-4.5 Team, 2025; GLM-5 Team, 2026; Xiaomi LLM-Core Team, 2026; NVIDIA, 2026). Granted, Chang et al. (2026) find better benchmark scores (in particular in OOD generalization) from using multi-task RL instead of multiple experts and supervised learning. However, other recent works also combine multiple experts with on-policy distillation, i.e. Multi-teacher On-Policy Distillation (MOPD) (Xiaomi LLM-Core Team, 2026; NVIDIA, 2026; Ma et al., 2026). Some ideas from our algorithms might be helpful in this area.

C Further Discrete-Action, Low-Dimensional Results

Table 6: To alleviate concerns about averaging and outlier environments, here we compare *rankings* of mean returns over all {algorithm, environment, experts} combinations, ignoring CI overlaps.

Algorithm	Mean rank ↓	Beats MaxAgg-NoSI ↑	Beats MaxAgg-SI ↑
PPO	7.28 [7.03, 7.45]	0.15 [0.12, 0.23]	0.60 [0.50, 0.65]
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	7.50 [7.30, 7.70]	0.20 [0.10, 0.20]	—
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	5.25 [5.12, 5.45]	—	0.80 [0.80, 0.90]
SIO, $\xi = 0$	4.62 [4.38, 4.88]	0.65 [0.62, 0.70]	0.70 [0.70, 0.78]
SIO, $\xi = 0$, $\beta = 0$	8.32 [8.10, 8.47]	0.10 [0.05, 0.15]	0.28 [0.23, 0.35]
SIO, $\xi = 0$, $\beta = 0.5$	5.17 [4.92, 5.40]	0.62 [0.60, 0.68]	0.75 [0.70, 0.80]
SIO, $\xi = 0$, NoSI	5.05 [4.83, 5.20]	**0.68 [0.62, 0.70]**	0.72 [0.68, 0.75]
SIO, $\xi = 0.5$	4.70 [4.42, 4.92]	0.55 [0.53, 0.60]	0.85 [0.78, 0.85]
AMSIO, $\xi = 0.5$	4.15 [3.85, 4.35]	0.62 [0.62, 0.72]	**0.93 [0.85, 0.93]**
ISSIO, $\xi = 0.5$	**2.95 [2.90, 3.33]**	**0.68 [0.60, 0.70]**	0.88 [0.85, 0.90]

Table 7: Per-environment normalized returns, averaged over expert sets (PPO uses no experts).

Algorithm	`Acrobot-v1`	`CartPole-v1`	`MountainCar-v0`	`PendulumDiscrete-v1`
PPO	119.0 [112.1, 125.7]	121.9 [119.6, 124.2]	3.1 [0.6, 6.6]	-45.8 [-50.3, -41.5]
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	119.4 [117.2, 121.6]	121.0 [120.2, 121.9]	11.4 [10.0, 12.8]	-59.5 [-61.0, -57.9]
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	130.1 [128.5, 131.8]	123.8 [123.1, 124.6]	30.9 [29.2, 32.6]	-23.2 [-24.4, -22.0]
SIO, $\xi = 0$	143.1 [142.0, 144.2]	123.3 [122.2, 124.4]	**61.4 [58.1, 64.7]**	-6.4 [-7.7, -5.0]
SIO, $\xi = 0.5$	147.8 [146.6, 148.9]	133.1 [132.4, 133.7]	31.6 [28.7, 34.4]	-2.8 [-3.8, -1.7]
AMSIO, $\xi = 0.5$	**154.6 [153.5, 155.8]**	135.6 [135.1, 136.2]	21.2 [19.0, 23.5]	-0.5 [-1.6, 0.6]
ISSIO, $\xi = 0.5$	153.9 [152.8, 155.0]	**136.2 [135.7, 136.7]**	34.4 [31.3, 37.7]	**1.4 [0.3, 2.4]**

Table 8: Table 1 with all LRs set to 10^{-3}. This further includes two new expert sets, `M` and `2M`: one "mediocre" expert and two "mediocre" experts. Each "mediocre" expert uses the "great" expert's training configuration, with different seeds, but is stopped roughly halfway up to its peak evaluation returns.

Experts	MaxAgg-SI $\lambda_{\text{MA}} = 0.95$	MaxAgg-NoSI $\lambda_{\text{MA}} = 0.95$	SIO $\xi = 0.5$	AMSIO $\xi = 0.5$	ISSIO $\xi = 0.5$
`R`	67.1 [62.7, 71.0]	60.4 [58.1, 62.5]	72.7 [66.4, 79.2]	**81.3 [74.5, 88.6]**	76.8 [69.4, 84.2]
`2R`	66.6 [62.5, 70.7]	57.6 [51.9, 61.5]	72.5 [66.6, 78.6]	**76.4 [70.6, 82.8]**	75.3 [68.9, 82.3]
`8R`	65.1 [60.3, 69.3]	53.6 [47.9, 58.6]	72.1 [67.4, 77.9]	**74.8 [68.2, 81.8]**	65.0 [62.9, 67.1]
`M`	85.7 [79.0, 91.9]	98.8 [91.8, 104.8]	102.0 [97.9, 106.0]	102.0 [98.9, 104.9]	**106.8 [102.5, 110.8]**
`2M`	76.5 [67.1, 85.0]	97.9 [95.0, 100.8]	112.3 [109.6, 114.9]	107.7 [103.7, 111.2]	**113.2 [110.3, 115.9]**
`G`	78.9 [68.5, 88.5]	86.2 [80.2, 92.1]	117.6 [114.2, 120.9]	108.2 [105.3, 111.0]	**118.0 [115.6, 120.5]**
`G2R`	87.5 [80.2, 94.4]	87.7 [79.9, 94.8]	**99.9 [91.9, 107.5]**	93.4 [85.5, 101.7]	98.8 [92.2, 105.1]
`G7R`	84.0 [78.3, 89.5]	78.8 [72.2, 84.7]	88.5 [81.8, 94.9]	87.4 [80.7, 93.6]	**98.6 [95.7, 101.6]**
`Comp1`	70.8 [63.4, 77.0]	89.7 [82.8, 96.3]	102.5 [95.0, 109.7]	97.5 [90.9, 102.9]	**105.8 [98.3, 112.3]**
`Comp2`	71.0 [66.4, 76.3]	71.1 [62.4, 79.7]	98.9 [93.0, 104.7]	100.4 [97.6, 102.7]	**107.4 [101.1, 112.4]**
`Comp3`	75.5 [69.8, 82.0]	94.2 [86.1, 101.5]	101.7 [95.4, 107.0]	93.3 [87.4, 98.7]	**106.0 [100.9, 110.2]**
`Comp4`	62.1 [56.6, 66.1]	62.8 [57.1, 68.1]	74.1 [67.1, 81.6]	**81.3 [75.7, 87.4]**	77.1 [70.4, 84.0]

Table 9: Table 6 with all LRs set to 10^{-3}.

Algorithm	Mean rank ↓	Beats MaxAgg-NoSI ↑	Beats MaxAgg-SI ↑
PPO	5.83 [5.60, 6.00]	0.56 [0.52, 0.62]	0.75 [0.67, 0.81]
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	7.00 [6.80, 7.24]	0.42 [0.35, 0.48]	—
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	6.71 [6.42, 6.84]	—	0.54 [0.46, 0.60]
SIO, $\xi = 0$	5.06 [4.77, 5.33]	0.69 [0.62, 0.75]	0.65 [0.60, 0.73]
SIO, $\xi = 0.5$	4.20 [3.83, 4.44]	0.81 [0.75, 0.85]	0.79 [0.73, 0.85]
AMSIO, $\xi = 0$	4.77 [4.43, 5.02]	**0.83 [0.73, 0.85]**	0.79 [0.73, 0.83]
AMSIO, $\xi = 0.5$	4.12 [3.83, 4.38]	0.77 [0.71, 0.83]	**0.88 [0.83, 0.94]**
ISSIO, $\xi = 0$	4.20 [4.04, 4.65]	0.77 [0.69, 0.79]	0.77 [0.69, 0.77]
ISSIO, $\xi = 0.5$	**3.10 [2.90, 3.44]**	**0.83 [0.79, 0.85]**	0.81 [0.79, 0.90]

Table 10: Table 7 with all LRs set to 10^{-3}.

Algorithm	`Acrobot-v1`	`CartPole-v1`	`MountainCar-v0`	`PendulumDiscrete-v1`
PPO	117.6 [113.5, 120.9]	123.2 [122.4, 123.9]	9.8 [5.1, 15.1]	59.1 [52.4, 65.2]
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	110.6 [106.5, 113.9]	121.7 [121.3, 122.1]	26.8 [22.0, 31.7]	37.8 [33.7, 41.8]
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	96.5 [95.1, 97.8]	115.0 [112.8, 117.2]	44.8 [39.8, 49.7]	56.7 [52.3, 60.9]
SIO, $\xi = 0$	101.1 [98.8, 103.4]	117.6 [114.6, 120.4]	56.8 [51.3, 62.8]	57.7 [53.7, 61.6]
SIO, $\xi = 0.5$	118.1 [116.9, 119.3]	122.1 [121.3, 122.8]	67.6 [61.6, 73.7]	**63.7 [61.0, 66.4]**
AMSIO, $\xi = 0.5$	121.7 [121.2, 122.1]	**124.5 [124.1, 125.0]**	69.7 [64.2, 75.2]	52.0 [48.6, 55.2]
ISSIO, $\xi = 0.5$	**121.7 [120.7, 122.6]**	123.2 [122.4, 123.9]	**75.6 [70.1, 81.0]**	62.5 [59.9, 65.1]

D ALGORITHM RUNTIMES

Table 11: Mean wallclock time for training runs from Table 9, averaged over seeds for each {algorithm, environment, experts} combination, then over all of those combinations per algorithm.

Algorithm	Runtime
PPO	6.2 min
MaxAgg-SI, $\lambda_{\mathrm{MA}} = 0.95$	11.7 min
MaxAgg-NoSI, $\lambda_{\mathrm{MA}} = 0.95$	11.0 min
SIO, $\xi = 0$	10.0 min
SIO, $\xi = 0.5$	10.4 min
AMSIO, $\xi = 0.5$	10.1 min
ISSIO, $\xi = 0.5$	10.5 min

E FROZEN SELECTOR, RANDOM EXPLORATION

This section helps answer how much of SIO's performance comes from (1) the learned selector, and from (2) random exploration from random experts.

Regarding the learned selector, also see our frozen selector experiments on our continuous-action environments, Section G.

We investigate random exploration in this section partly because, empirically, algorithms that use the experts occasionally learn faster than PPO, even when the experts are all random. This is visible in Section K. We speculate this is because the random experts cause more exploration. Note that, for simplicity, we use no entropy coefficient in our algorithms and our baselines.

All results in this section — Tables 12 to 14 — set all LRs to 10^{-3}. These results collectively suggest that the learnable selector is more important than the exploration from random experts, even though learning the selector does not always help. The continuous-action frozen selector results, which we show later (Section G), further suggest that learning the selector can sometimes greatly increase returns.

To ablate (1) the learned selector, SIO-F freezes the selector policy at its initialization. In our tables below, SIO with $\xi = 0.5$ is typically better than SIO-F, though SIO-F is typically better when most of the experts are great or near great.

To ablate (2) the random exploration from random experts, PPO-RR is PPO using random roll-ins, i.e. roll-ins from a random expert during training. In our tables below, the random-expert roll-in only decreases mean returns. (To ensure the policy gets a chance to learn on `CartPole`, which often terminates within e.g. 10 steps, the random roll-in is skipped on half of trajectories. For the other half, we sample roll-in length uniformly from the environment horizon.)

Another option to ablate the random exploration would be adding random actions throughout training, but these actions would be off-policy. SIO would have an unfair advantage in that it never uses its RL component to learn in an off-policy way.

However, we also emphasize that SIO (and AMSIO and ISSIO) tends to match or outperform MaxAgg even when there are zero random experts (e.g. `G` and `M`), as shown in our many other tables (especially our continuous-action results).

Table 12: Table 8 but with only SIO plus our frozen-selector ablation (SIO-F). PPO does not use experts, and PPO with random roll-ins (PPO-RR) always uses only the random expert (for roll-ins): their means and CIs are (respectively) 77.4 [75.2, 79.6] and 72.1 [68.3, 75.8].

Experts	SIO-F $\xi = 0$	SIO-F $\xi = 0.5$	SIO $\xi = 0$	SIO $\xi = 0.5$
R	24.6 [23.3, 25.9]	55.2 [53.5, 56.9]	54.9 [46.5, 63.6]	**72.7 [66.4, 79.2]**
2R	16.9 [15.4, 18.5]	43.5 [41.2, 45.9]	43.9 [38.2, 49.9]	**72.5 [66.5, 78.8]**
8R	-1.5 [-3.0, 0.1]	29.0 [27.2, 30.6]	13.7 [4.6, 22.4]	**72.1 [67.3, 77.9]**
M	83.7 [83.2, 84.1]	87.5 [86.6, 88.4]	98.0 [93.6, 102.5]	**102.0 [97.9, 106.1]**
2M	92.1 [91.4, 92.7]	99.9 [99.3, 100.7]	107.3 [104.0, 110.7]	**112.3 [109.6, 114.9]**
G	**127.3 [126.7, 127.8]**	124.7 [123.7, 125.7]	124.1 [121.0, 126.3]	117.6 [114.4, 120.9]
G2R	93.7 [93.2, 94.2]	96.2 [95.0, 97.5]	93.1 [83.7, 103.5]	**99.9 [91.9, 107.7]**
G7R	63.6 [60.7, 66.3]	72.7 [71.3, 74.2]	86.7 [76.5, 97.9]	**88.5 [81.8, 94.9]**
Comp1	**115.5 [113.7, 117.3]**	109.8 [108.3, 111.3]	102.7 [95.6, 109.7]	102.5 [95.2, 109.7]
Comp2	**114.9 [113.2, 116.5]**	109.9 [108.3, 111.4]	101.4 [94.6, 108.6]	98.9 [93.1, 104.7]
Comp3	**116.0 [115.2, 116.9]**	108.8 [106.2, 111.4]	113.0 [110.3, 115.5]	101.7 [95.4, 107.1]
Comp4	65.3 [61.7, 68.7]	70.8 [68.1, 73.4]	60.8 [57.5, 64.1]	**74.1 [67.1, 81.6]**

Table 13: Table 9 for PPO, PPO-RR, SIO-F, and SIO. MaxAgg-SI and MaxAgg-NoSI remain among the ranked algorithms, as the reference methods for the two win-rate columns; their mean ranks differ from Table 9 because the set of ranked algorithms differs.

Algorithm	Mean rank ↓	Beats MaxAgg-NoSI ↑	Beats MaxAgg-SI ↑
PPO	3.96 [3.92, 4.27]	0.56 [0.52, 0.62]	0.75 [0.67, 0.81]
PPO-RR	5.42 [5.16, 5.61]	0.52 [0.42, 0.54]	0.62 [0.46, 0.67]
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	5.60 [5.34, 5.78]	0.42 [0.35, 0.48]	—
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	5.23 [5.02, 5.40]	—	0.54 [0.46, 0.60]
SIO-F, $\xi = 0$	4.71 [4.58, 4.86]	0.56 [0.52, 0.60]	0.60 [0.56, 0.62]
SIO-F, $\xi = 0.5$	4.25 [4.02, 4.33]	0.65 [0.58, 0.71]	0.62 [0.58, 0.67]
SIO, $\xi = 0$	3.94 [3.69, 4.12]	0.69 [0.62, 0.75]	0.65 [0.60, 0.73]
SIO, $\xi = 0.5$	**2.90 [2.75, 3.12]**	**0.81 [0.75, 0.85]**	**0.79 [0.73, 0.85]**

Table 14: Table 10 for PPO, PPO-RR, SIO, and SIO-F.

Algorithm	Acrobot-v1	CartPole-v1	MountainCar-v0	PendulumDiscrete-v1
PPO	117.6 [113.6, 120.8]	**123.2 [122.4, 123.9]**	9.8 [5.0, 15.2]	59.1 [52.5, 65.1]
PPO-RR	115.8 [108.0, 121.6]	121.9 [120.4, 123.2]	2.3 [0.0, 6.7]	48.6 [35.6, 60.7]
SIO-F, $\xi = 0$	94.6 [94.0, 95.1]	105.7 [105.1, 106.4]	57.8 [56.9, 58.7]	45.9 [44.3, 47.3]
SIO-F, $\xi = 0.5$	100.8 [100.2, 101.4]	118.5 [118.2, 118.9]	55.8 [54.6, 57.0]	60.9 [59.4, 62.3]
SIO, $\xi = 0$	101.1 [98.7, 103.4]	117.6 [114.7, 120.4]	56.8 [51.4, 62.6]	57.7 [53.8, 61.6]
SIO, $\xi = 0.5$	**118.1 [117.0, 119.3]**	122.1 [121.4, 122.8]	**67.6 [61.4, 73.7]**	**63.7 [60.9, 66.6]**

F Higher-Dimensional, Continuous-Action Environments

On four continuous-action environments (Todorov et al., 2012; Brockman et al., 2016), we run the same algorithms as in the discrete-action environments, with all LRs set to 10^{-3}. These environments also have higher dimensionality than the discrete-action environments we test. Table 15 shows that, like the discrete-action environments, SIO and its variants tend to get higher returns than PPO and MaxAgg.

In Table 2, AMSIO matches or exceeds MaxAgg's returns on all expert sets (averaged over environments). Moreover, SIO, AMSIO, and ISSIO all tend to do far better than MaxAgg and PPO given

expert sets that are collectively good policies. The notable exception is `Comp4`, where PPO instead does better than SIO, AMSIO, and ISSIO, perhaps because `Comp4` has the worst individual policies among the `Comp` expert sets despite having the same best compositional policy.

Further, that table gives more satisfying results for MaxAgg-SI vs. MaxAgg-NoSI than the results from the discrete-action environments, in that, for these results, MaxAgg-SI does far better than MaxAgg-NoSI on collectively bad expert sets (the random expert sets). Explaining why our discrete-action results do not show as much of this effect is an interesting direction for future work (Table 8 shows that with all LRs set to 10^{-3}, MaxAgg-SI indeed gets higher returns than MaxAgg-NoSI on the collectively bad expert sets, though not by a lot, and the pattern is actually reversed for Table 1 where we average over the entire LR grid).

In these results, MaxAgg (regardless of SI) typically does worse than PPO even with good expert sets. Conclusively explaining this is an additional interesting direction for future work, but: in our experiments, MaxAgg-SI is quite sensitive to its LRs, and yet even with tuning does not score highly (Section H); note that unlike PPO, MaxAgg uses many of its training environment steps for expert value function pretraining and (discarded) RIRO roll-ins; and for `HalfCheetah`, Cheng et al. (2020) use BC pretraining from the best expert, whereas we do not. BC pretraining is again an interesting direction for future work, though: we speculate that pretraining from the best expert is less efficient and effective than the selective BC within our algorithms, especially for expert sets like our `Comp` sets, where no single expert is great; BC pretraining introduces at least one hyperparameter (the length of BC pretraining); and a fair comparison arguably would require also testing BC pretraining for PPO and our algorithms.

Table 15: Table 6 for continuous-action environments instead of discrete-action environments, and with all LRs set to 10^{-3}.

Algorithm	Mean rank ↓	Beats MaxAgg-NoSI ↑	Beats MaxAgg-SI ↑
PPO	3.79 [3.75, 3.90]	**1.00 [0.96, 1.00]**	**0.90 [0.83, 0.94]**
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	5.27 [5.15, 5.40]	0.67 [0.58, 0.77]	—
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	6.23 [6.10, 6.35]	—	0.33 [0.23, 0.42]
SIO, $\xi = 0$	3.67 [3.52, 3.77]	0.81 [0.79, 0.85]	0.71 [0.71, 0.73]
SIO, $\xi = 0.5$	3.21 [3.02, 3.27]	0.92 [0.90, 0.96]	0.79 [0.79, 0.81]
AMSIO, $\xi = 0.5$	**2.46 [2.40, 2.62]**	**1.00 [0.98, 1.00]**	0.81 [0.79, 0.83]
ISSIO, $\xi = 0.5$	3.38 [3.29, 3.50]	0.83 [0.79, 0.88]	0.73 [0.71, 0.75]

Table 16: Table 7 for continuous-action environments instead of discrete-action environments, and with all LRs set to 10^{-3}. Normalized so that PPO's returns are 100 (and a uniformly random policy's returns are 0).

Algorithm	`HalfCheetah-v4`	`Hopper-v4`	`Pusher-v4`	`Walker2d-v4`
PPO	100.0 [79.5, 123.1]	100.0 [83.9, 116.7]	100.0 [99.6, 100.4]	100.0 [79.7, 122.6]
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	32.5 [31.1, 33.8]	95.1 [88.4, 101.9]	85.3 [83.6, 86.7]	47.1 [43.7, 50.8]
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	25.5 [23.8, 27.4]	56.8 [53.8, 60.0]	80.3 [79.0, 81.5]	31.8 [30.6, 33.0]
SIO, $\xi = 0$	162.3 [160.5, 164.1]	148.9 [145.3, 151.9]	98.7 [98.6, 98.9]	242.4 [238.9, 245.8]
SIO, $\xi = 0.5$	187.4 [185.3, 189.4]	154.0 [151.9, 156.1]	99.1 [98.9, 99.3]	236.6 [234.1, 239.1]
AMSIO, $\xi = 0.5$	**195.7 [193.2, 198.0]**	**165.6 [162.7, 168.6]**	98.2 [97.9, 98.4]	**253.2 [250.1, 256.2]**
ISSIO, $\xi = 0.5$	150.6 [149.1, 152.1]	146.3 [144.5, 148.0]	**100.5 [100.4, 100.6]**	240.0 [236.0, 244.1]

G Frozen Selector in Continuous-Action Environments

We repeat the frozen-selector ablation from Section E on the continuous-action environments. Table 17 compares SIO-F with SIO at $\xi = 0.5$, using the same fixed-learning-rate setting and PPO normalization as Table 2.

Table 17: Continuous-action analogue of Table 12, restricted to SIO-F and SIO at $\xi = 0.5$. For each environment, average evaluation returns during training are normalized so a uniformly random policy's returns are 0 and PPO's returns are 100, then averaged equally over the four environments. All LRs are 10^{-3}.

Experts	SIO-F $\xi = 0.5$	SIO $\xi = 0.5$
`R`	47.2 [46.3, 48.2]	**78.3 [75.9, 80.6]**
`2R`	41.3 [40.3, 42.3]	**74.3 [71.3, 77.1]**
`8R`	35.9 [35.4, 36.5]	**69.3 [66.0, 72.7]**
`M`	195.9 [193.6, 198.3]	**196.8 [193.9, 199.9]**
`2M`	98.0 [97.7, 98.4]	**146.3 [141.7, 151.4]**
`G`	**251.6 [249.0, 254.2]**	250.4 [248.5, 252.6]
`G2R`	83.5 [81.9, 85.1]	**242.9 [239.6, 246.3]**
`G7R`	45.4 [44.6, 46.2]	**213.8 [210.6, 216.7]**
`Comp1`	176.6 [173.2, 179.7]	**241.5 [239.1, 244.0]**
`Comp2`	177.5 [173.1, 182.2]	**241.2 [238.5, 243.8]**
`Comp3`	177.9 [173.5, 182.4]	**208.6 [203.3, 214.1]**
`Comp4`	46.7 [45.9, 47.5]	**68.0 [64.9, 70.8]**

H Continuous-Action Learning-Rate Sensitivity

Table 18: A focused learning-rate sensitivity analysis: MaxAgg-SI on `HalfCheetah-v4`, the "great" expert (`G`), PPO-normalized returns like Table 2. Even with tuning, MaxAgg-SI's best setting gives worse returns (66.1 [58.1, 74.2]) than our untuned algorithms (SIO: 258.8 [257.0, 260.3]; AMSIO: 259.6 [257.7, 261.4]; ISSIO: 244.6 [243.1, 246.0]).

Actor LR	Critic LR					
	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
10^{-6}	14.4 [14.4, 14.4]	14.4 [14.4, 14.4]	14.4 [14.4, 14.4]	14.4 [14.4, 14.4]	14.4 [14.4, 14.4]	14.4 [14.4, 14.4]
10^{-5}	13.1 [11.7, 14.4]	14.2 [14.1, 14.4]	14.1 [14.0, 14.1]	14.1 [14.1, 14.2]	13.7 [13.6, 13.8]	13.1 [12.8, 13.4]
10^{-4}	5.1 [0.2, 9.6]	31.2 [23.7, 38.8]	48.2 [38.8, 57.6]	35.3 [29.3, 41.4]	**66.1 [58.1, 74.2]**	36.9 [30.5, 44.2]
10^{-3}	6.4 [3.3, 10.5]	10.5 [6.8, 14.7]	20.3 [15.7, 24.6]	20.1 [16.7, 23.2]	27.7 [22.3, 33.0]	32.9 [30.4, 35.5]
10^{-2}	-11.2 [-12.6, -9.7]	-10.5 [-12.3, -8.5]	-10.7 [-12.7, -8.7]	-10.2 [-12.2, -8.1]	-3.6 [-7.8, 2.2]	9.2 [4.8, 13.5]
10^{-1}	-17.9 [-19.3, -16.5]	-20.0 [-23.0, -17.5]	-19.6 [-23.0, -16.8]	-18.2 [-19.7, -16.8]	-18.8 [-21.7, -16.6]	-19.8 [-21.8, -17.7]

Table 19: A broader learning-rate sensitivity analysis: MaxAgg-SI and AMSIO on all four continuous-action environments, expert sets `R`, `M`, `G`, and `Comp1` with the best three actor and critic LRs inferred from our prior experiments (actor LRs $10^{-5}, 10^{-4}, 10^{-3}$; critic LRs $10^{-3}, 10^{-2}, 10^{-1}$; ten seeds). Averaging over the environments and expert sets, the below table shows that AMSIO's (PPO-normalized) return CIs exceed MaxAgg-SI's CIs — both when averaging over all searched LRs, and when instead choosing the single best actor and critic LR pair per algorithm. (The single best LR pairs per algorithm are: MaxAgg-SI: actor LR 10^{-4}, critic LR 10^{-2}; AMSIO: actor LR 10^{-4}, critic LR 10^{-3}.)

Algorithm	Default $(10^{-3}, 10^{-3})$	LR-grid mean	Best observed shared LR
MaxAgg-SI, $\lambda_{\mathrm{MA}} = 0.95$	61.9 [58.8, 65.3]	59.1 [58.4, 59.8]	82.4 [80.0, 84.7]
AMSIO, $\xi = 0.5$	**195.6 [193.7, 197.5]**	**159.3 [158.6, 160.0]**	**214.3 [211.3, 217.1]**

Table 20: To alleviate concerns about outliers, this table compares the LR sensitivity experiment data by the same protocol as Table 6. AMSIO's mean returns exceed MaxAgg-SI's in all {environment, experts} combinations.

Algorithm	Beats MaxAgg-SI ↑
MaxAgg-SI, $\lambda_{\mathrm{MA}} = 0.95$	—
AMSIO, $\xi = 0.5$	**1.00 [1.00, 1.00]**

I Omitted Proofs

We analyze the exact population updates over unrestricted tabular policies on finite state and action spaces. The parameter $\eta > 0$ is fixed across rounds and shared by the selector and the student. As in the main text, $|A^\rho(s,a)| \le L$ for all policies ρ, states s, and actions a. The population objectives are separable across states.

All occupancy measures below use the fixed initial distribution d_0. Recall that $J(\rho) = (1-\gamma)\mathbb{E}_{s\sim d_0}[V^\rho(s)]$ and that d^ρ is the normalized discounted occupancy. Thus the performance-difference identity takes the form

$$J(\alpha) - J(\omega) = \mathbb{E}_{s\sim d^\alpha}\left[\langle \alpha(\cdot \mid s), A^\omega(s,\cdot)\rangle\right]. \tag{3}$$

We also use $\langle \omega(\cdot \mid s), A^\omega(s,\cdot)\rangle = 0$.

I.1 The Regret of the Behavior Policy

Proof of Theorem 1. By the performance-difference identity and the comparator decomposition,

$$\begin{aligned}
J(\mu) - J(\mu_t) &= \mathbb{E}_{s\sim d^\mu}\left[\langle \mu(\cdot \mid s) - \mu_t(\cdot \mid s), A^{\mu_t}(s,\cdot)\rangle\right] \\
&= \mathbb{E}_{s\sim d^\mu}\Bigg[\sum_{k=1}^{K}\big(p_\mu(k \mid s) - p_t(k \mid s)\big)\left\langle \pi^{(k)}(\cdot \mid s), A^{\mu_t}(s,\cdot)\right\rangle \\
&\qquad + \big(p_\mu(K+1 \mid s) - p_t(K+1 \mid s)\big)\langle \pi_t(\cdot \mid s), A^{\mu_t}(s,\cdot)\rangle \\
&\qquad + p_\mu(K+1 \mid s)\langle \mu_\perp(\cdot \mid s) - \pi_t(\cdot \mid s), A^{\mu_t}(s,\cdot)\rangle\Bigg] \\
&= \mathbb{E}_{s\sim d^\mu}\left[\sum_{k=1}^{K+1}\big(p_\mu(k \mid s) - p_t(k \mid s)\big)A_t^{\mu_t}(s,k)\right] \\
&\qquad + \mathbb{E}_{s\sim d^\mu}\left[p_\mu(K+1 \mid s)\langle \mu_\perp(\cdot \mid s) - \pi_t(\cdot \mid s), A^{\mu_t}(s,\cdot)\rangle\right].
\end{aligned}$$

For each state, the selector update is

$$p_{t+1}(\cdot \mid s) = \operatorname*{argmax}_{p\in\Delta_{K+1}}\left\{\sum_{k=1}^{K+1} p(k)A_t^{\mu_t}(s,k) - \eta^{-1}D_{\mathrm{KL}}(p\|p_t(\cdot \mid s))\right\}.$$

Since $|A_t^{\mu_t}(s,k)| \le L$, applying Lemma 3 with $\tau = 0$ gives

$$\begin{aligned}
&\sum_{k=1}^{K+1}\big(p_\mu(k \mid s) - p_t(k \mid s)\big)A_t^{\mu_t}(s,k) \\
&\qquad \le 2L^2\eta + \eta^{-1}\Big(D_{\mathrm{KL}}(p_\mu(\cdot \mid s)\|p_t(\cdot \mid s)) - D_{\mathrm{KL}}(p_\mu(\cdot \mid s)\|p_{t+1}(\cdot \mid s))\Big).
\end{aligned}$$

Uniform initialization gives $D_{\mathrm{KL}}(p_\mu(\cdot \mid s)\|p_1(\cdot \mid s)) \le \log(K+1)$. Summing over rounds, taking expectation under d^μ, and using the definition of $\mathrm{Reg}_{\mathrm{stu}}$ therefore yields

$$\sum_{t=1}^{T}\big(J(\mu) - J(\mu_t)\big) \le 2L^2T\eta + \eta^{-1}\log(K+1) + \mathbb{E}_{s\sim d^\mu}\left[p_\mu(K+1 \mid s)\mathrm{Reg}_{\mathrm{stu}}(\mu_\perp; s)\right].$$

□

I.2 Regret of the Student Policy

Throughout this subsection, assume the initialization and parameter choices in Theorem 2. In particular, $\beta_t(s) = \lambda p_t(K+1 \mid s)$, where $\lambda \geq 0$ is fixed across rounds. Write

$$q_t(s) := p_t(K+1 \mid s), \qquad \tau_t(s) := \lambda(1 - q_t(s)).$$

When $q_t(s) < 1$, define the conditional expert mixture

$$\overline{\mu}_t(\cdot \mid s) := \frac{\sum_{k=1}^{K} p_t(k \mid s)\pi^{(k)}(\cdot \mid s)}{1 - q_t(s)}.$$

When $q_t(s) = 1$, define $\overline{\mu}_t(\cdot \mid s)$ arbitrarily and omit its zero-weight contribution to the objective. Then

$$\mu_t(\cdot \mid s) = (1 - q_t(s))\overline{\mu}_t(\cdot \mid s) + q_t(s)\pi_t(\cdot \mid s).$$

The experts may be arbitrary stochastic policies.

Also, the concentrability constant is defined as below:

Assumption 1. *There exists a constant $C > 0$ such that for any pair of stationary policies ρ and σ, it holds that $d^{\rho}(s) \leq C d^{\sigma}(s)$ on all states s.*

Lemma 1 (Primitive regret of the student update)**.** *For every stationary comparator policy ρ and every state s,*

$$\begin{aligned}\mathrm{Reg}_{\mathrm{stu}}(\rho; s) &:= \sum_{t=1}^{T} \langle \rho(\cdot \mid s) - \pi_t(\cdot \mid s), A^{\mu_t}(s, \cdot)\rangle \\ &\leq 2L^2 T\eta + \eta^{-1} D_{\mathrm{KL}}(\rho(\cdot \mid s)\|\pi_1(\cdot \mid s)) + \sum_{t=1}^{T} \tau_t(s).\end{aligned}$$

Consequently, under uniform student initialization,

$$\mathrm{Reg}_{\mathrm{stu}}(\rho; s) \leq 2L^2 T\eta + \eta^{-1} \log|\mathcal{A}| + \lambda T.$$

Proof. The selector's entropic update preserves full support, so $q_t(s) > 0$ at every finite round. Dividing the statewise student objective by $q_t(s)$, and using $\beta_t(s) = \lambda q_t(s)$, gives

$$\pi_{t+1}(\cdot \mid s) = \operatorname*{argmax}_{\pi' \in \Delta_{\mathcal{A}}} \Big\{ \langle \pi', A^{\mu_t}(s, \cdot)\rangle - \eta^{-1} D_{\mathrm{KL}}(\pi'\|\pi_t(\cdot \mid s)) - \tau_t(s) D_{\mathrm{KL}}(\overline{\mu}_t(\cdot \mid s)\|\pi') \Big\}.$$

Indeed, the behavior-cloning term after division is

$$\lambda \sum_{k=1}^{K} p_t(k \mid s)\mathbb{E}_{a \sim \pi^{(k)}(\cdot|s)}[\log \pi'(a)] = \tau_t(s) \sum_{a \in \mathcal{A}} \overline{\mu}_t(a \mid s) \log \pi'(a),$$

which differs from $-\tau_t(s) D_{\mathrm{KL}}(\overline{\mu}_t(\cdot \mid s)\|\pi')$ only by a term independent of π'.

Apply Lemma 3 with $x_t = \pi_t(\cdot \mid s)$, $y_t = \overline{\mu}_t(\cdot \mid s)$, $u = \rho(\cdot \mid s)$, $r_t = A^{\mu_t}(s, \cdot)$, and $\tau = \tau_t(s)$. The same lemma ensures that the student iterates retain full support. Summing the resulting inequalities telescopes the comparator KL terms. Dropping the final nonnegative KL term proves the first bound. The second uses $D_{\mathrm{KL}}(\rho(\cdot \mid s)\|\pi_1(\cdot \mid s)) \leq \log|\mathcal{A}|$ and $\tau_t(s) \leq \lambda$. □

Lemma 2 (Direct comparison of behavior and student policies)**.** *Let $\mu^\star$ be any optimal stationary policy. For any stationary policies μ and π, define*

$$F(\mu, \pi) := \mathbb{E}_{s \sim d^{\mu^\star}} \left[\langle \mu^\star(\cdot \mid s) - \pi(\cdot \mid s), A^{\mu}(s, \cdot)\rangle \right].$$

Then

$$J(\mu^\star) - J(\pi) \leq C F(\mu, \pi) + C(C-1)\big(J(\mu^\star) - J(\mu)\big).$$

Proof. Set $r := J(\mu^\star) - J(\mu)$ and $e := J(\mu^\star) - J(\pi)$, and write

$$b(s) := \langle \pi(\cdot \mid s), A^\mu(s,\cdot)\rangle, \qquad b_+(s) := \max\{b(s), 0\}, \qquad b_-(s) := \max\{-b(s), 0\}.$$

Let $M_+ := \mathbb{E}_{d^{\mu^\star}}[b_+]$ and $M_- := \mathbb{E}_{d^{\mu^\star}}[b_-]$. The performance-difference identity gives

$$F(\mu,\pi) = r - M_+ + M_-, \qquad e = r - \mathbb{E}_{d^\pi}[b]. \tag{4}$$

To control the positive part, define the stationary policy

$$\nu(\cdot \mid s) := \begin{cases} \pi(\cdot \mid s), & b(s) \geq 0, \\ \mu(\cdot \mid s), & b(s) < 0. \end{cases}$$

Since $\langle \mu(\cdot \mid s), A^\mu(s,\cdot)\rangle = 0$, $\langle \nu(\cdot \mid s), A^\mu(s,\cdot)\rangle = b_+(s)$. Optimality of $\mu^\star$ therefore implies

$$\mathbb{E}_{d^\nu}[b_+] = J(\nu) - J(\mu) \leq J(\mu^\star) - J(\mu) = r.$$

Using concentrability on the nonnegative function b_+ gives

$$M_+ \leq C\mathbb{E}_{d^\nu}[b_+] \leq Cr. \tag{5}$$

Also, applying concentrability separately to b_+ and b_- yields

$$\mathbb{E}_{d^\pi}[b_+] \geq C^{-1}M_+, \qquad \mathbb{E}_{d^\pi}[b_-] \leq CM_-.$$

Combining these inequalities with (4) gives

$$\begin{aligned} e &\leq r - C^{-1}M_+ + CM_- \\ &= CF(\mu,\pi) + (1-C)r + (C - C^{-1})M_+ \\ &\leq CF(\mu,\pi) + C(C-1)r, \end{aligned}$$

where the last step uses (5) and $C \geq 1$. No sign assumption on $F(\mu,\pi)$ is used. □

Proof of Theorem 2. Define

$$B_p(T) := 2L^2T\eta + \eta^{-1}\log(K+1), \qquad B_\pi(T) := 2L^2T\eta + \eta^{-1}\log|\mathcal{A}|,$$

and let $F_t := F(\mu_t, \pi_t)$. By Lemma 1,

$$\sum_{t=1}^T F_t = \mathbb{E}_{s\sim d^{\mu^\star}}[\mathrm{Reg}_{\mathrm{stu}}(\mu^\star; s)] \leq B_\pi(T) + \lambda T. \tag{6}$$

Next, apply Theorem 1 to the comparator decomposition $p_{\mu^\star}(K+1 \mid s) = 1$, with residual policy $\mu^\star$. This gives

$$R_{\mathrm{beh}}(T) := \sum_{t=1}^T \big(J(\mu^\star) - J(\mu_t)\big) \leq B_p(T) + \sum_{t=1}^T F_t. \tag{7}$$

Finally, summing Lemma 2 and using $C(C-1) \geq 0$, we obtain

$$\begin{aligned} \sum_{t=1}^T \big(J(\mu^\star) - J(\pi_t)\big) &\leq C\sum_{t=1}^T F_t + C(C-1)R_{\mathrm{beh}}(T) \\ &\leq C^2 \sum_{t=1}^T F_t + C(C-1)B_p(T) \\ &\leq C^2\big(B_\pi(T) + \lambda T\big) + C(C-1)B_p(T) \\ &\leq C^2\big(B_p(T) + B_\pi(T) + \lambda T\big). \end{aligned}$$

With $\lambda = T^{-1/2}$ and $\eta = \sqrt{\log((K+1)|\mathcal{A}|)/T}/L$, the last expression is bounded by

$$O\left(C^2 L\sqrt{T\log((K+1)|\mathcal{A}|)}\right).$$

□

I.3 Auxiliary Lemmas

Lemma 3 (General ascent lemma)**.** *Let $m \in \mathbb{N}$, let $x_t \in \Delta_m$ have full support, and let $y_t \in \Delta_m$, $r_t \in [-L, L]^m$, $\eta > 0$, and $\tau \geq 0$. Suppose*

$$x_{t+1} = \operatorname*{argmax}_{x \in \Delta_m} \left\{ \langle x, r_t \rangle - \eta^{-1} D_{\mathrm{KL}}(x \| x_t) - \tau D_{\mathrm{KL}}(y_t \| x) \right\},$$

where the last term is omitted when $\tau = 0$. Then x_{t+1} has full support, and, for every $u \in \Delta_m$,

$$\langle u - x_t, r_t \rangle \leq 2L^2\eta + \eta^{-1}\big(D_{\mathrm{KL}}(u \| x_t) - D_{\mathrm{KL}}(u \| x_{t+1})\big) + \tau.$$

Proof. The objective is strictly concave and has a unique maximizer. This maximizer has full support: a zero coordinate with $\tau y_t(i) > 0$ would give objective value $-\infty$; at a zero coordinate with $\tau y_t(i) = 0$, moving a small amount of mass to that coordinate increases the entropy term at order $\varepsilon \log(1/\varepsilon)$, while changing all other terms by only $O(\varepsilon)$. Thus all derivatives below are well defined.

Write $z = x_{t+1}$. The first-order optimality condition gives

$$0 \geq \langle u - z, r_t \rangle + \eta^{-1}\big(D_{\mathrm{KL}}(u \| z) - D_{\mathrm{KL}}(u \| x_t) + D_{\mathrm{KL}}(z \| x_t)\big) + \tau \left(\sum_{i=1}^{m} \frac{u(i) y_t(i)}{z(i)} - 1 \right).$$

Since every summand $u(i)y_t(i)/z(i)$ is nonnegative,

$$\langle u - x_t, r_t \rangle \leq \eta^{-1}\big(D_{\mathrm{KL}}(u \| x_t) - D_{\mathrm{KL}}(u \| z)\big) + \tau + \langle z - x_t, r_t \rangle - \eta^{-1} D_{\mathrm{KL}}(z \| x_t).$$

By Pinsker's inequality and $\|r_t\|_\infty \leq L$,

$$\begin{aligned} \langle z - x_t, r_t \rangle - \eta^{-1} D_{\mathrm{KL}}(z \| x_t) &\leq L\|z - x_t\|_1 - \frac{1}{2\eta}\|z - x_t\|_1^2 \\ &\leq \frac{\eta L^2}{2} \leq 2L^2\eta. \end{aligned}$$

Combining the last two displays proves the lemma. □

J Further Experiment Details

`PendulumDiscrete-v1` is the classic `Pendulum-v1` environment, but with the continuous torque action space replaced by three discrete actions applying torques -2, 0, and 2.

J.1 Hyperparameters

J.1.1 Discrete-Action Environments

For all algorithms in the discrete-action experiments, we follow the conventions and default hyperparameters of CleanRL (Huang et al., 2022; CleanRL contributors, 2026b), except that: for simplicity we do not include entropy or LR annealing; and for computational throughput, instead of CleanRL's default collection of 128 steps in each of four parallel environments, we use one environment with 512 steps per PPO rollout, preserving the default effective PPO batch size of 512 transitions per update. Further, we note where some results sweep the actor and critic LRs over a fixed grid rather than using the default LRs.

For additional evidence of the performance of our algorithms, below we show results even closer to CleanRL's PPO setup (four parallel environments, 128 steps per PPO rollout, 10^{-3} LRs). The below Table 21, Table 22, and Table 23 show our three best algorithms ($\xi = 0.5$) match or outperform both MaxAgg-NoSI and MaxAgg-SI over almost all environment-averaged or expert-set-averaged results.

Table 21: Table 8 with default hyperparameters (four parallel environments, 128 steps per PPO rollout, and all LRs set to 10^{-3}; does not include `M` and `2M`).

Experts	MaxAgg-SI $\lambda_{\text{MA}} = 0.95$	MaxAgg-NoSI $\lambda_{\text{MA}} = 0.95$	SIO $\xi = 0.5$	AMSIO $\xi = 0.5$	ISSIO $\xi = 0.5$
`R`	47.2 [40.6, 53.9]	58.5 [55.4, 61.2]	62.0 [56.7, 67.3]	**74.2 [68.4, 79.4]**	66.0 [60.8, 71.4]
`2R`	61.8 [58.4, 65.5]	59.0 [55.5, 62.2]	60.5 [55.7, 65.5]	**69.5 [64.7, 74.4]**	58.3 [53.0, 64.0]
`8R`	55.8 [50.7, 60.0]	54.7 [50.7, 58.3]	52.0 [47.5, 56.6]	**64.9 [59.9, 70.3]**	49.2 [44.7, 53.7]
`G`	71.7 [65.2, 77.9]	71.4 [66.0, 77.6]	**139.2 [128.8, 148.1]**	112.9 [102.5, 124.0]	137.2 [127.2, 146.4]
`G2R`	68.5 [61.5, 74.7]	79.2 [73.2, 86.1]	**110.2 [95.3, 125.9]**	88.6 [78.8, 99.9]	107.9 [94.9, 122.6]
`G7R`	71.1 [65.7, 76.7]	79.2 [70.6, 86.7]	**96.8 [81.0, 113.4]**	86.5 [75.1, 100.2]	96.2 [82.9, 110.2]
`Comp1`	59.6 [56.4, 63.0]	71.6 [69.3, 73.8]	**105.7 [99.9, 111.4]**	97.2 [90.1, 104.7]	104.0 [97.9, 110.7]
`Comp2`	57.2 [53.3, 61.3]	77.5 [69.1, 86.4]	96.8 [92.9, 101.0]	91.1 [88.6, 93.8]	**98.9 [92.9, 105.2]**
`Comp3`	59.5 [52.7, 65.2]	77.4 [72.9, 84.2]	**109.7 [101.9, 116.4]**	86.0 [80.6, 91.4]	104.9 [98.4, 111.2]
`Comp4`	51.1 [44.6, 56.6]	66.9 [64.7, 69.0]	69.1 [64.4, 73.4]	**78.2 [73.8, 82.4]**	68.8 [62.7, 74.6]

Table 22: Table 9 with default hyperparameters (four parallel environments, 128 steps per PPO rollout, and all LRs set to 10^{-3}; does not include `M` and `2M`).

Algorithm	Mean rank ↓	Beats MaxAgg-NoSI ↑	Beats MaxAgg-SI ↑
PPO	5.60 [5.33, 5.75]	0.57 [0.53, 0.70]	0.85 [0.78, 0.88]
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	7.70 [7.42, 7.79]	0.30 [0.23, 0.38]	—
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	6.83 [6.58, 7.01]	—	0.57 [0.50, 0.65]
SIO, $\xi = 0$	4.92 [4.60, 5.22]	0.65 [0.62, 0.75]	0.78 [0.72, 0.82]
SIO, $\xi = 0.5$	3.48 [3.25, 3.92]	**0.90 [0.82, 0.90]**	0.90 [0.82, 0.93]
AMSIO, $\xi = 0$	4.65 [4.34, 4.96]	0.78 [0.70, 0.80]	0.82 [0.75, 0.85]
AMSIO, $\xi = 0.5$	4.15 [3.75, 4.28]	0.82 [0.78, 0.88]	**0.97 [0.88, 0.97]**
ISSIO, $\xi = 0$	4.33 [4.22, 4.88]	0.80 [0.70, 0.80]	0.80 [0.72, 0.82]
ISSIO, $\xi = 0.5$	**3.35 [3.05, 3.62]**	**0.90 [0.85, 0.90]**	0.90 [0.85, 0.95]

Table 23: Table 10 with default hyperparameters (four parallel environments, 128 steps per PPO rollout, and all LRs set to 10^{-3}; does not include `M` and `2M`).

Algorithm	`Acrobot-v1`	`CartPole-v1`	`MountainCar-v0`	`PendulumDiscrete-v1`
PPO	**108.9 [106.7, 110.7]**	119.8 [118.3, 121.2]	1.5 [0.0, 3.7]	**64.3 [56.5, 72.9]**
MaxAgg-SI, $\lambda_{\text{MA}} = 0.95$	87.7 [82.6, 92.4]	117.0 [116.3, 117.8]	5.9 [3.0, 9.1]	30.8 [27.1, 34.6]
MaxAgg-NoSI, $\lambda_{\text{MA}} = 0.95$	88.2 [86.8, 89.6]	113.5 [111.2, 115.5]	14.2 [9.2, 19.7]	62.3 [58.3, 65.8]
SIO, $\xi = 0$	85.3 [82.9, 87.6]	114.6 [112.5, 116.7]	68.2 [58.0, 78.9]	58.2 [55.1, 61.4]
SIO, $\xi = 0.5$	102.5 [101.4, 103.6]	122.4 [122.0, 122.7]	**74.3 [64.2, 84.6]**	61.7 [56.7, 66.4]
AMSIO, $\xi = 0.5$	106.7 [106.1, 107.3]	**123.1 [122.8, 123.3]**	55.7 [47.3, 64.3]	54.2 [49.8, 58.7]
ISSIO, $\xi = 0.5$	104.6 [103.7, 105.4]	122.3 [121.9, 122.7]	70.9 [61.4, 80.9]	58.8 [54.5, 63.1]

Early experiments for both MaxAgg and SIO suggested an actor LR grid of $\{10^{-6}, 10^{-4}, 10^{-2}\}$ and a critic LR grid of $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, which we use for all algorithms (including PPO) and all tables (unless otherwise specified). (However, the final results perhaps suggest that a better hyperparameter search space for the actor LR grid for both algorithms might be $\{10^{-5}, 10^{-3}, 10^{-1}\}$.)

In the learning-rate sweep plots, we abbreviate the actor LR as `alr`, and the critic LR as `clr`.

For all MaxAgg-NoSI and MaxAgg-SI results, we use $\lambda_{\text{MA}} = 0.95$, and (following Cheng et al., 2020) $N_{\text{pretrain}} = 16$ (per expert). For all SIO, AMSIO, and ISSIO results other than the β ablation, we use $\beta = 1$. We likewise use $\xi = 0.5$ unless otherwise specified.

J.1.2 HIGHER-DIMENSIONAL, CONTINUOUS-ACTION ENVIRONMENTS

For the higher-dimensional, continuous-action experiments, our experiments even more closely follow CleanRL's continuous-action PPO defaults (Huang et al., 2022). The only defaults we change are: we disable LR annealing, and set all LRs $= 10^{-3}$ instead of the default 3×10^{-4}. Note that, for continuous-action PPO, CleanRL does not use entropy by default, and we again likewise do not use entropy.

J.2 EXPERT CONSTRUCTIONS

J.2.1 DISCRETE-ACTION EXPERTS

`G` (great expert). For each environment, the great expert `G` is a DQN policy trained for 500k environment steps with the best LR (by peak returns) in $\{10^{-5}, 3 \times 10^{-5}, \ldots, 10^{-1}\}$, saving the weights at the peak return. This gave LRs 3×10^{-4} for `Acrobot-v1` and `CartPole-v1`, 3×10^{-3} for `MountainCar-v0`, and 10^{-2} for `PendulumDiscrete-v1`. Apart from the LR, we use CleanRL's defaults (Huang et al., 2022; CleanRL contributors, 2026a). The saved final Q-network acts greedily by taking $\mathrm{argmax}_a\, Q(s, a)$.

`M` (mediocre expert). Each "mediocre" expert uses the "great" expert's training configuration, with different seeds, but is stopped roughly halfway up to its peak evaluation returns. The `2M` setting uses two "mediocre" experts, i.e. two different seeds.

`R` (random expert). The random expert takes a uniformly random action at each step.

Every expert in `Comp1`–`Comp4` uses the same `G` policy, but acts uniformly randomly outside its per-expert great region. Within each set, the great regions partition the state space. We describe them below; values are clamped to the stated ranges before binning.

`Comp1` (position split). A scalar position-related score is computed from the state, and one expert acts randomly where this score is ≥ 0, the other where it is < 0.

- `CartPole-v1`: pole angle.
- `MountainCar-v0`: car position $+ 0.5$ (so the split is at the bottom of the valley, $x = -0.5$).
- `Acrobot-v1`: end-effector horizontal coordinate, $\sin(\theta_1) + \sin(\theta_1 + \theta_2)$.
- `PendulumDiscrete-v1`: $\sin(\theta)$ (horizontal component of the pole tip).

`Comp2` (velocity split). Same ≥ 0 vs. < 0 convention as `Comp1`, but on a velocity coordinate of the state.

- `CartPole-v1`: pole angular velocity.
- `MountainCar-v0`: car velocity.
- `Acrobot-v1`: angular velocity of the second link, $\dot{\theta}_2$.
- `PendulumDiscrete-v1`: angular velocity, $\dot{\theta}$.

`Comp3` (2-class position checkerboard). One or two position-related features are uniformly binned over the listed range, and the parity (sum of bin indices, mod 2) determines which expert is great there. The two experts are great on opposite parities.

- `CartPole-v1`: 2-D, cart position over $[-2.4, 2.4]$ in 20 bins $\times$ pole angle over $[-\pi/15, \pi/15]$ in 20 bins.
- `MountainCar-v0`: 1-D, car position over $[-1.2, 0.6]$ in 101 bins.
- `Acrobot-v1`: 2-D, end-effector $x = \sin(\theta_1) + \sin(\theta_1 + \theta_2)$ over $[-2, 2]$ in 20 bins $\times$ end-effector $y = -\cos(\theta_1) - \cos(\theta_1 + \theta_2)$ over $[-2, 2]$ in 20 bins.
- `PendulumDiscrete-v1`: 1-D, $\sin(\theta)$ over $[-1, 1]$ in 101 bins.

`Comp4` (8-class position checkerboard). Same construction as `Comp3`, but the bin-index sum is taken mod 8, and bin counts are slightly larger to make them divisible by 8 (in retrospect, ideally we

would have varied only the number of classes compared to the 2-class checkerboard, but the number of bins remains close to the 2-class checkerboard). Expert $i \in \{1, \ldots, 8\}$ is great only on class $i - 1$ and acts randomly elsewhere.

- `CartPole-v1`: cart position in 24 bins $\times$ pole angle in 24 bins (same ranges as `Comp3`).
- `MountainCar-v0`: car position over $[-1.2, 0.6]$ in 104 bins.
- `Acrobot-v1`: end-effector x in 24 bins $\times$ end-effector y in 24 bins (same ranges as `Comp3`).
- `PendulumDiscrete-v1`: $\sin(\theta)$ over $[-1, 1]$ in 104 bins.

J.2.2 CONTINUOUS-ACTION EXPERTS

`G` (great expert). For each environment, the great expert `G` is a continuous-action PPO policy trained for 1M environment steps, saving the final weights. The actor LR is 10^{-4} for every environment. The critic LR is 10^{-4} for `Walker2d-v4` and 10^{-2} for `HalfCheetah-v4`, `Hopper-v4`, and `Pusher-v4`. As an expert, the saved policy normalizes observations with its saved statistics and acts deterministically with the mean of its Gaussian, which the environment then clips to the action-space bounds.

`M` (mediocre expert). Each mediocre expert uses the corresponding `G` training configuration, with a different seed, but like the discrete-action `M` is stopped roughly halfway up to its peak evaluation returns: after 300k environment steps for `HalfCheetah-v4` and `Walker2d-v4`, 140k for `Hopper-v4`, and 60k for `Pusher-v4`. The `2M` setting again uses two such experts, i.e. two different seeds.

`R` (random expert). Like the discrete-action case, the random expert takes a uniformly random action at each step.

Like the discrete-action case, every expert in `Comp1`–`Comp4` uses the same `G` policy, but acts uniformly randomly outside its per-expert great region.

`Comp1` (position split). For each environment, we subtract the listed threshold from the listed position-related feature. One expert is great where the result is < 0, the other where it is ≥ 0.

- `HalfCheetah-v4`: root height, split at -0.055.
- `Walker2d-v4`: torso height, split at 1.20.
- `Hopper-v4`: torso height, split at 1.39.
- `Pusher-v4`: object–goal horizontal displacement, $x_{\text{obj}} - x_{\text{goal}}$, split at 0.01.

`Comp2` (velocity split). Same convention as `Comp1`, but on the listed velocity-related feature.

- `HalfCheetah-v4`: horizontal root velocity, split at 5.05.
- `Walker2d-v4`: horizontal root velocity, split at 4.28.
- `Hopper-v4`: horizontal root velocity, split at 2.79.
- `Pusher-v4`: shoulder-pan joint angular velocity, split at 0.004.

`Comp3` (2-class position checkerboard). Two position-related features are each uniformly binned into 20 bins over the listed range, and the parity (sum of bin indices, mod 2) determines which expert is great there. The two experts are great on opposite parities.

- `HalfCheetah-v4`: root height over $[-0.5, 0.5]$ $\times$ root angle over $[-1, 1]$.
- `Walker2d-v4`: torso height over $[0.8, 2]$ $\times$ torso angle over $[-1, 1]$.
- `Hopper-v4`: torso height over $[0.7, 2]$ $\times$ torso angle over $[-1, 1]$.
- `Pusher-v4`: fingertip y-position over $[-0.6, -0.08]$ $\times$ object y-position over $[-0.35, -0.06]$.

`Comp4` (8-class position checkerboard). Same features and ranges as `Comp3`, but with 24 bins each and the bin-index sum taken mod 8. Expert $i \in \{1, \ldots, 8\}$ is great only on class $i - 1$.

K DISCRETE-ACTION LEARNING-RATE SWEEP PLOTS ($\xi = 0.5$)

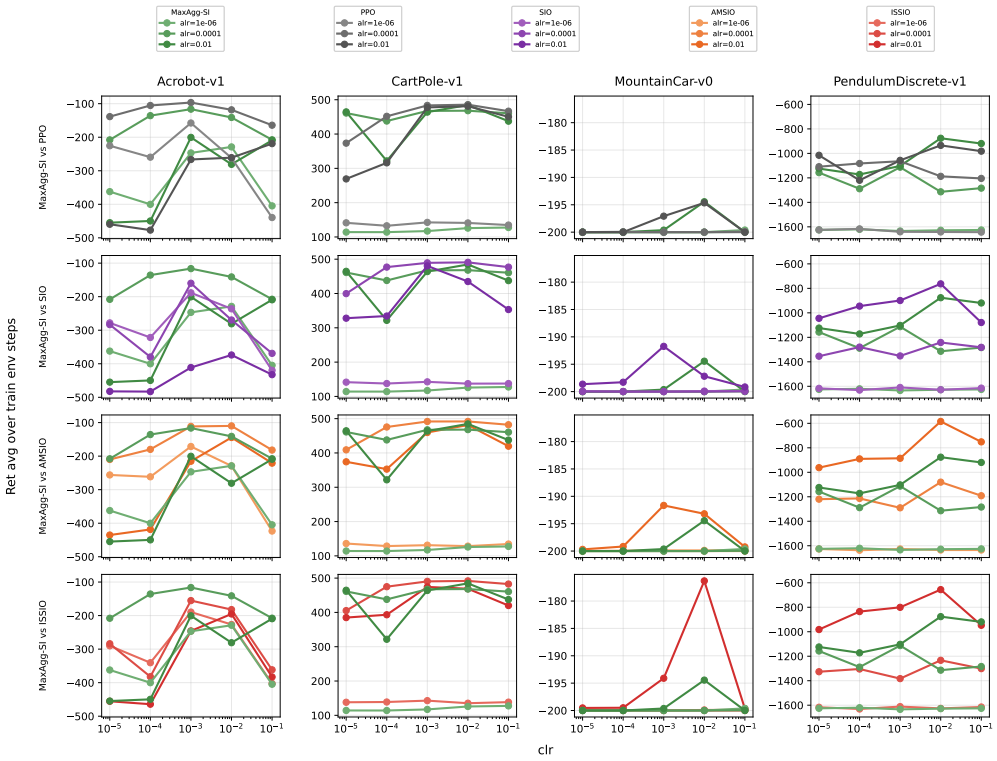

Figure 4: Mean evaluation returns over training for expert set R, with MaxAgg-SI.

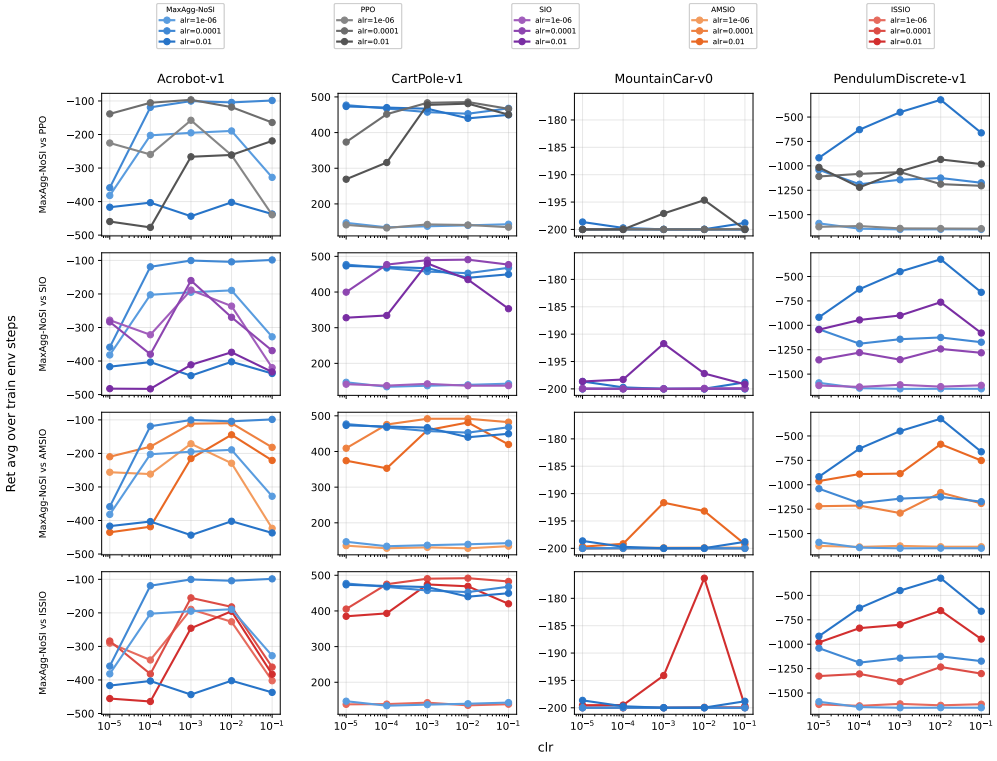

Figure 5: Mean evaluation returns over training for expert set R, with MaxAgg-NoSI.

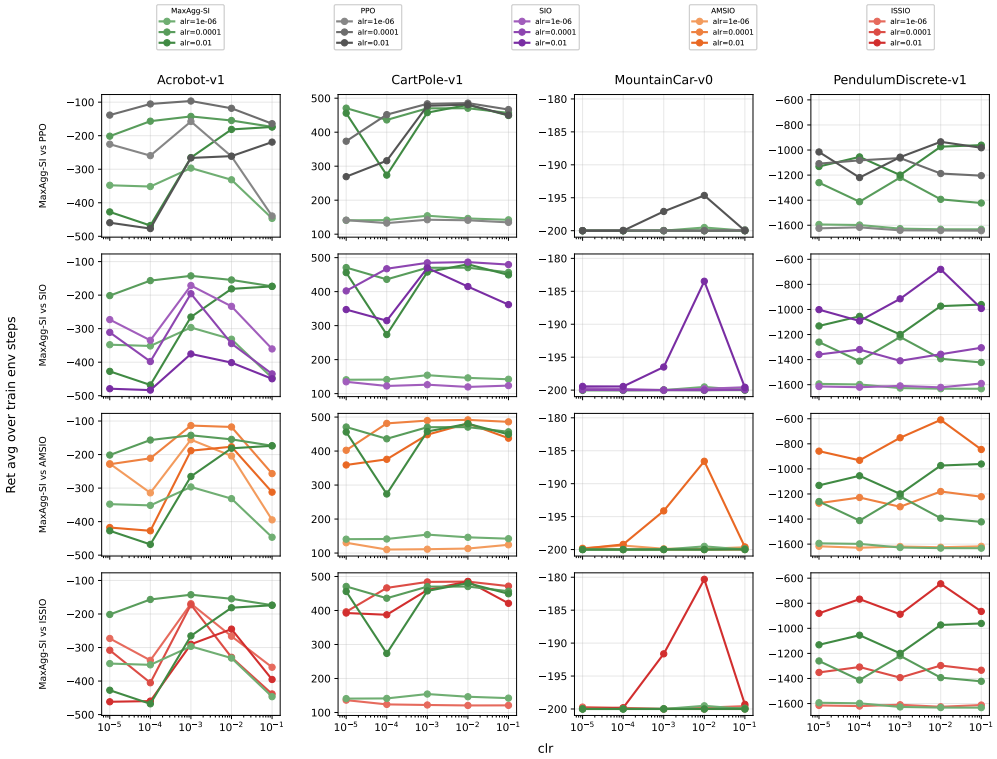

Figure 6: Mean evaluation returns over training for expert set 2R, with MaxAgg-SI.

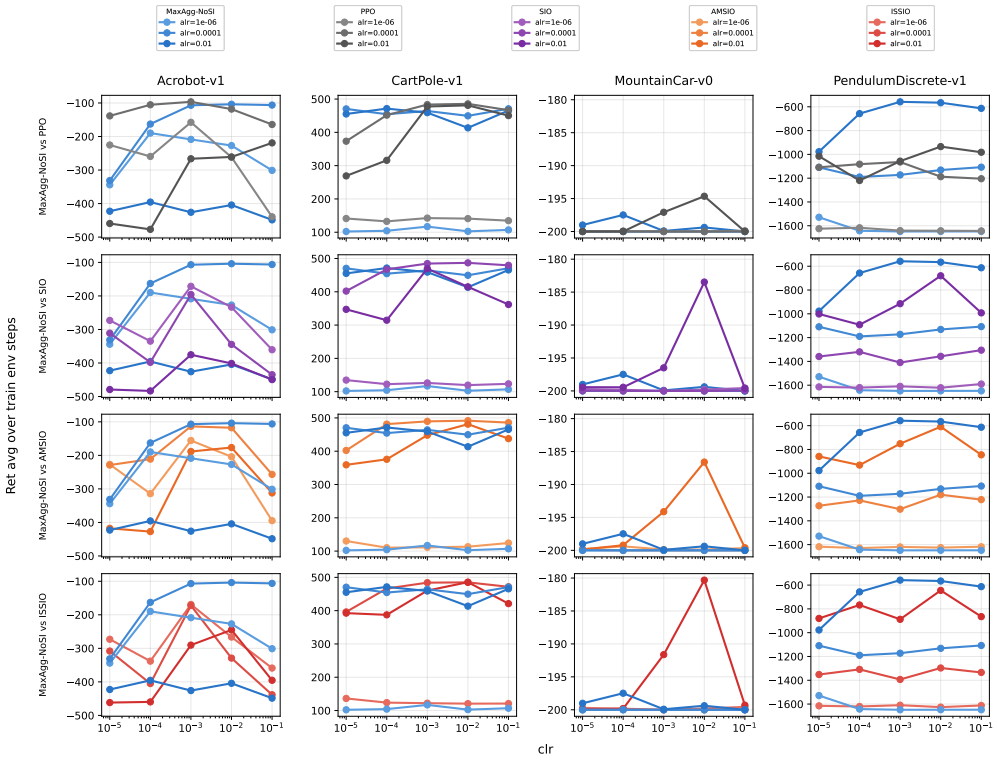

Figure 7: Mean evaluation returns over training for expert set 2R, with MaxAgg-NoSI.



Figure 8: Mean evaluation returns over training for expert set `8R`, with MaxAgg-SI.



Figure 9: Mean evaluation returns over training for expert set `8R`, with MaxAgg-NoSI.

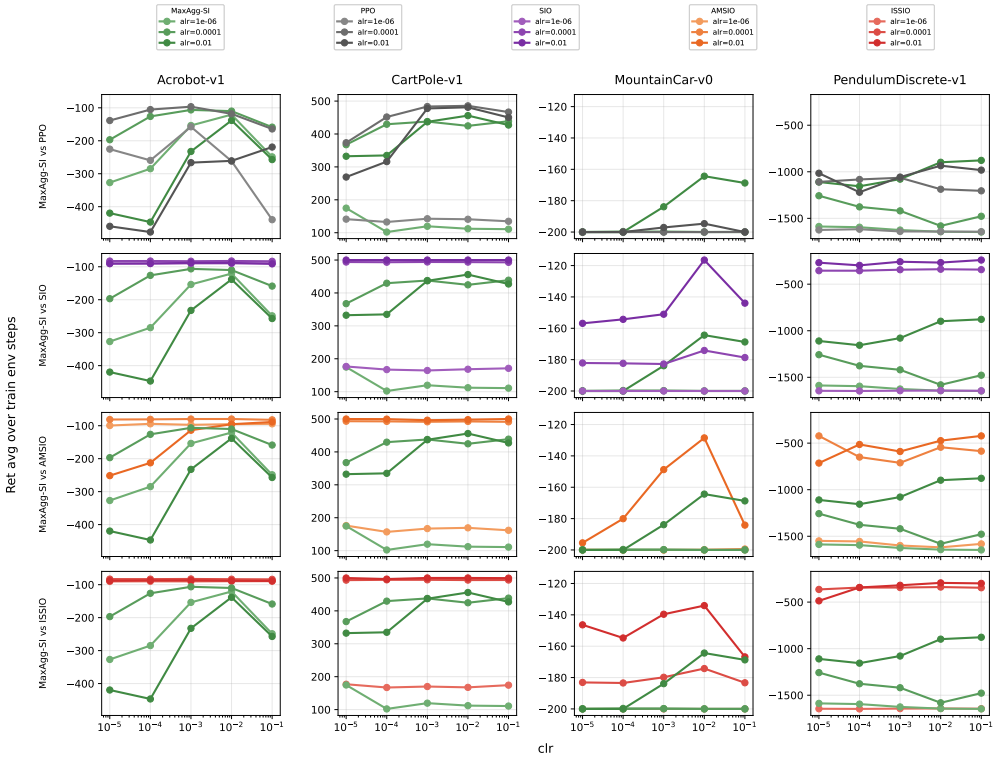

Figure 10: Mean evaluation returns over training for expert set G, with MaxAgg-SI.

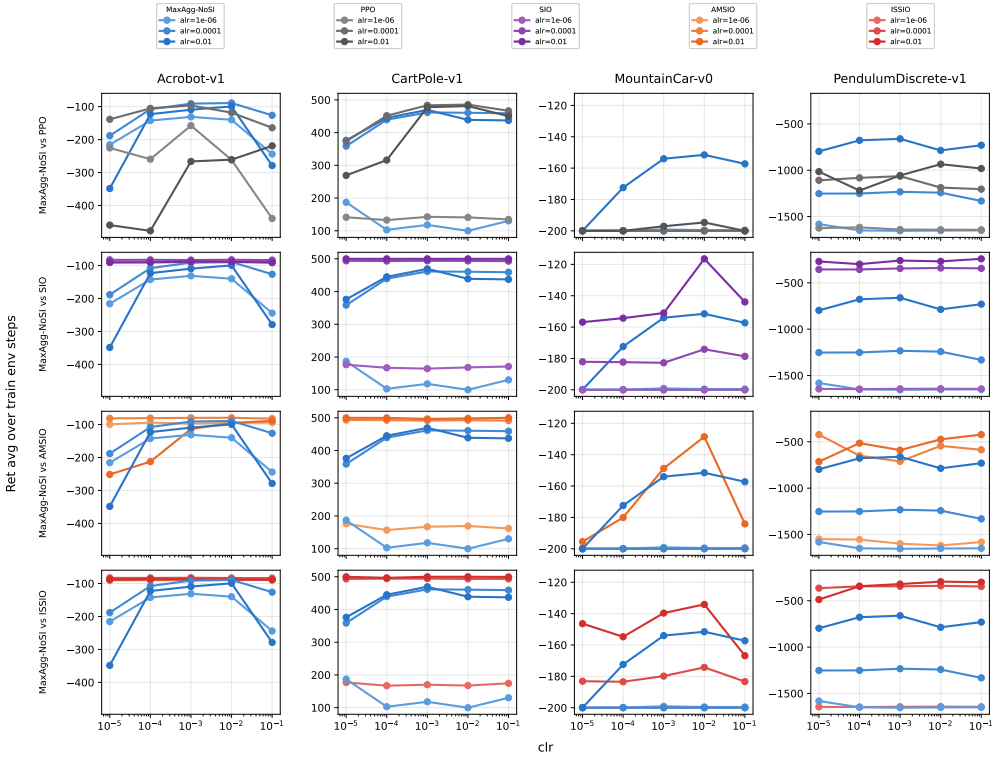

Figure 11: Mean evaluation returns over training for expert set G, with MaxAgg-NoSI.

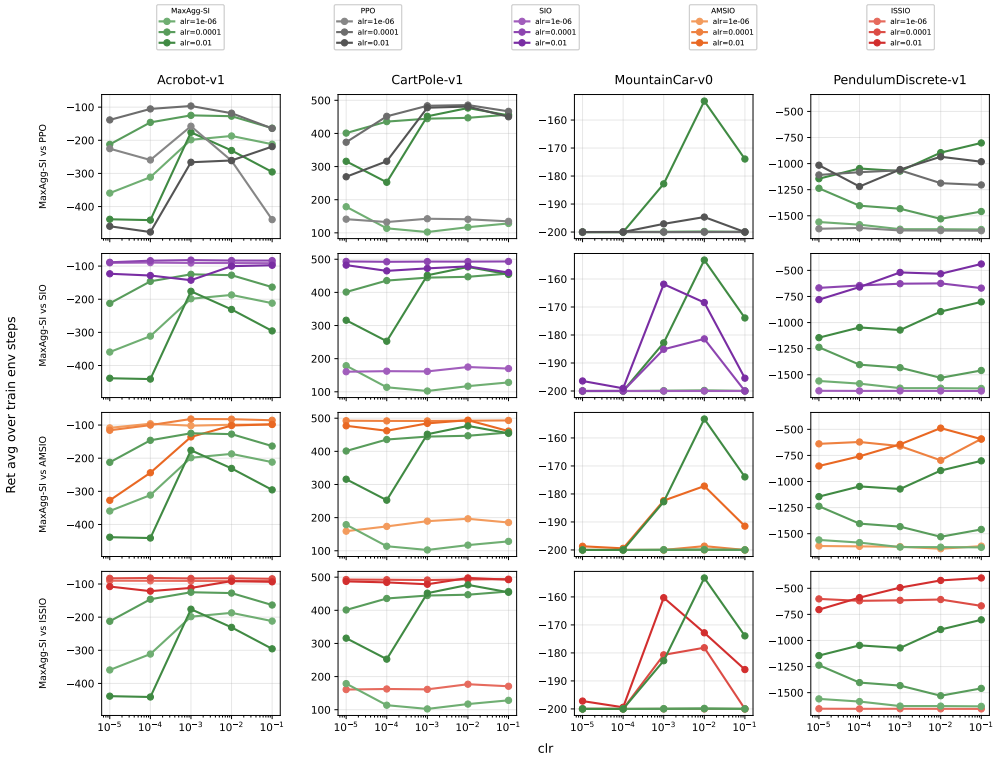

Figure 12: Mean evaluation returns over training for expert set G2R, with MaxAgg-SI.

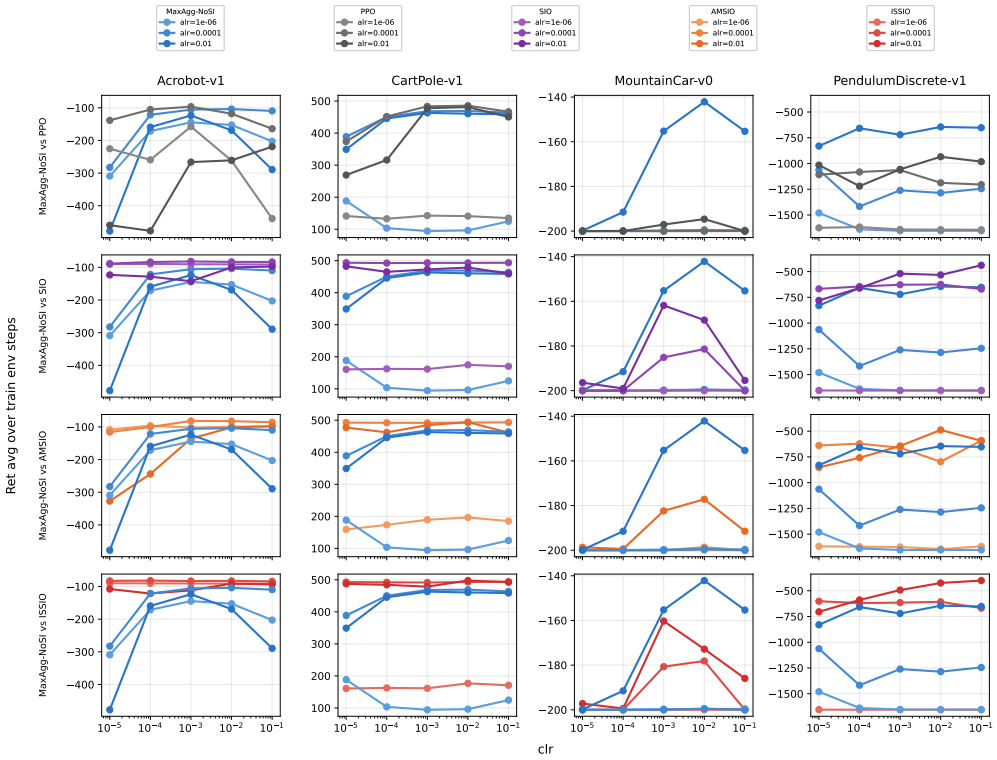

Figure 13: Mean evaluation returns over training for expert set G2R, with MaxAgg-NoSI.



Figure 14: Mean evaluation returns over training for expert set G7R, with MaxAgg-SI.

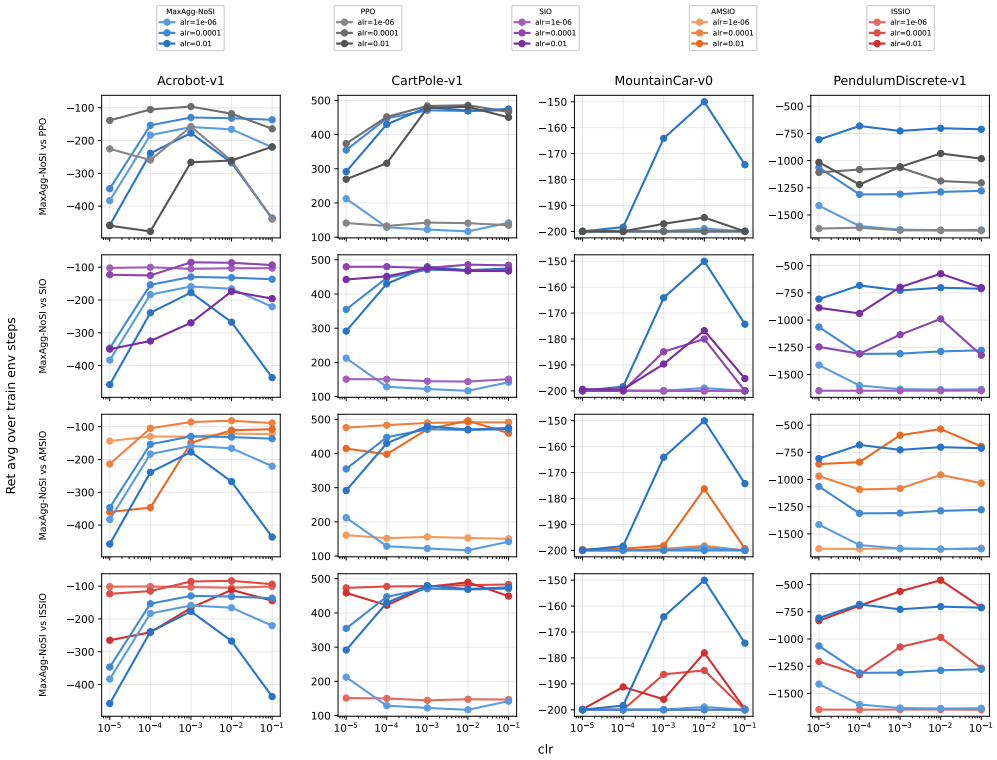

Figure 15: Mean evaluation returns over training for expert set G7R, with MaxAgg-NoSI.

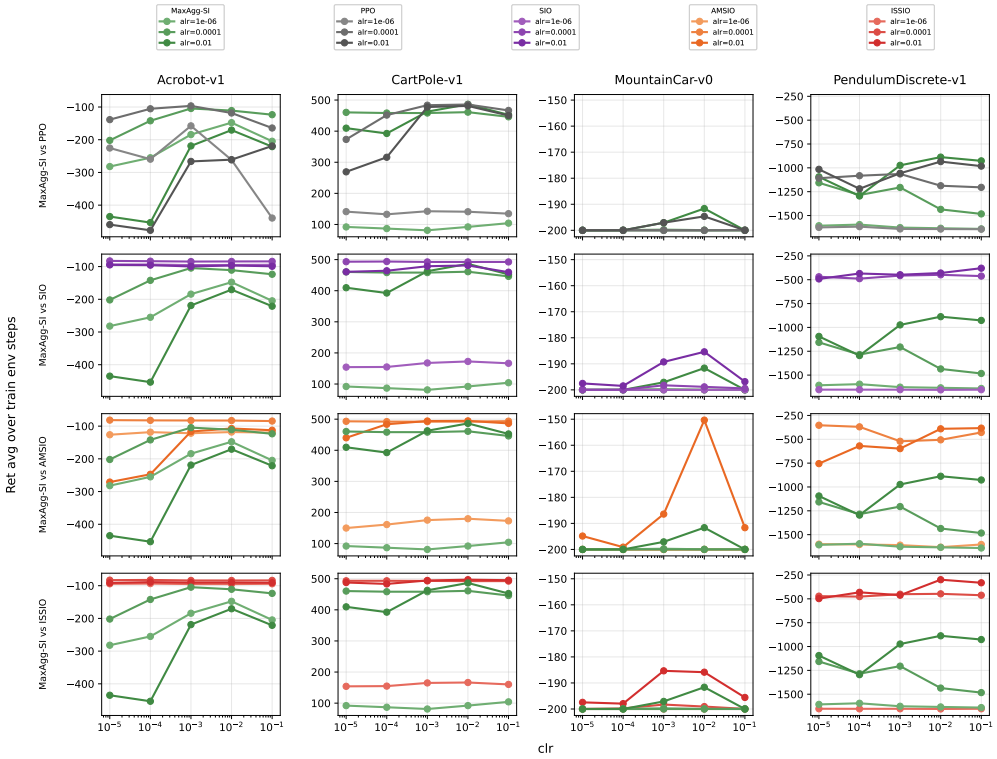

Figure 16: Mean evaluation returns over training for expert set `Comp1`, with MaxAgg-SI.

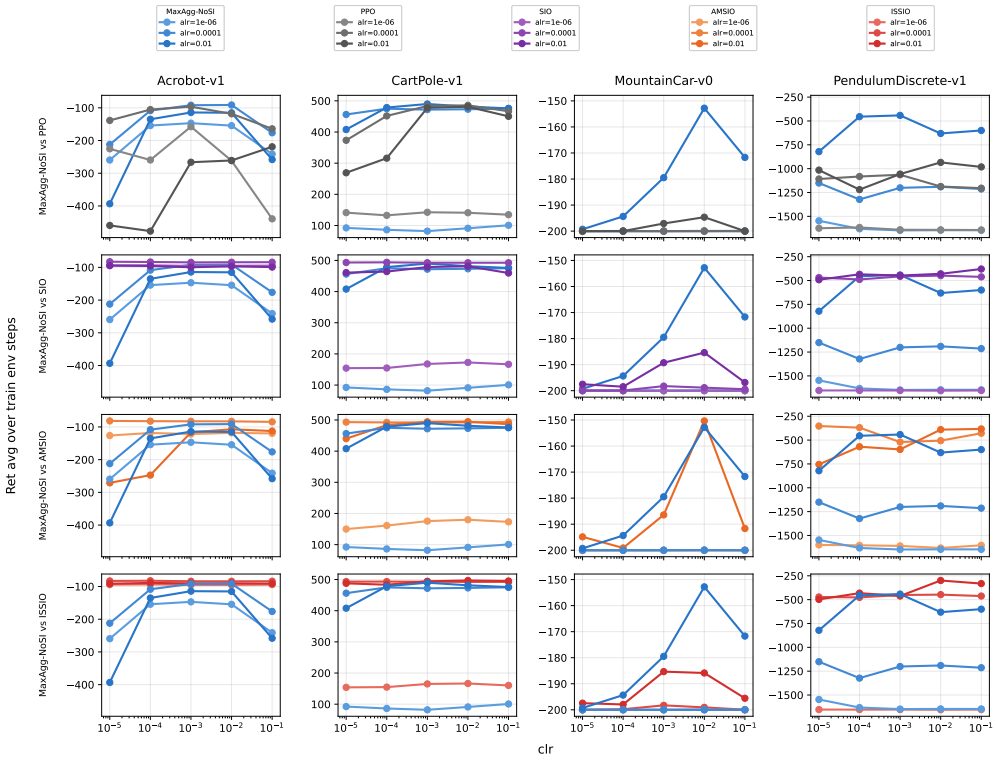

Figure 17: Mean evaluation returns over training for expert set `Comp1`, with MaxAgg-NoSI.

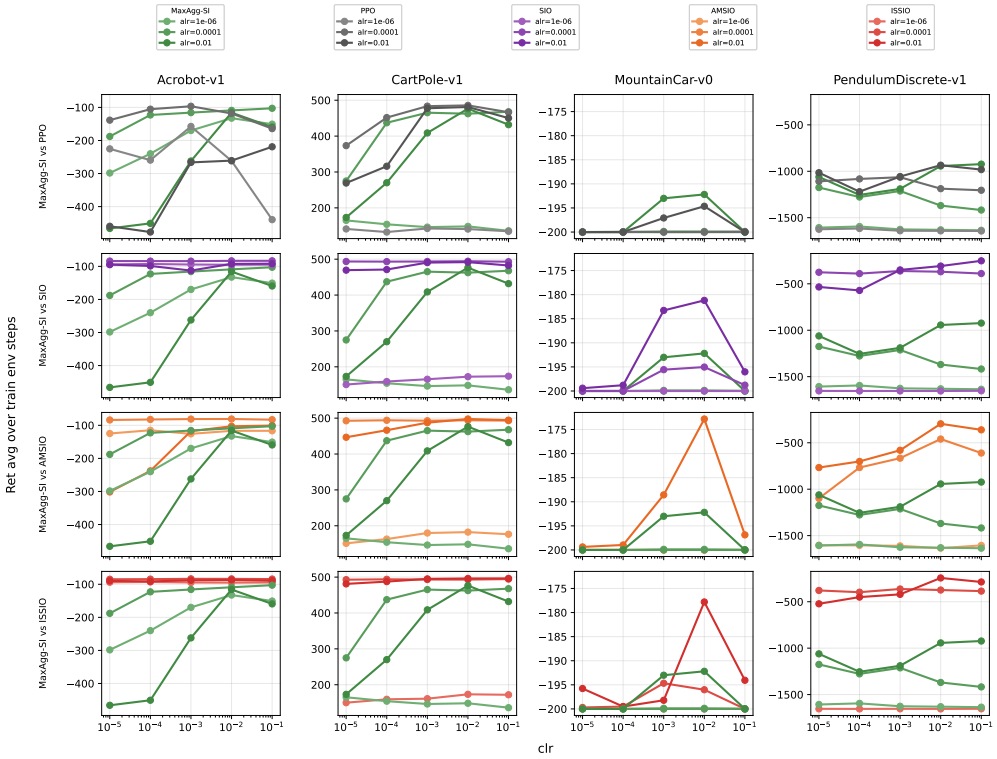

Figure 18: Mean evaluation returns over training for expert set `Comp2`, with MaxAgg-SI.

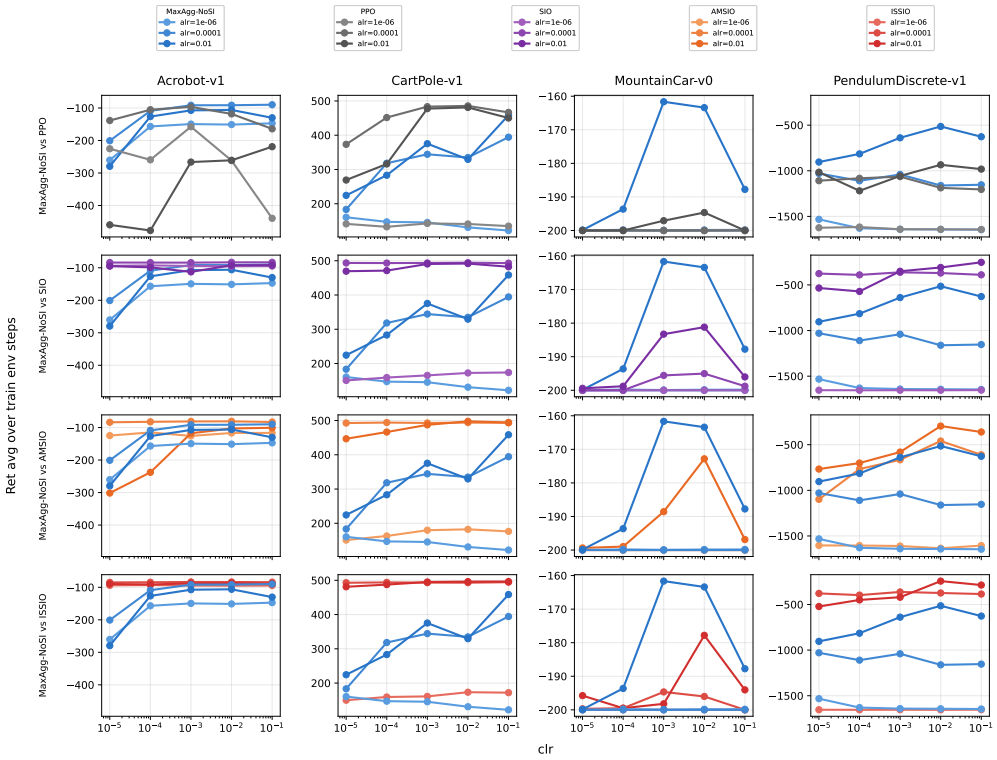

Figure 19: Mean evaluation returns over training for expert set `Comp2`, with MaxAgg-NoSI.

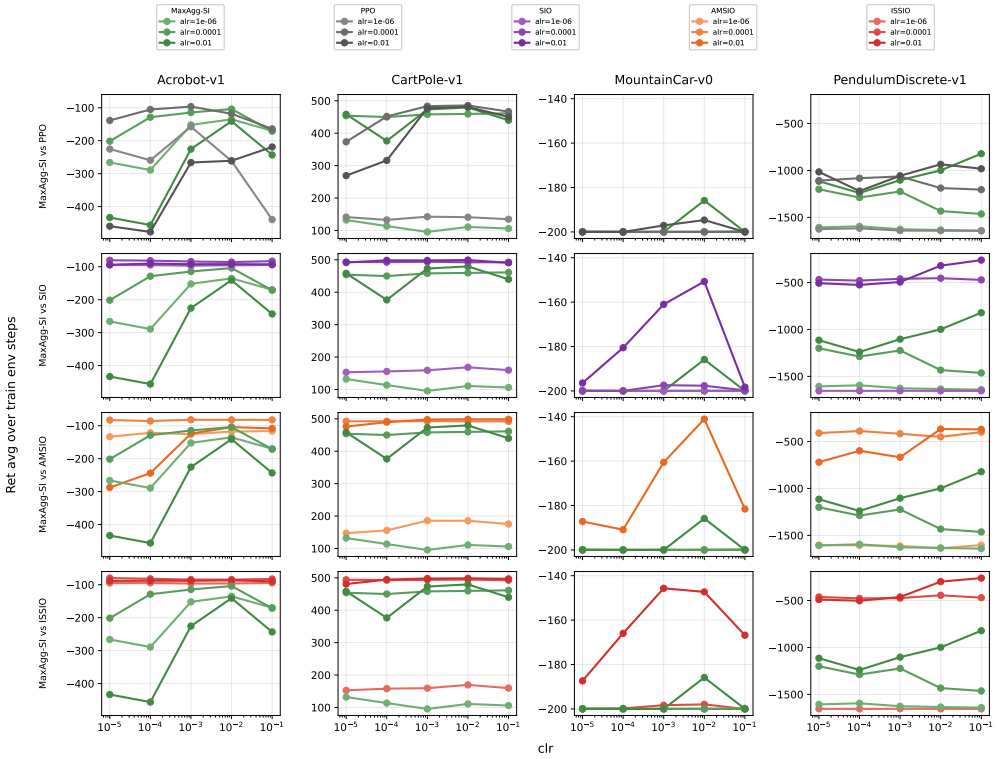

Figure 20: Mean evaluation returns over training for expert set `Comp3`, with MaxAgg-SI.

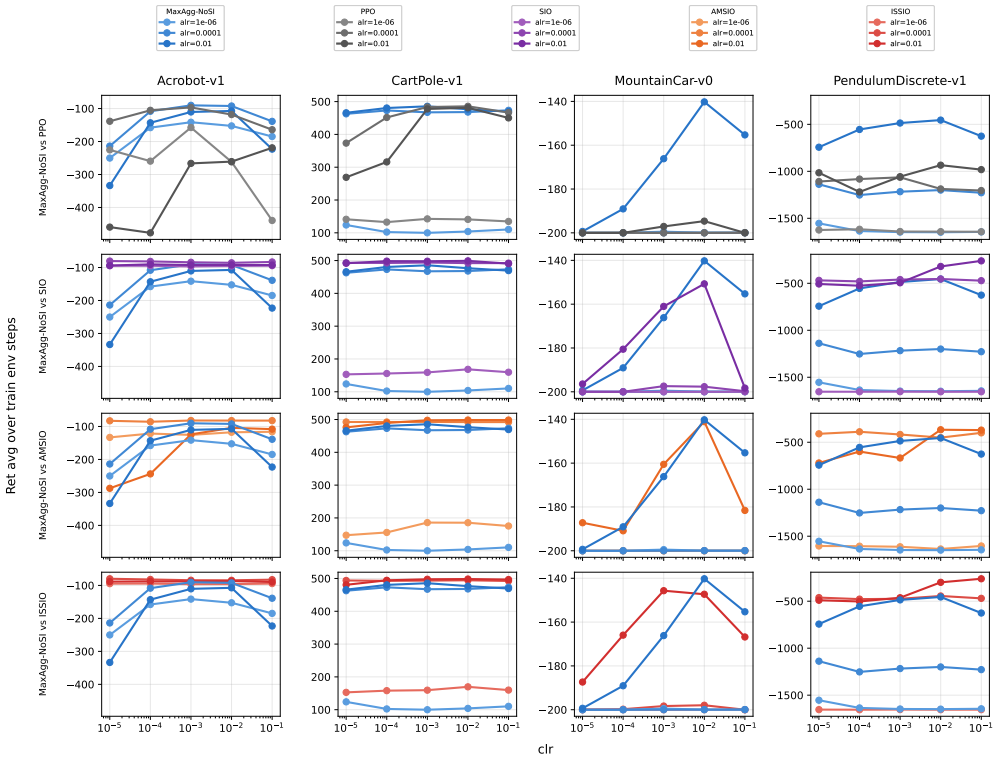

Figure 21: Mean evaluation returns over training for expert set `Comp3`, with MaxAgg-NoSI.



Figure 22: Mean evaluation returns over training for expert set `Comp4`, with MaxAgg-SI.



Figure 23: Mean evaluation returns over training for expert set `Comp4`, with MaxAgg-NoSI.

L DISCRETE-ACTION EXPERT-SELECTION PLOTS ($\xi = 0.5$)

Note that with $\xi = 0.5$ our parameterization causes an even larger initial student probability, e.g. with $K = 1$ our parameterization gives the student an initial 75% probability. Further note that the BC updates are somewhat incomparable to the RL updates in general (for example, the former could have far lower variance), which may cause the selector to choose BC less often than one might expect.

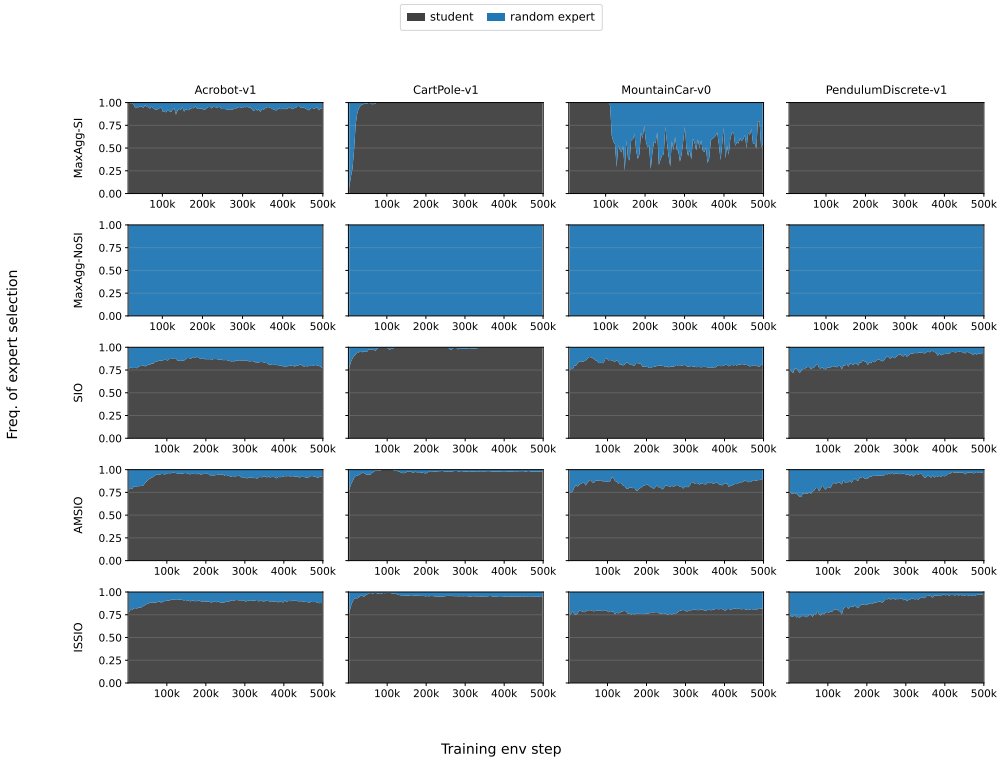

Figure 24: Expert-selection frequencies for expert set R. All LRs at 10^{-3}.

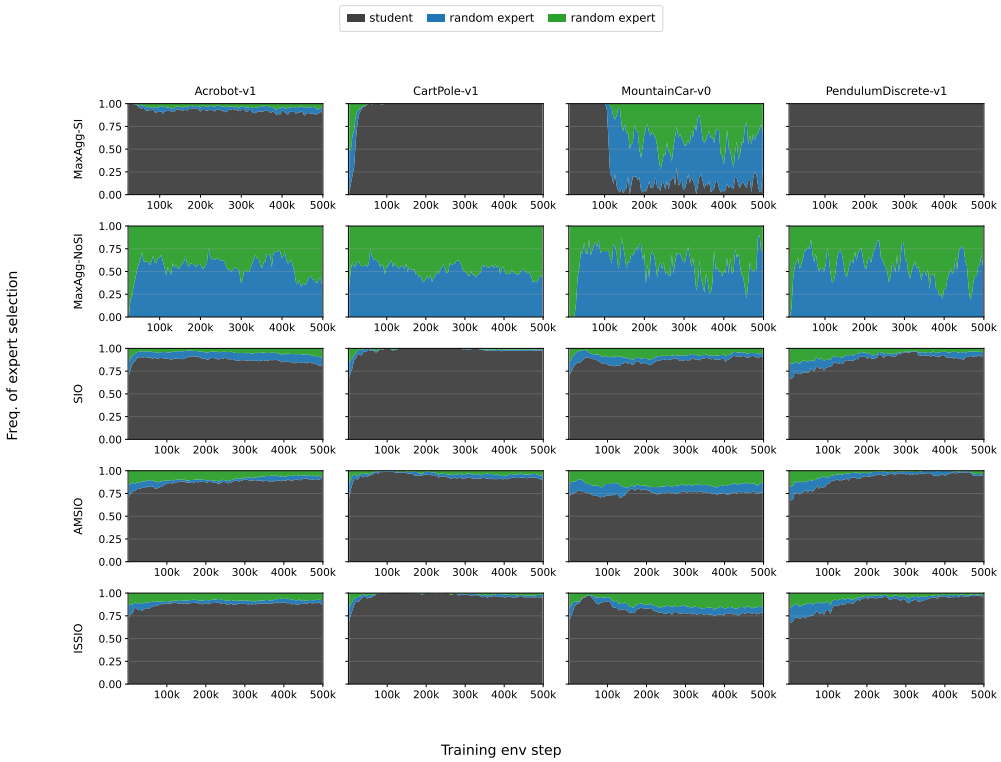

Figure 25: Expert-selection frequencies for expert set 2R. All LRs at 10^{-3}.

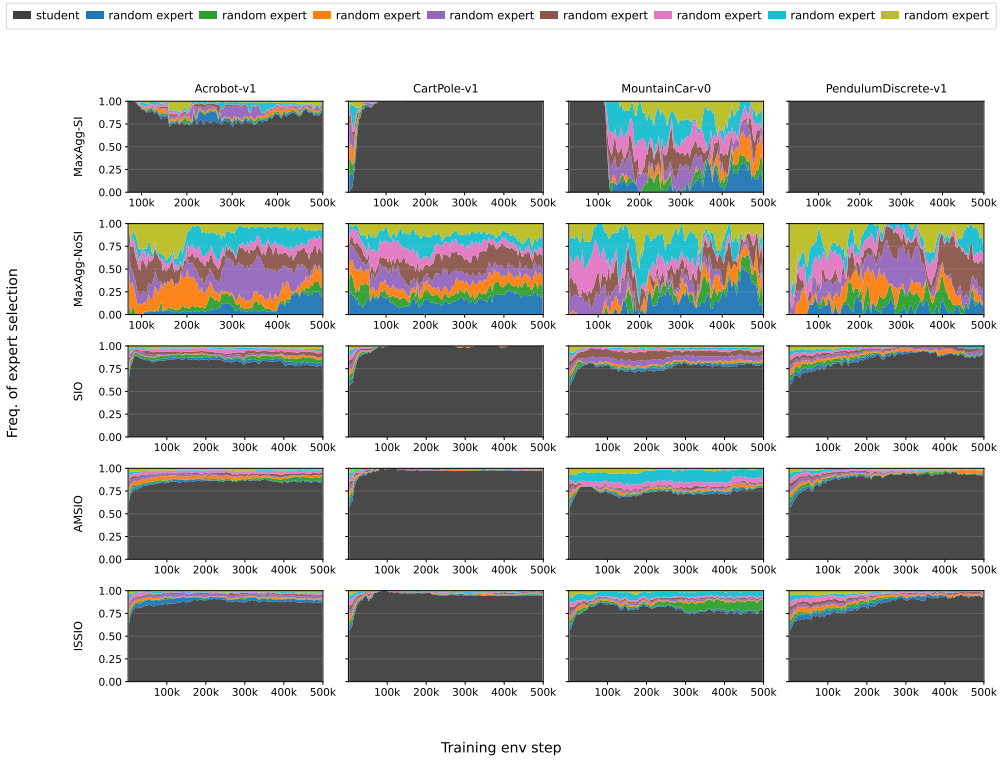

Figure 26: Expert-selection frequencies for expert set 8R. All LRs at 10^{-3}.

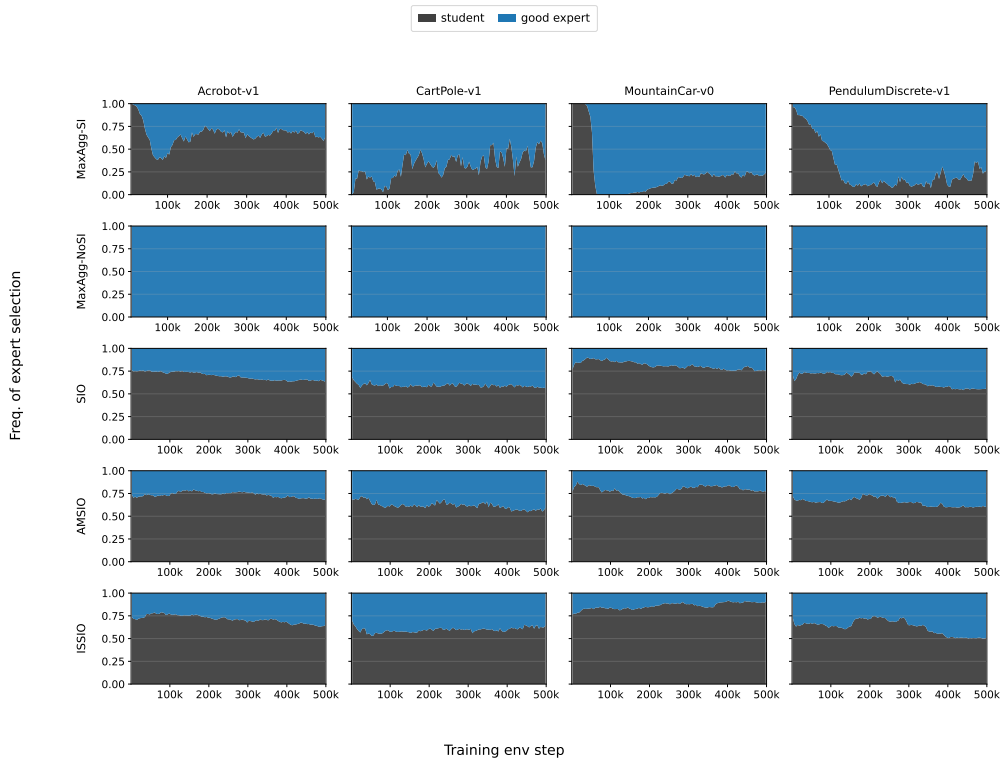

Figure 27: Expert-selection frequencies for expert set `G`. All LRs at 10^{-3}.

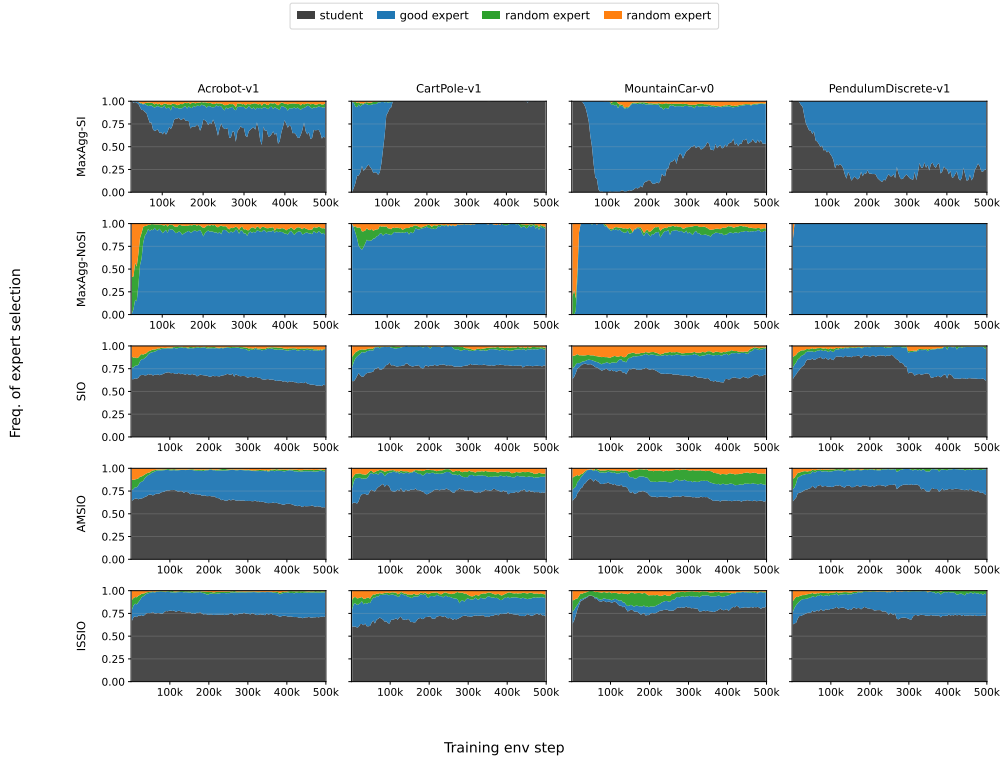

Figure 28: Expert-selection frequencies for expert set `G2R`. All LRs at 10^{-3}.

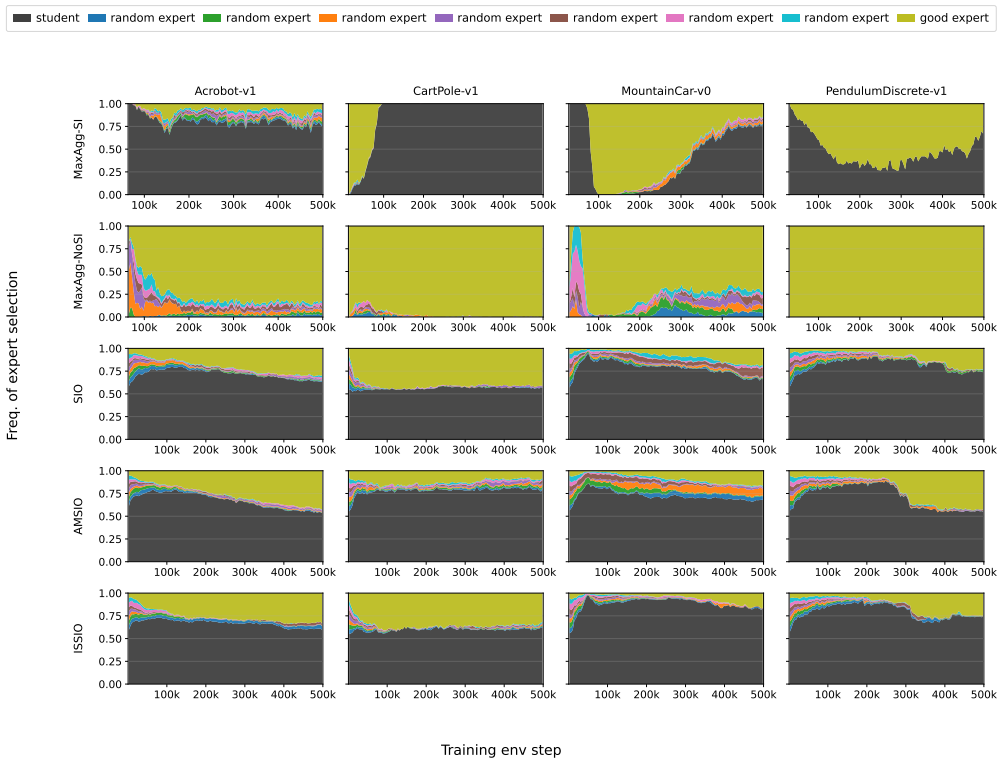

Figure 29: Expert-selection frequencies for expert set `G7R`. All LRs at 10^{-3}.

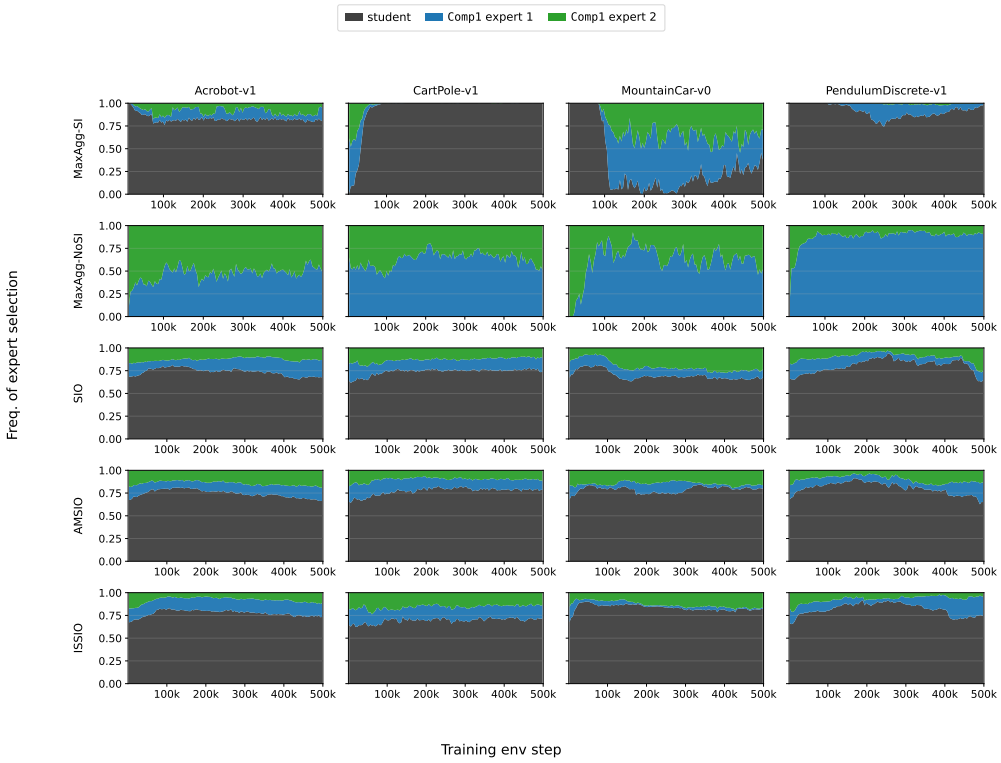

Figure 30: Expert-selection frequencies for expert set `Comp1`. All LRs at 10^{-3}.

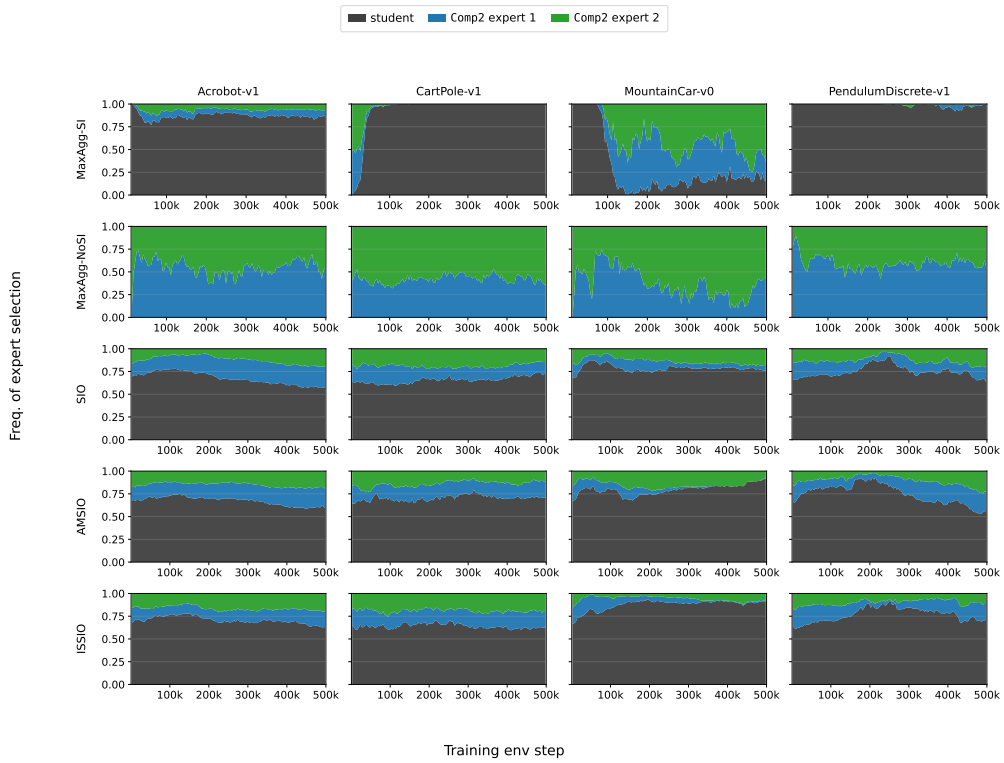

Figure 31: Expert-selection frequencies for expert set `Comp2`. All LRs at 10^{-3}.

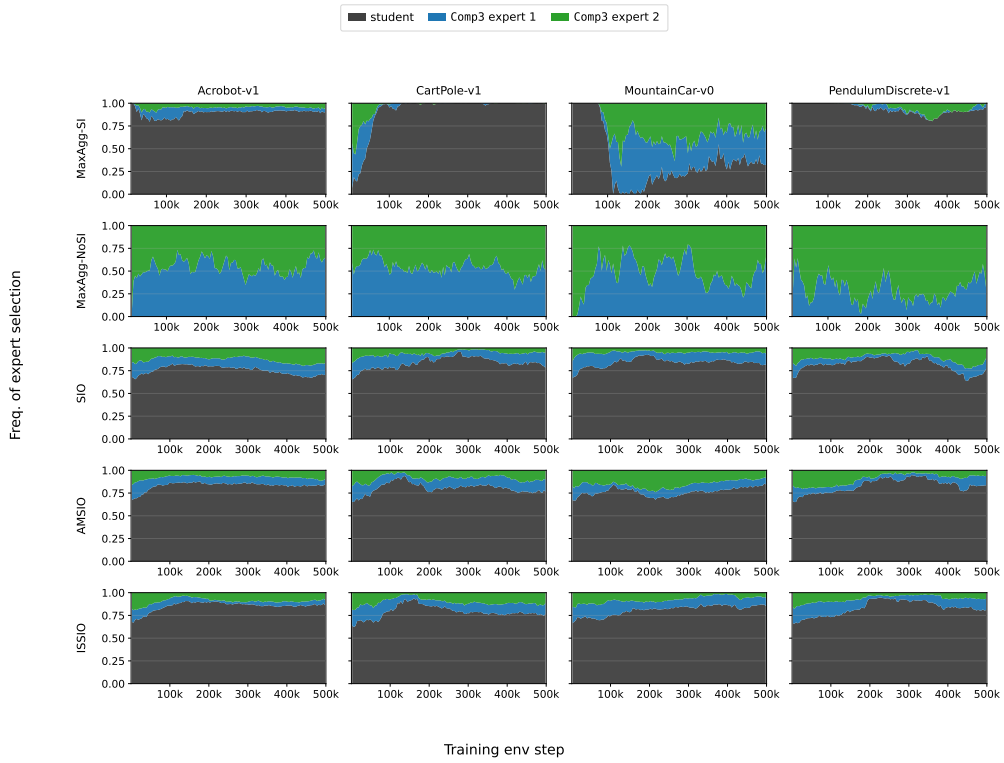

Figure 32: Expert-selection frequencies for expert set `Comp3`. All LRs at 10^{-3}.



Figure 33: Expert-selection frequencies for expert set `Comp4`. All LRs at 10^{-3}.