A Simple Baseline for Learning Approximate State Abstractions in Factored State Spaces

Anshuman Senapati, Josiah P. Hanna

Keywords: deep RL, state abstractions, inductive biases

Summary

The ability to identify and ignore task irrelevant environment variables is central to intelligent behavior. In reinforcement learning (RL), existing methods for learning such state abstractions typically rely on auxiliary objectives and such objectives tend to add significant complexity to the base RL algorithm. In this work, we take a step back and ask: can task-specific abstraction emerge from return optimization alone, without any additional objectives? We introduce a surprisingly simple neural network architecture change: a learnable, state-independent attention mask applied to the inputs of the policy and value networks and trained end-to-end using only the RL objective. Despite its simplicity, this architectural modification consistently improves sample efficiency and learns to mask out distracting input variables across 12 continuous control tasks. We analyze the dynamics of gradient descent using this method on a linear regression task and demonstrate suppression of distracting input features. Finally, we conduct experiments on toy MDPs and show that the attention mask increases the accuracy of action-values and induces a soft abstraction over a factored state space. Our findings challenge the need for complex auxiliary objectives to learn state abstractions in this setting and suggest a simple baseline for future research.

Contribution(s)

1. We introduce a state-independent attention mask for factored state spaces that is trained end-to-end using either an actor or critic loss function such that task-irrelevant state variables are suppressed relative to task-relevant variables.
   Context: Prior work has largely focused on using auxiliary objectives to encourage abstraction, either through similarity metrics (Zhang et al., 2020; Castro et al., 2021), causal modeling of the environment dynamics (Wang et al., 2024) or other losses that shape representations (Zhou et al., 2023; Wu et al., 2021; Liu et al., 2023). In contrast, we introduce a simple change to existing neural architectures and show that task-irrelevant variables are down-weighted relative to task-relevant ones (see Figure 3).
2. We show that the proposed architectural modification increases sample efficiency in the presence of task-irrelevant, distracting state variables.
   Context: We conduct experiments on 12 tasks from the DM Control Suite, augmented with distractor variables following prior work on learning in the presence of irrelevant or spurious features in deep RL (Wang et al., 2022b; 2024; Fu et al., 2021; Wang et al., 2022a; Zhang et al., 2020). We show that our architecture increases sample efficiency and sometimes obtains higher final performance than other baselines, and approaches the performance of an oracle that uses the ground-truth state-abstraction (see Figure 1).
3. We identify a setting where distracting variables introduce inherent optimization difficulties even in the infinite data regime, and that the proposed mechanism mitigates these.
   Context: We consider a linear regression task and show that distractors slow convergence even with expected gradient updates, thus showing that distracting inputs are an optimization challenge and not just a statistical one (Propositions 1 and 2). Our analysis also shows that the learnable mask we introduce tends to suppress distracting inputs (Proposition 3).

A Simple Baseline for Learning Approximate State Abstractions in Factored State Spaces

Anshuman Senapati[1], **Josiah P. Hanna**[2]
anshu001@bu.edu, jphanna@cs.wisc.edu

[1]**Boston University**
[2]**University of Wisconsin—Madison**

Abstract

The ability to identify and ignore task irrelevant environment variables is central to intelligent behavior. In reinforcement learning (RL), existing methods for learning such state abstractions typically rely on auxiliary objectives and such objectives tend to add significant complexity to the base RL algorithm. In this work, we take a step back and ask: can task-specific abstraction emerge from return optimization alone, without any additional objectives? We introduce a surprisingly simple neural network architecture change: a learnable, state-independent attention mask applied to the inputs of the policy and value networks and trained end-to-end using only the RL objective. Despite its simplicity, this architectural modification consistently improves sample efficiency and learns to mask out distracting input variables across 12 continuous control tasks. We analyze the dynamics of gradient descent using this method on a linear regression task and demonstrate suppression of distracting input features. Finally, we conduct experiments on toy MDPs and show that the attention mask increases the accuracy of action-values and induces a soft abstraction over a factored state space. Our findings challenge the need for complex auxiliary objectives to learn state abstractions in this setting and suggest a simple baseline for future research.

1 Introduction

In many practical RL applications, observations naturally take the form of collections of structured variables, such as sensor readings, system variables, and engineered features (Zhao et al., 2021; Evans et al., 2020; Dulac-Arnold et al., 2021). For example, a household robot may simultaneously perceive furniture, humans, ambient sounds, lighting, temperature, and floor texture, yet only certain features contribute meaningfully to its task. This abundance of candidate variables presents a fundamental challenge for RL algorithms lacking prior knowledge of which components are relevant. Without such knowledge, RL agents can learn policies and value functions that depend on spurious or weakly correlated variables, degrading sample efficiency and generalization (Wang et al., 2024).

To identify and discard task-irrelevant state variables, prior work has introduced auxiliary objectives that provide abstraction signals beyond reward alone. These include optimizing metrics that quantify state similarity based on reward and transition dynamics (Zhang et al., 2020; Castro et al., 2021) and modeling reward and transition dynamics (Wang et al., 2024; 2022b; Fu et al., 2021; Wang et al., 2022a). However, the role of architectural choices in enabling or shaping such abstraction has received comparatively little attention. Motivated by this gap, we aim to answer the following question:

Can architectural choices and the RL objective alone suffice to learn abstract state representations in factored state spaces?

In this work, we show that the answer can be yes, at least for the specific form of abstraction known as feature selection. In this setting, the agent is presented with a large number of variables—some relevant to optimal decision-making and some irrelevant—and the agent must select which ones can be ignored without loss of performance. Building on the framework of Wang et al. (2024), we investigate the capacity of neural networks to suppress such distracting input variables. We then introduce a learnable attention mask applied to the inputs of the neural network. Our approach draws inspiration from prior work on masking mechanisms, which have been integrated into various architectures and shaped using diverse optimization objectives (Wang et al., 2024; Wu et al., 2021; Grooten et al., 2023; Salter et al., 2021). Moreover, there has been limited empirical or theoretical inquiry into why distractors are so detrimental or how such seemingly minor architectural modifications can produce substantial gains in performance. We provide insight into this phenomenon from two complementary perspectives. First, we perform a gradient dynamics analysis of stochastic gradient descent updates in-expectation for linear models with and without attention-based masking. We show that the detrimental effects of distractors and the benefits of bounded masking extend beyond RL to general function approximation. Second, using a Deep Q-Network (DQN) (Mnih et al., 2015) trained on randomly generated toy MDPs adapted from Yang et al. (2022), we show that even in non-linear regimes, our method yields statistically significantly better estimates of the optimal Q-value function compared to vanilla MLPs.

Together, our findings show that an architectural inductive bias, when trained solely under the standard RL objective, is sufficient to give rise to an abstract state representation. These findings suggest an alternative to more complex methods relying on auxiliary losses to induce state abstraction.

2 Related Work

In this section, we review the significant literature on learning state abstractions, abstract state representations, and handling distracting inputs in deep RL.

2.1 Learning Approximate State Abstractions

Bisimulation (Givan et al., 2003) formalizes exact abstraction by grouping states with behaviorally indistinguishable dynamics, while MDP homomorphisms (Ravindran, 2004) define surjective mappings that preserve rewards and transitions in expectation, enabling approximate abstractions. Subsequent work relaxed exact equivalence to bounded or factored settings (Dean et al., 2013), related homomorphisms to lax bisimulation with approximation guarantees (Taylor et al., 2008), established polynomial error bounds for approximate abstractions (Abel et al., 2016), and unified abstraction notions while showing that approximate abstractions can yield near-optimal policies (Li et al., 2006). Building on this foundation, auxiliary losses based on bisimulation metrics (Ferns et al., 2004) have been used to shape representation spaces so that distances reflect behavioral similarity (Zhang et al., 2020; Castro et al., 2021). Related approaches such as DeepMDP (Gelada et al., 2019) learn latent MDP models by predicting rewards and transitions, ensuring dissimilar states are embedded distinctly. In contrast, we show that at least in settings with distracting, task-irrelevant variables, a simple architectural bias can induce task-aligned abstractions solely through reward-driven end-to-end learning, without auxiliary supervision.

2.2 Handling distractions and masking in Deep RL

A standard approach to studying distractors in deep RL appends irrelevant variables to observations and evaluates whether agents can ignore them. Within this framework, CBM (Wang et al., 2024) learn bisimulation-consistent abstractions by combining reward modeling, implicit dynamics estimation, and conditional mutual information tests to derive a binary mask of reward- and dynamics-relevant variables, though at the cost of training multiple models and maintaining a causal graph during learning. Other methods mitigate distractors through auxiliary representation-shaping objectives: sequential reward prediction (Zhou et al., 2023) and bisimulation-based prototype clustering

(Liu et al., 2023). Architectural attention mechanisms have also been explored: self-attention in policy and value networks (Bramlage & Cortese, 2022), recurrent key-query-value attention (Mott et al., 2019; Vaswani et al., 2017), and attention in noisy tasks (Salter et al., 2021). Some approaches decouple distractor suppression from policy learning, extracting invariant foreground features via keypoint detection (Wang et al., 2021) or learning input-dependent masks through reconstruction (Wu et al., 2021). Most recently, Grooten et al. (2023) train an observation-conditioned mask using critic loss alone. Many of these methods operate by masking out irrelevant pixels whose irrelevance can change from observation to observation, and the mask is often trained with auxiliary objectives. Our method, by contrast, applies to settings where irrelevance is conditioned on the task rather than on the observation: the same input dimensions are irrelevant at all timesteps. While learning an input mask is not itself a new idea, we show that in such settings, a single, state-independent mask, trained end-to-end using only the RL objective, successfully identifies the features that can be ignored.

3 Problem Setting: Factored MDPs with Task-Irrelevant State Variables

We consider the setting introduced by Wang et al. (2024), where observations are state-based and contain both task-relevant and task-irrelevant components. We model these environments as *factored MDPs*, defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathbf{s} \in \mathcal{S}$ denotes the state, $\mathbf{a} \in \mathcal{A}$ the action, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function with $P(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) = \mathbb{P}(\mathbf{s}_{t+1} = \mathbf{s}' \mid \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a})$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. A stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ defines a distribution over actions, such that $\pi(\mathbf{s}, \mathbf{a}) = \mathbb{P}(\mathbf{a}_t = \mathbf{a} \mid \mathbf{s}_t = \mathbf{s})$. The state space factorizes as $\mathcal{S} = \mathcal{X}_{\text{rel}} \times \mathcal{X}_{\text{irr}}$. A full state $\mathbf{s} \in \mathcal{S}$ is represented as $\mathbf{s} = (\mathbf{x}_{\text{rel}}, \mathbf{x}_{\text{irr}})$, where $\mathbf{x}_{\text{rel}} \in \mathcal{X}_{\text{rel}}$ comprises task-relevant variables that influence both the transition dynamics and the reward, and $\mathbf{x}_{\text{irr}} \in \mathcal{X}_{\text{irr}}$ comprises task-irrelevant variables that evolve independently and have no causal influence on either reward or dynamics. Specifically, the transition and reward functions factor as $P(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}) = P(\mathbf{x}'_{\text{rel}} \mid \mathbf{x}_{\text{rel}}, \mathbf{a}) \cdot P(\mathbf{x}'_{\text{irr}} \mid \mathbf{x}_{\text{irr}}, \mathbf{a})$ and $r(\mathbf{s}, \mathbf{a}) = r(\mathbf{x}_{\text{rel}}, \mathbf{a})$. In practice, the exact factorization may only hold approximately. In our experiments, we include a mixture of distractor types: some evolve independently of the agent's actions (e.g., following their own stochastic processes), while others evolve conditionally on the agent's actions via $P(\mathbf{x}'_{\text{irr}} \mid \mathbf{x}_{\text{irr}}, \mathbf{a})$. Crucially, these variables remain irrelevant to both the reward and the transitions of $\mathbf{x}_{\text{rel}}$, thereby ensuring they are unnecessary inputs for the optimal policy.

4 State Abstraction through State-Independent Input Masking

Given a standard model-free deep RL algorithm, we introduce a lightweight architectural modification that identifies and suppresses task-irrelevant variables, inducing a task-specific abstraction learned during end-to-end training. Let the state be factored as $\mathbf{s} = (x_1, \ldots, x_n) \in \mathbb{R}^n$, where each x_i is a state variable. We associate each x_i with a learnable parameter $\phi_i \in \mathbb{R}$ and define the masking vector $\phi = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^n$. These parameters are initialized to zero and shared across all function approximators (policy and value networks). At each training and deployment step we compute a gating vector $\alpha = \sigma(\phi) \in (0, 1)^n$, where $\sigma(\cdot)$ is the element-wise sigmoid, and apply it via a Hadamard product $\tilde{\mathbf{s}} = \alpha \odot \mathbf{s}$ to obtain the masked state used as input to both actor and critic.

For actor–critic methods, ϕ is updated using either the actor or critic loss—but not both. In our experiments we evaluate both choices and stop gradients through the unused loss. For example, when updating via the actor loss only, the critic receives $\tilde{\mathbf{s}}_{\text{critic}} = \texttt{detach}(\alpha) \odot \mathbf{s}$ while the actor receives $\tilde{\mathbf{s}}_{\text{actor}} = \alpha \odot \mathbf{s}$. The gating parameters are updated through backpropagation of the actor loss: $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}_{\text{actor}}(\theta, \phi)$, where $\mathcal{L}_{\text{actor}}(\theta, \phi)$ denotes the actor loss (e.g., from PPO (Schulman et al., 2017) or SAC (Haarnoja et al., 2018)), θ are the policy parameters, and η_ϕ is the learning rate for ϕ. While θ parameterizes the policy network, ϕ is a separate parameter vector whose gradients arise solely through its effect on the masked input $\tilde{\mathbf{s}}_{\text{actor}}$.

This design learns ϕ solely from the RL objective and produces a *soft* abstraction rather than a hard partition over the state space. Since $\alpha = \sigma(\phi) \in (0, 1)^n$, each variable can be partially sup-

pressed, yielding a graded notion of relevance compatible with gradient-based optimization. Ideally, task-irrelevant variables receive weights near zero and task-relevant variables near one. In practice, exact hard abstractions may not emerge, but the learned weights still suppress irrelevant variables sufficiently to improve sample efficiency in distractor-rich settings.

5 DM Control Suite Experiments

To evaluate the effectiveness of our proposed abstraction mechanism, we conduct extensive experiments across a range of continuous control benchmarks. We show that the presence of distractors in the state space leads to significant performance degradation of SAC, TD3, and PPO in standard MuJoCo tasks compared to learning without the distractors. We then show that the simple architecture modification we introduced in the previous section is sufficient to significantly decrease the gap between the methods.

5.1 Empirical Setup

Tasks. Our experiments span 12 continuous control tasks from the DeepMind Control Suite (MuJoCo): `walker-walk`, `walker-run`, `cheetah-run`, `hopper-hop`, `hopper-stand`, `finger-spin`, `finger-turn_easy`, `finger-turn_hard`, `fish-swim`, `fish-upright`, `reacher-hard`, and `swimmer-swimmer6`. These tasks cover a broad spectrum of locomotion and manipulation challenges, with varying levels of complexity in dynamics, control frequency, and reward structure. The diversity of tasks ensures that our findings are not tied to a narrow class of dynamics or reward functions. Each environment features continuous state and action spaces.

Distractor Augmentation. The distractor-augmented MDP includes two types of distractor variables: *uncontrollable* and *controllable* (Wang et al., 2024). Let $\mathbf{a}_t \in \mathbb{R}^d$ denote the action at time step t. Uncontrollable distractors are noise vectors $\mathbf{x}_t^{(\text{unc})} \in \mathbb{R}^{d_{\text{unc}}}$ sampled independently at each step: $\mathbf{x}_t^{(\text{unc})} \sim \mathcal{U}(\mu_{\text{unc}} - \delta, \mu_{\text{unc}} + \delta)$, where $\mu_{\text{unc}} \in \mathbb{R}^{d_{\text{unc}}}$ is a bias sampled once per experiment and $\delta \in \mathbb{R}_+^{d_{\text{unc}}}$ defines the variation range. These variables evolve independently of the agent's behavior and act as exogenous noise. Controllable distractors evolve deterministically from the agent's actions via $\mathbf{x}_t^{(\text{con})} = \mathbf{W}\mathbf{a}_t + \mathbf{b}$, where $\mathbf{W} \in \mathbb{R}^{d_{\text{con}} \times d}$ and $\mathbf{b} \in \mathbb{R}^{d_{\text{con}}}$ are sampled uniformly once per experiment and then fixed. This makes controllable distractors correlated with actions but irrelevant to task performance, as they do not influence rewards or the transition dynamics of $\mathbf{x}_{\text{rel}}$. To create high-dimensional observations, we augment the task-relevant state $\mathbf{x}_{\text{rel}}$ with 40 distractor variables, 20 controllable and 20 uncontrollable, yielding the factored observation $\mathbf{s} = (\mathbf{x}_{\text{rel}}, \mathbf{x}_{\text{con}}, \mathbf{x}_{\text{unc}})$. This setup tests whether agents can filter both deterministic and stochastic distractor sources across control tasks.

Implementation Details. We evaluate our learned masking mechanism across three deep reinforcement learning algorithms: Soft Actor-Critic (SAC) (Haarnoja et al., 2018), Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018), and Proximal Policy Optimization (PPO)(Schulman et al., 2017). In all cases, the actor and critic networks are implemented as multilayer perceptrons (MLPs), with ReLU activations used for SAC and TD3, and Tanh activations for PPO. The temperature of the sigmoid in the gating function is kept at 1 for these experiments. SAC and TD3 are trained off-policy using a replay buffer of size 1 million and run for 1 million environment steps per seed. For SAC, the entropy regularization coefficient is automatically tuned using a dedicated Adam optimizer. PPO, being an on-policy algorithm, is trained longer for 3 million steps. The full set of hyperparameters for each algorithm will be provided in the Appendix C. The results are compared against two baselines. First, *oracle* receives only task relevant variables as input (the ground truth abstraction) and thus serves as a strong upper bound on performance in that task and, second, *full* observes the full state augmented with distractors, which we show performs poorly compared to oracle performance.

All experiments were conducted on a high-throughput computing cluster. The compute pool consists of heterogeneous CPU-only worker nodes with Intel Xeon processors, ranging from 8 to 64 cores and 16–256 GB of RAM per node.

Evaluation Protocol. SAC and TD3 use a fixed protocol: after every training episode, the agent is evaluated on 50 episodic rollouts using a separate environment with independently sampled distractor biases and projection matrices. This ensures that the evaluation distractors are in-distribution but different from those seen during training. The average episodic return across the 50 test episodes is logged. For PPO, evaluation is conducted online by recording episodic returns directly during training rollouts.

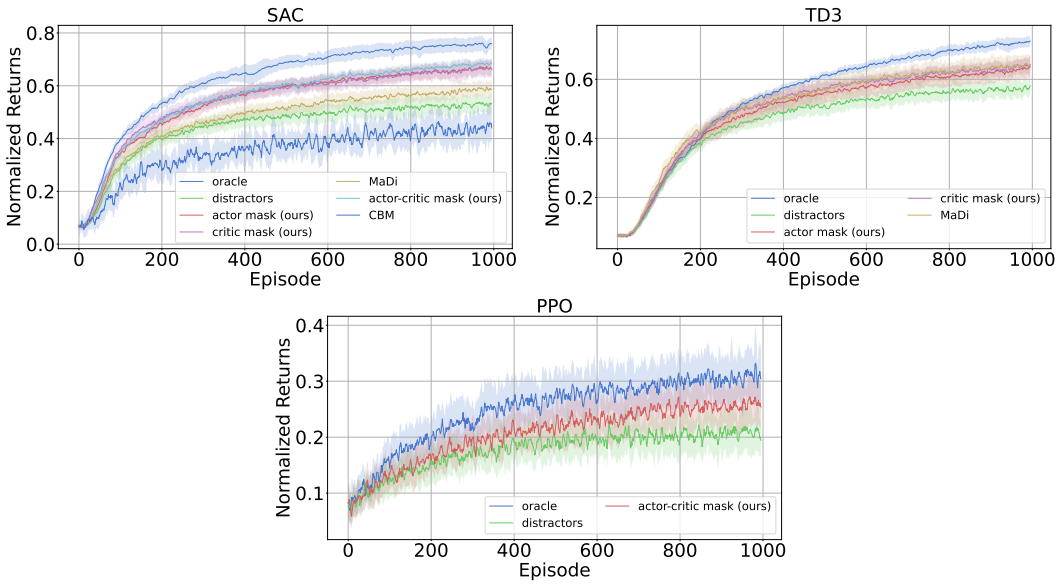


Figure 1: Performance of SAC, TD3, and PPO averaged across 12 DM Control tasks. For each task, returns are first normalized to $[0, 1]$ using the minimum and maximum reward observed across all methods and seeds for that task. For each method and seed, the normalized reward curves are then averaged across the 12 tasks, yielding one task-averaged curve per seed. The plotted curve is the mean of these 10 per-seed curves; shaded regions show 95% confidence intervals across these random seeds. Oracle, distractors (without attention), and our method are shown for each algorithm. MaDi is included for SAC and TD3, while CBM is included for SAC only.

5.2 Results

Figure 1 reports the average performance of each algorithm across all 12 tasks. In the presence of distractors, using vanilla MLP function approximators performs consistently worse than the oracle. In contrast, our method closes this gap, recovering performance towards the oracle despite having access only to reward-based supervision. Individual plots for each task are provided in the Appendix B.

Actor vs. Critic. When using the learned attention mask, using either the actor or critic signal in isolation yields stable training, suggesting that each loss alone is sufficient to drive task-aligned abstraction. For SAC, we also evaluate a variant where separate masks are trained for the actor and critic using their respective gradients. All these approaches achieve comparable performance. This similarity may stem from an implicit overlap in relevant features: in many con-

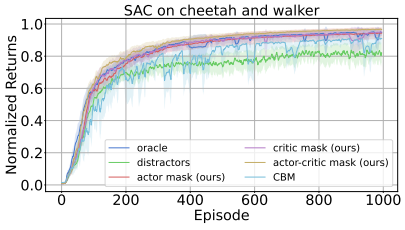


Figure 2: CBM on `cheetah` and `walker` environments.

tinuous control tasks, variables useful for value estimation also aid policy learning. In contrast, training a single shared mask using both losses simultaneously results in degenerate attention weights and near-zero returns. This indicates that the actor and critic provide conflicting gradient signals when applied jointly, consistent with findings from Garcin et al. (2025), which show that actor and critic networks tend to learn representations that are optimized for different purposes.

Comparison with CBM. On the environments considered in their original work, namely `cheetah` and `walker`, CBM (Wang et al., 2024) performs better than the distractor baseline and achieves performance close to our method as shown in Figure 2. However, when evaluated across the full set of tasks considered in this paper as shown in Figure 1, its performance degrades considerably and falls well below that of our approach on average.

Comparison with MaDi. Our state-independent masking performs comparably (for TD3) or better (for SAC) than the observation-conditioned mask proposed in Grooten et al. (2023). In contrast to our approach, such methods may overfit to idiosyncrasies in the input and learn contextual abstractions, rather than identifying a globally relevant set of features for RL.

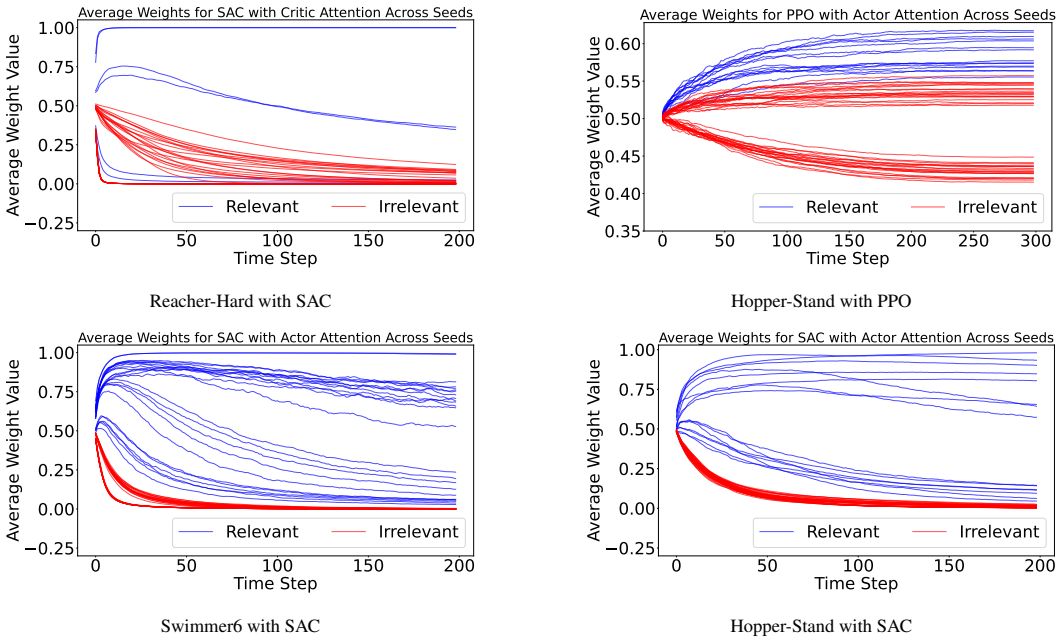


Figure 3: Trajectory of average attention weights across training. (a) SAC on `reacher-hard` and `swimmer-swimmer6`: confident masks emerge in easier tasks like `reacher-hard`, while `swimmer` shows more ambiguous gating. (b) PPO vs. SAC on `hopper-stand`: SAC shows sharper convergence; PPO remains diffuse. Task-relevant features are highlighted in blue, task-irrelevant in red. Each curve shows the attention weight for a single input dimension at each logging step, averaged across 10 seeds.

Correctness of Learned Abstraction. We track the evolution of attention weights during training and plot the weights for each observation variable for two distinct environments using the SAC agent in Figure 3a. While the attention weights do not strictly go to either 0 or 1 (producing a hard abstraction), they generally correspond to task-relevant variables receiving higher weights and task-irrelevant variables receiving lower weights. Notably, we observe that the sharpness of the learned gating values and the overall task performance are correlated. In environments where the agent performs well, such as `reacher-hard`, the attention mechanism tends to converge to a confident binary-like mask. As shown in Figure 3a, most attention weights (all but two) converge to either

0 or 1, indicating that the model has learned to clearly distinguish between relevant and irrelevant input variables. For more challenging tasks such as `swimmer-swimmer6`, where the agent's performance is relatively lower, the learned attention masks are less confident. The attention weights remain diffuse and fail to clearly separate important features from distractors. However, while the mask is less sharp, it still preserves the correct relative ranking of feature relevance. Similar plots are provided for all environments in Appendix B.

6 Controlled Analysis

In this section, we show mathematically and through controlled toy experiments that an input-independent attention mask can lead to meaningful abstractions that discards task-irrelevant variables. Of particular note, we show mathematically that the presence of distractors can slow learning even when using *expected updates* (i.e., using infinite data to compute each gradient step). We then show that the input-independent attention mask serves to suppress the effect of the distracting variable.

6.1 Gradient Dynamics in Linear Models

As critic training in RL amounts to repeatedly solving a regression task, in this section, we analyze the expected gradient updates for linear regression in the presence of distracting inputs. This setting is equivalent to learning a critic in a contextual bandit setting where value prediction reduces to supervised regression from state to reward. We analyze expected gradient descent updates for three settings: (i) in the absence of distractor variables (oracle case), (ii) in the presence of distractors without any form of masking or attention (full case), and (iii) in the presence of distractors with our proposed soft attention mechanism. We use expected updates (i.e., the infinite data regime) to focus on optimization rather than statistical challenges. All proofs are provided in Section A.

Consider a regression setting in which data is generated according to $Y = mX + c$ where $X \sim \mathcal{N}(0,1)$. There is also a distracting variable $D \sim \mathcal{U}(0,1)$. We consider training linear functions of the form $f_{\mathbf{w}}(x,d) = \mathbf{w}^\top [1, x, d]$ using gradient descent. The gradient descent update is

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta\, \mathbb{E}_{X\sim\mathcal{N}(0,1)} \mathbb{E}_{D\sim\mathcal{U}(0,1)} \left[\nabla_{\mathbf{w}^t} (f_{\mathbf{w}^t}(X,D) - Y)^2\right] \tag{1}$$

Proposition 1 (Gradient Updates with Known Distractors (Oracle))**.** *The linear model is given as:*

$$f_{\mathbf{w}}(x) = w_0 + w_1 x \tag{2}$$

and the expected update for $\mathbf{w}$ *is given by:*

$$\langle w_0^{t+1}, w_1^{t+1} \rangle \leftarrow \langle (1-\eta) w_0^t + \eta c,\ (1-\eta) w_1^t + \eta m \rangle \tag{3}$$

Proposition 2 (Gradient Updates with Unknown Distractors (Full))**.** *The linear model with distractors and no attention mechanism is given by:*

$$f_{\mathbf{w}}(x,d) = w_0 + w_1 x + w_2 d \tag{4}$$

and the expected updates for each component of $\mathbf{w}$ *are given by:*

$$\begin{aligned} w_0^{t+1} &\leftarrow (1-\eta) w_0^t - \tfrac{\eta}{2} w_2^t + \eta c \\ w_1^{t+1} &\leftarrow (1-\eta) w_1^t + \eta m \\ w_2^{t+1} &\leftarrow \left(1 - \tfrac{\eta}{3}\right) w_2^t - \tfrac{\eta}{2} w_0^t + \tfrac{\eta}{2} c \end{aligned} \tag{5}$$

Observation 1 (Distracting inputs mislead bias learning in the full model)**.** *With oracle knowledge of the distractor variable, the expected update results in $w_0 \to c$ and $w_1 \to m$ and the true data generating function is recovered. Without this oracle knowledge, the update for w_1 is unchanged but w_0 and w_2 now depend upon one another and change to try and explain the bias term, c, in the data generation function. Consequently, for any non-zero values of w_0 and w_2, w_0 moves toward $c - \frac{w_2}{2}$ and w_2 moves toward $\frac{3c}{2} - \frac{3w_0}{2}$, albeit the latter moves at the slower rate of $\frac{\eta}{3}$. While w_2 will eventually converge to zero and w_0 to c, the movement is less direct than the update in Proposition 1.*

Proposition 3 (Gradient Update: Attention-Based Model)**.** *The linear model with input-independent attention mask is given as:*

$$f_{\mathbf{w}}(x, d) = w_0 + w_1(x\,\sigma(\phi_1)) + w_2(d\,\sigma(\phi_2)), \tag{6}$$

where ϕ_1 and ϕ_2 are learnable parameters and σ is the sigmoid function. The expected update equations for $\mathbf{w}$, ϕ_1, and ϕ_2 are given as:

$$\begin{aligned}
w_0^{t+1} &\leftarrow w_0^t - \eta \left(w_0 + \tfrac{w_2\,\sigma(\phi_2)}{2} - c\right) \\
w_1^{t+1} &\leftarrow w_1^t - \eta \left(w_1\,\sigma(\phi_1)^2 - m\,\sigma(\phi_1)\right) \\
w_2^{t+1} &\leftarrow w_2^t - \eta\,\sigma(\phi_2) \left(\tfrac{w_0}{2} + \tfrac{w_2\,\sigma(\phi_2)}{3} - \tfrac{c}{2}\right) \\
\phi_1^{t+1} &\leftarrow \phi_1^t - \eta\,w_1\,\sigma(\phi_1)(1 - \sigma(\phi_1))(w_1\,\sigma(\phi_1) - m) \\
\phi_2^{t+1} &\leftarrow \phi_2^t - \eta\,w_2\,\sigma(\phi_2)(1 - \sigma(\phi_2)) \left(\tfrac{w_0}{2} + \tfrac{w_2\,\sigma(\phi_2)}{3} - \tfrac{c}{2}\right)
\end{aligned} \tag{7}$$

Observation 2 (Attention mask updates suppress distractors.)**.** *The direction of the update for ϕ_2 is determined by the expression:*

$$w_2 \left(\frac{w_0}{2} + \frac{w_2\,\sigma(\phi_2)}{3} - \frac{c}{2}\right)$$

as the factor $\sigma(\phi_2)(1 - \sigma(\phi_2))$ in Equation 7 is always non-negative and thus only serves to scale the update. When the distractor's contribution is large, i.e., when $|w_2|$ is large or $\sigma(\phi_2)$ closer to one, this term is typically positive, leading to a downward update that drives $\sigma(\phi_2) \to 0$ and effectively suppressing the distractor. Notably, suppression emerges without explicit knowledge of distracting variables; the model learns to attenuate irrelevant inputs purely from the error signal.

Empirical Validation We also validate these conclusions empirically in a synthetic regression setting using the derived gradient update equations for both the full and attention-based model. All weights are initialized using a uniform distribution $\mathcal{U}(-1/\sqrt{2}, 1/\sqrt{2})$, and updates are performed using fixed-step gradient descent with a step size of 0.01. For the attention-based model, the attention weights are initialized to zero and updated jointly with the main weights via backpropagation through the sigmoid gating functions. At each training step, we compute the loss over a sampled batch of 5,000 data points drawn from the true data-generating process. This process is repeated across 50 random seeds to account for variance due to initialization. As shown in Figure 4, even with large sample sizes, the attention-based updates consistently converge faster and more stably to the optimal solution compared to updates without the attention interactions.

6.2 Policy Evaluation

Finally, we run a policy evaluation experiment in a controlled, low-dimensional MDP to understand if the attention mask is improving the accuracy of policy evaluation. The base environment is a custom MDP with continuous one-dimensional state space and two discrete actions adapted from Yang et al. (2022). The transition dynamics for each action are defined via randomly sampled piecewise linear functions, making the environment deterministic but non-trivial. We discretize the

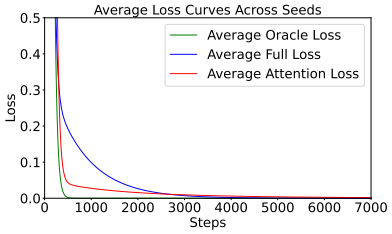


Figure 4: Loss with respect to the true data generating function. Results aggregated over 50 seeds.

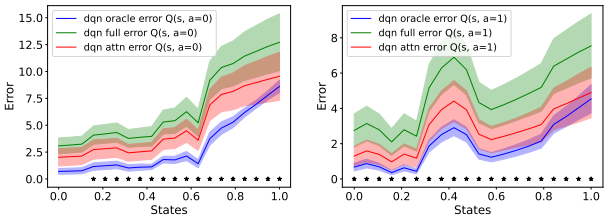


Figure 5: Mean squared error of Q-value estimation for action 0 and 1 across oracle, full, and attention-based agents.

continuous state space into 20 evenly spaced states and compute the ground-truth Q-values via value iteration. For the full setting, the agent is trained with 20 additional task-irrelevant dimensions—10 controllable and 10 uncontrollable—appended to the 1D input. Each DQN variant is trained for 2,000 episodes using a small MLP. The distractors are generated similar to the main experiments. To evaluate learned Q-values, we run all trained models on a fixed evaluation set of states with the same distractor statistics and compare the predicted Q-values against the ground-truth values for both actions using per-state mean squared error.

We repeat all experiments over 20 random seeds and report the mean Q-value estimation errors along with 95% confidence intervals. Additionally, we perform paired t-tests across seeds to assess the statistical significance of the differences in estimation error between the attention-based and vanilla models. Each black star denotes a statistically significant difference in Q-value estimation error for that state, as determined by a paired t-test. The results are presented in Figure 5 for 20 discretized states in a randomly generated MDP. The sigmoid temperature used for the attention masks in this setting is the default value of 1 as used in the main experiments. Plots for other temperature settings are provided in Appendix B. Temperature is a sensitive hyperparameter: higher values yield sharper, more confident masks, but excessively high temperatures can lead to vanishing gradients and unstable training, resulting in overly confident but incorrect masking. As shown, the attention-based model yields significantly lower errors than the baseline in most of the 20 discretized states, indicating more accurate approximation of the true Q-values.

7 Conclusion

In this paper, we studied whether state abstractions in RL can emerge without auxiliary objectives. We introduced a simple architectural modification, a state-independent attention mask applied to actor and critic inputs, trained end-to-end using only the RL objective. Through empirical evaluation and theoretical analysis in linear settings, we showed that this enables the suppression of task-irrelevant inputs and closes the gap to agents with access to the ideal state abstraction. These findings (1) establish this simple architecture as a baseline for future work in abstraction learning and (2) motivate further exploration of architecture designs that promote abstraction solely through RL objectives.

A Gradient Updates Derivation

Let the true data generating function be given by $y(X) = mX + c$,
and the oracle function with learnable weights w_0 and w_1 be $f(x) = w_0 + w_1 x$
The gradient descent update rule for the weight vector $\mathbf{w} = \langle w_0, w_1 \rangle$ is,
$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \mathbb{E}_{X \sim P(X)} \left[\nabla_{\mathbf{w}^t} (f(X) - Y)^2 \right] = \mathbf{w}^t - 2\eta \mathbb{E}_X \left[(f_{\mathbf{w}}(X) - Y) \nabla_{\mathbf{w}} f(X) \right]$

A.1 Oracle Updates

Assuming the input X follows a standard normal distribution i.e., $X \sim \mathcal{N}(0, 1)$, we have $\mathbb{E}[X] = 0$ and $\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \implies \mathbb{E}[X^2] = 1$. We denote the expectation over $X \sim \mathcal{N}(0, 1)$ as $\mathbb{E}[(\cdot)]$. The gradient descent update for the weights $\langle w_0, w_1 \rangle$ is derived as follows:

$$
\begin{aligned}
&\langle w_0^{t+1}, w_1^{t+1} \rangle \\
&= \langle w_0^t, w_1^t \rangle - \eta \mathbb{E} \left[\nabla_{\mathbf{w}} \left((w_0^t + w_1^t X) - (mX + c) \right)^2 \right] \\
&= \langle w_0^t, w_1^t \rangle - \eta \mathbb{E} \left[\left((w_0^t + w_1^t X) - (mX + c) \right) \nabla_{\mathbf{w}} (w_0^t + w_1^t X) \right] \\
&= \langle w_0^t, w_1^t \rangle - \eta \mathbb{E} \left[\left((w_0^t + w_1^t X) - (mX + c) \right) \left\langle \tfrac{\partial (w_0^t + w_1^t X)}{\partial w_0^t}, \tfrac{\partial (w_0^t + w_1^t X)}{\partial w_1^t} \right\rangle \right] \\
&= \langle w_0^t, w_1^t \rangle - \eta \mathbb{E} \left[\left((w_0^t + w_1^t X) - (mX + c) \right) \langle 1, X \rangle \right] \\
&= \langle w_0^t, w_1^t \rangle - \eta \mathbb{E} \left[\langle w_0^t + w_1^t X - mX - c, w_0^t X + w_1^t X^2 - mX^2 - cX \rangle \right] \\
&= \langle w_0^t, w_1^t \rangle - \eta \left(\langle \mathbb{E}[w_0^t + w_1^t X - mX - c], \mathbb{E}[w_0^t X + w_1^t X^2 - mX^2 - cX] \rangle \right) \\
&= \langle w_0^t, w_1^t \rangle - \eta \left(\langle w_0^t - c, w_1^t - m \rangle \right) \\
&= \langle (1 - \eta) w_0^t + \eta c, (1 - \eta) w_1^t + \eta m \rangle.
\end{aligned}
$$

A.2 Full Updates

The model is given by $f(x, d) = w_0 + w_1 x + w_2 d$. Let the noise variable D be uniformly distributed between 0 and 1 i.e., $D \sim \mathcal{U}(0, 1)$. Then, its expected value and variance are $\mathbb{E}[D] = \frac{1}{2}$ and $\mathrm{Var}(D) = \mathbb{E}[D^2] - (\mathbb{E}[D])^2 = \frac{1}{12} \implies \mathbb{E}[D^2] = \frac{1}{3}$. Since $X \sim \mathcal{N}(0, 1)$ and $D \sim \mathcal{U}(0, 1)$ are sampled independently, their covariance is zero, and thus $\mathbb{E}[XD] = \mathbb{E}[X]\mathbb{E}[D] = 0 \cdot \frac{1}{2} = 0$. We denote the expectation over $X \sim \mathcal{N}(0, 1)$ and $D \sim \mathcal{U}(0, 1)$ as $\mathbb{E}[(\cdot)]$. The gradient descent update for the weight vector $\mathbf{w} = \langle w_0, w_1, w_2 \rangle$ is

$$
\begin{aligned}
\langle w_0^{t+1}, w_1^{t+1}, w_2^{t+1} \rangle &= \langle w_0^t, w_1^t, w_2^t \rangle - \eta \mathbb{E} \left[\nabla_{\mathbf{w}^t} \left\{ (w_0^t + w_1^t X + w_2^t D) - (mX + c) \right\}^2 \right] \\
&= \langle w_0^t, w_1^t, w_2^t \rangle - \eta \mathbb{E} \left[\left\{ (w_0^t + w_1^t X + w_2^t D) - (mX + c) \right\} \nabla_{\mathbf{w}^t} (w_0^t + w_1^t X + w_2^t D) \right] \\
&= \langle w_0^t, w_1^t, w_2^t \rangle - \eta \mathbb{E} \left[\left\{ (w_0^t + w_1^t X + w_2^t D) - (mX + c) \right\} \left\langle \frac{\partial f}{\partial w_0^t}, \frac{\partial f}{\partial w_1^t}, \frac{\partial f}{\partial w_2^t} \right\rangle \right] \\
&= \langle w_0^t, w_1^t, w_2^t \rangle - \eta \mathbb{E} \left[\left\{ (w_0^t + w_1^t X + w_2^t D) - (mX + c) \right\} \langle 1, X, D \rangle \right] \\
&= \langle w_0^t, w_1^t, w_2^t \rangle - \eta \mathbb{E} \left[\langle w_0^t + w_1^t X + w_2^t D - mX - c, \right. \\
&\qquad \left. w_0^t X + w_1^t X^2 + w_2^t XD - mX^2 - cX, w_0^t D + w_1^t XD + w_2^t D^2 - mXD - cD \rangle \right] \\
&= \langle w_0^t, w_1^t, w_2^t \rangle - \eta \left\{ \langle \mathbb{E}[w_0^t + w_1^t X + w_2^t D - mX - c], \right. \\
&\qquad \left. \mathbb{E}[w_0^t X + w_1^t X^2 + w_2^t XD - mX^2 - cX], \mathbb{E}[w_0^t D + w_1^t XD + w_2^t D^2 - mXD - cD] \rangle \right\} \\
&= \langle w_0^t, w_1^t, w_2^t \rangle - \eta \left\{ \left\langle w_0^t + \frac{w_2^t}{2} - c, w_1^t - m, \frac{w_0^t}{2} + \frac{w_2^t}{3} - \frac{c}{2} \right\rangle \right\} \\
&= \left\langle (1 - \eta) w_0^t - \frac{\eta}{2} w_2^t + \eta c, (1 - \eta) w_1^t + \eta m, \left(1 - \frac{\eta}{3}\right) w_2^t - \frac{\eta}{2} w_0^t + \frac{\eta c}{2} \right\rangle.
\end{aligned}
$$

A.3 Attention Updates

The model with attention mechanisms is $f(x, d) = w_0 + w_1 (x\, \sigma(\phi_1)) + w_2 (d\, \sigma(\phi_2))$, where $\sigma(\phi) = \frac{1}{1 + e^{-\phi}}$ is the sigmoid function, and its derivative is $\frac{d}{d\phi} \sigma(\phi) = \sigma(\phi)(1 - \sigma(\phi))$. The gradient descent update for the parameter vector $\langle w_0, w_1, w_2, \phi_1, \phi_2 \rangle$ is

$\langle w_0^{t+1}, w_1^{t+1}, w_2^{t+1}, \phi_1^{t+1}, \phi_2^{t+1} \rangle$
$= \langle w_0^t, w_1^t, w_2^t, \phi_1^t, \phi_2^t \rangle - \eta \mathbb{E}\left[\nabla_{\mathbf{w},\phi} \left\{(w_0 + w_1(X\,\sigma(\phi_1)) + w_2(D\,\sigma(\phi_2))) - (mX + c)\right\}^2\right]$
$= \langle w_0^t, w_1^t, w_2^t, \phi_1^t, \phi_2^t \rangle - \eta \mathbb{E}\left[\{f(X, D) - (mX + c)\} \nabla_{\mathbf{w},\phi} f(X, D)\right],$
where the gradient of $f(X, D)$ with respect to the parameters is $\nabla_{\mathbf{w},\phi} f(X, D) = \langle 1,\ X\,\sigma(\phi_1),\ D\,\sigma(\phi_2),\ w_1\,X\,\sigma(\phi_1)(1 - \sigma(\phi_1)),\ w_2\,D\,\sigma(\phi_2)(1 - \sigma(\phi_2)) \rangle$. Let the error be denoted as error $= f(X, D) - (mX + c)$. Taking expectations, we derive the update rules for each parameter.

For the w_0 component: $\mathbb{E}[\text{error} \cdot (1)] = w_0 + \frac{w_2\,\sigma(\phi_2)}{2} - c$, leading to the update $w_0^{t+1} = w_0 - \eta \left(w_0 + \frac{w_2\,\sigma(\phi_2)}{2} - c\right)$.

For the w_1 component: $\mathbb{E}[\text{error} \cdot (X\,\sigma(\phi_1))] = \sigma(\phi_1)(w_1\,\sigma(\phi_1) - m)$, leading to the update $w_1^{t+1} = w_1 - \eta \left(w_1\,\sigma(\phi_1)^2 - m\,\sigma(\phi_1)\right)$.

For the w_2 component: $\mathbb{E}[\text{error} \cdot (D\,\sigma(\phi_2))] = \sigma(\phi_2) \left(\frac{w_0}{2} + \frac{w_2\,\sigma(\phi_2)}{3} - \frac{c}{2}\right)$, leading to the update $w_2^{t+1} = w_2 - \eta\,\sigma(\phi_2) \left(\frac{w_0}{2} + \frac{w_2\,\sigma(\phi_2)}{3} - \frac{c}{2}\right)$.

For the ϕ_1 component:
$\mathbb{E}[\text{error} \cdot (w_1\,X\,\sigma(\phi_1)(1 - \sigma(\phi_1)))] = w_1\,\sigma(\phi_1)(1 - \sigma(\phi_1))(w_1\,\sigma(\phi_1) - m)$, leading to the update $\phi_1^{t+1} = \phi_1 - \eta\,w_1\,\sigma(\phi_1)(1 - \sigma(\phi_1))(w_1\,\sigma(\phi_1) - m)$.

For the ϕ_2 component:
$\mathbb{E}[\text{error} \cdot (w_2\,D\,\sigma(\phi_2)(1 - \sigma(\phi_2)))] = w_2\,\sigma(\phi_2)(1 - \sigma(\phi_2)) \left(\frac{w_0}{2} + \frac{w_2\,\sigma(\phi_2)}{3} - \frac{c}{2}\right)$, leading to the update $\phi_2^{t+1} = \phi_2 - \eta\,w_2\,\sigma(\phi_2)(1 - \sigma(\phi_2)) \left(\frac{w_0}{2} + \frac{w_2\,\sigma(\phi_2)}{3} - \frac{c}{2}\right)$.

This completes the proof.

Acknowledgments

A portion of this work took place in the Prediction and Action Lab (PAL) at the University of Wisconsin–Madison. PAL research is supported by NSF (IIS-2410981) and the Wisconsin Alumni Research Foundation.

References

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pp. 2915–2923. PMLR, 2016.

Lennart Bramlage and Aurelio Cortese. Generalized attention-weighted reinforcement learning. *Neural Networks*, 145:10–21, 2022.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34:30113–30126, 2021.

Thomas L Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for markov decision processes. *arXiv preprint arXiv:1302.1533*, 2013.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Richard Andrew Evans, Jim Gao, Michael C Ryan, Gabriel Dulac-Arnold, Jonathan Karl Scholz, and Todd Andrew Hester. Optimizing data center controls using neural networks, May 5 2020. US Patent 10,643,121.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pp. 162–169, 2004.

Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning task informed abstractions. In *International Conference on Machine Learning*, pp. 3480–3491. PMLR, 2021.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Samuel Garcin, Trevor McInroe, Pablo Samuel Castro, Prakash Panangaden, Christopher G Lucas, David Abel, and Stefano V Albrecht. Studying the interplay between the actor and critic representations in reinforcement learning. *arXiv preprint arXiv:2503.06343*, 2025.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International conference on machine learning*, pp. 2170–2179. PMLR, 2019.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial intelligence*, 147(1-2):163–223, 2003.

Bram Grooten, Tristan Tomilin, Gautham Vasan, Matthew E Taylor, A Rupam Mahmood, Meng Fang, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Madi: Learning to mask distractions for generalization in visual deep reinforcement learning. *arXiv preprint arXiv:2312.15339*, 2023.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Qiyuan Liu, Qi Zhou, Rui Yang, and Jie Wang. Robust representation learning by clustering with bisimulation metrics for visual reinforcement learning with distractions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 8843–8851, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. *Advances in neural information processing systems*, 32, 2019.

Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004.

Sasha Salter, Dushyant Rao, Markus Wulfmeier, Raia Hadsell, and Ingmar Posner. Attention-privileged reinforcement learning. In *Conference on Robot Learning*, pp. 394–408. PMLR, 2021.

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

Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate mdp homomorphisms. *Advances in Neural Information Processing Systems*, 21, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Tongzhou Wang, Simon S Du, Antonio Torralba, Phillip Isola, Amy Zhang, and Yuandong Tian. Denoised mdps: Learning world models better than the world itself. *arXiv preprint arXiv:2206.15477*, 2022a.

Xudong Wang, Long Lian, and Stella X Yu. Unsupervised visual attention and invariance for reinforcement learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6677–6687, 2021.

Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. *arXiv preprint arXiv:2206.13452*, 2022b.

Zizhao Wang, Caroline Wang, Xuesu Xiao, Yuke Zhu, and Peter Stone. Building minimal and reusable causal state abstractions for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15778–15786, 2024.

Haiping Wu, Khimya Khetarpal, and Doina Precup. Self-supervised attention-aware reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10311–10319, 2021.

Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. *arXiv preprint arXiv:2206.04672*, 2022.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

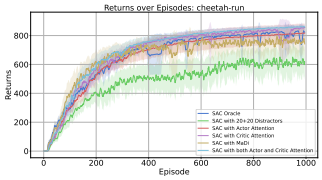
Huan Zhao, Junhua Zhao, Ting Shu, and Zibin Pan. Hybrid-model-based deep reinforcement learning for heating, ventilation, and air-conditioning control. *Frontiers in Energy Research*, 8:610518, 2021.

Qi Zhou, Jie Wang, Qiyuan Liu, Yufei Kuang, Wengang Zhou, and Houqiang Li. Learning robust representation for reinforcement learning with distractions by reward sequence prediction. In *Uncertainty in Artificial Intelligence*, pp. 2551–2562. PMLR, 2023.

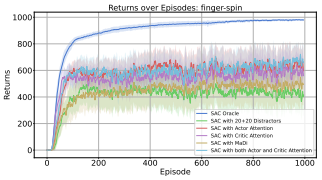
Supplementary Materials

The following content was not necessarily subject to peer review.

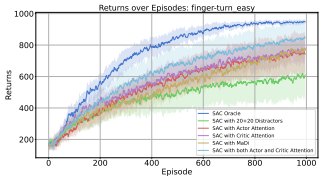
B Additional Results



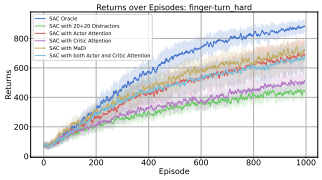
(a) cheetah-run



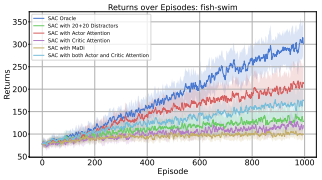
(b) finger-spin



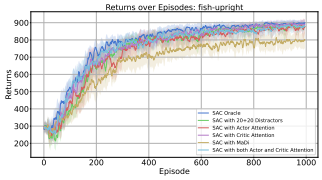
(c) finger-turneasy



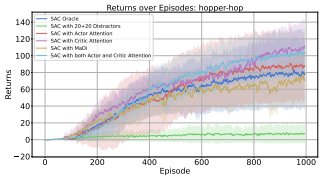
(d) finger-turnhard



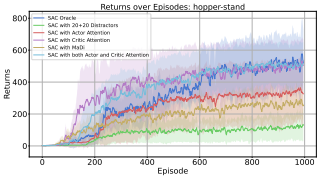
(e) fish-swim



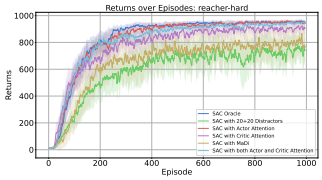
(f) fish-upright



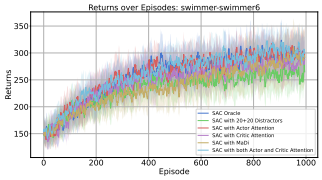
(g) hopper-hop



(h) hopper-stand

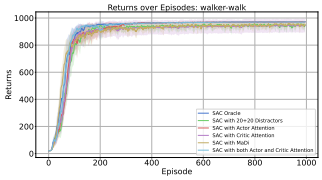


(i) reacher-hard



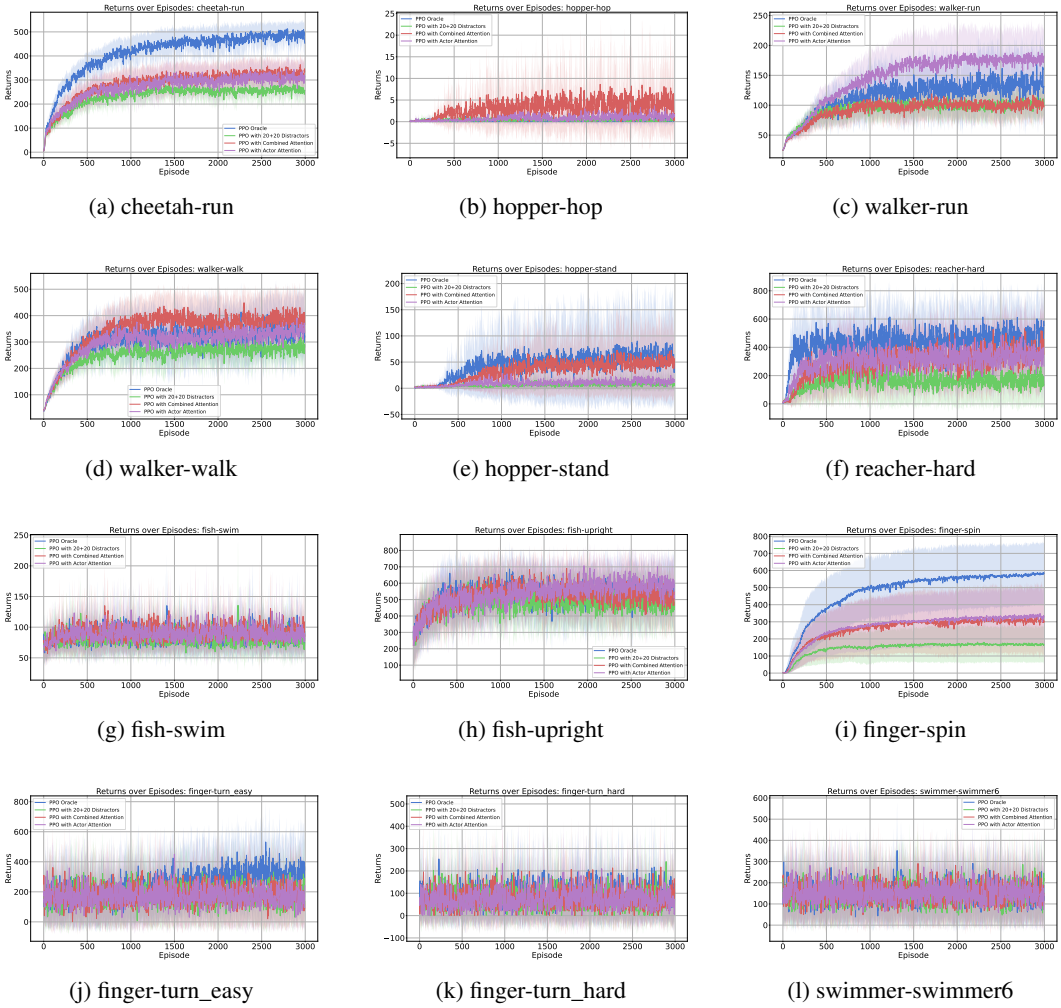
(j) swimmer-swimmer6

(k) walker-run



(l) walker-walk

Figure 6: Performance of SAC agents with and without the proposed observation-conditioned gating mechanism across various DM Control Suite tasks. The presence of distractors significantly impairs performance when no attention is applied. Incorporating our learned gating improves robustness and sample efficiency.



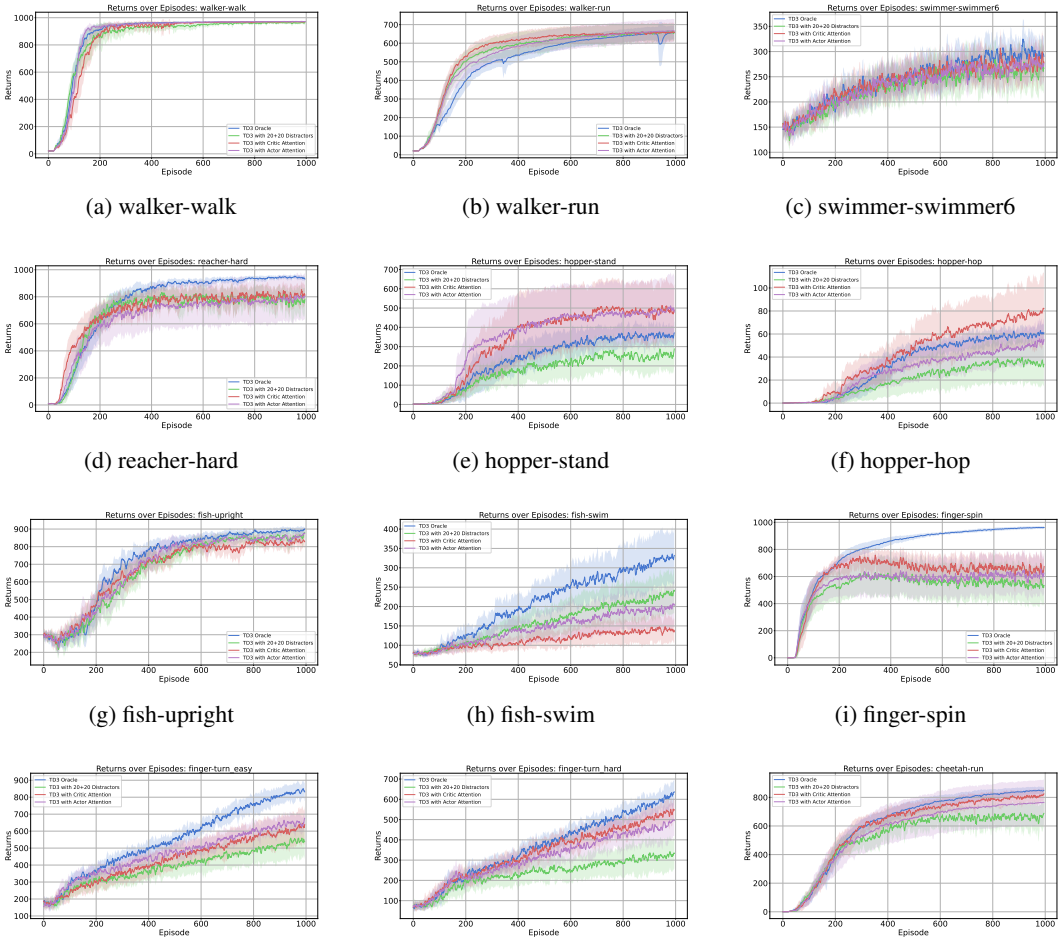
(a) cheetah-run (b) hopper-hop (c) walker-run

(d) walker-walk (e) hopper-stand (f) reacher-hard

(g) fish-swim (h) fish-upright (i) finger-spin

(j) finger-turn_easy (k) finger-turn_hard (l) swimmer-swimmer6

Figure 7: Performance of PPO with and without the proposed observation-conditioned gating mechanism across various DM Control Suite tasks. Plots are averaged over 10 random seeds



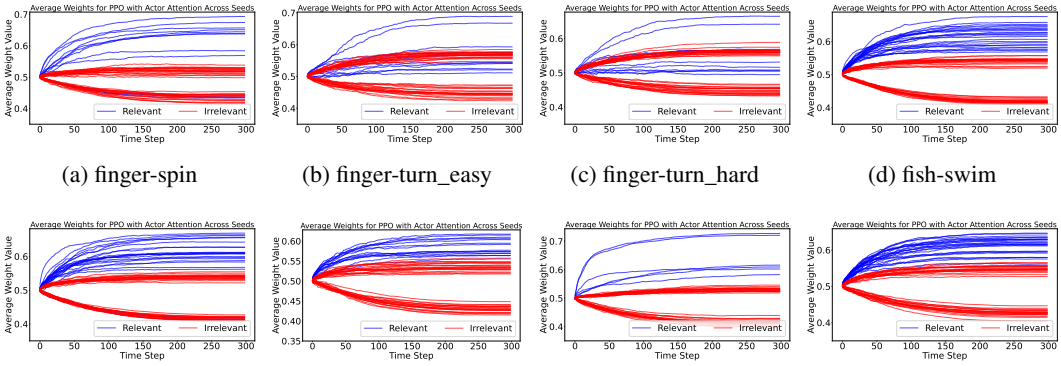
(a) walker-walk (b) walker-run (c) swimmer-swimmer6

(d) reacher-hard (e) hopper-stand (f) hopper-hop

(g) fish-upright (h) fish-swim (i) finger-spin

(j) finger-turn_easy (k) finger-turn_hard (l) cheetah-run

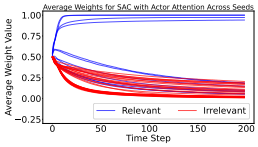
Figure 8: Performance of TD3 on the DM Control Suite tasks. Plots are averaged over 10 random seeds



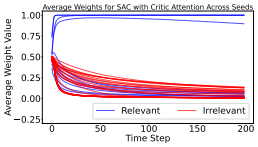
(a) finger-spin (b) finger-turn_easy (c) finger-turn_hard (d) fish-swim

(e) fish-upright (f) hopper-stand (g) reacher-hard (h) swimmer-swimmer6

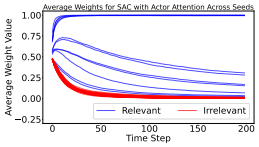
Figure 9: Masks for PPO. The task-relevant variables are plotted in blue while the task-irrelevant variables are plotted in red



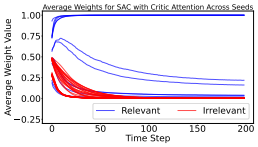
(a) finger-spin with actor loss



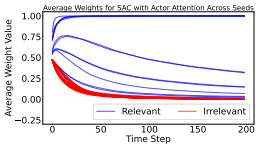
(b) finger-spin with critic loss



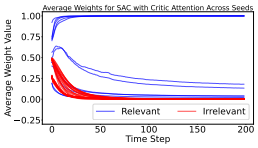
(c) finger-turn_easy with actor loss



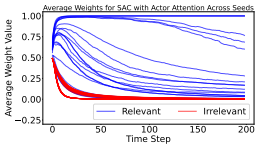
(d) finger-turn_easy with critic loss



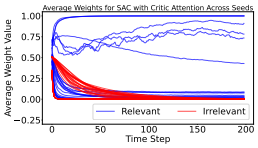
(e) finger-turn_hard with actor loss



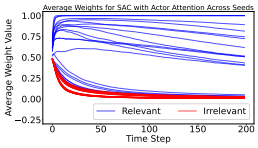
(f) finger-turn_hard with critic loss



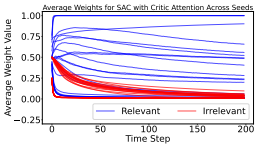
(g) fish-swim with actor loss



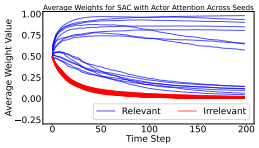
(h) fish-swim with critic loss



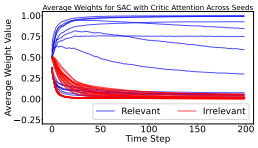
(i) fish-upright with actor loss



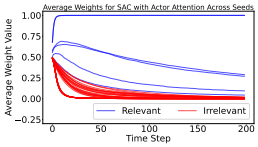
(j) fish-upright with critic loss



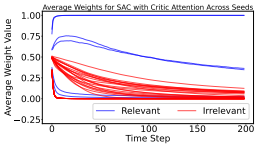
(k) hopper-stand with actor loss



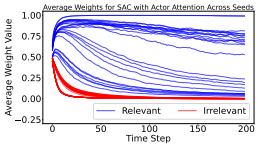
(l) hopper-stand with critic loss



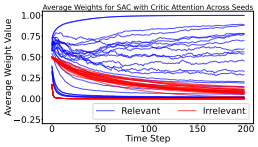
(m) reacher-hard with actor loss



(n) reacher-hard with critic loss

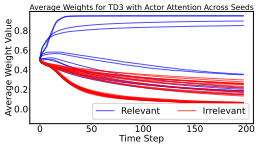


(o) swimmer-swimmer6 with actor loss

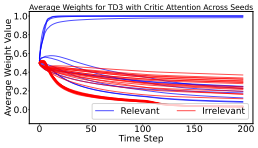


(p) swimmer-swimmer6 with critic loss

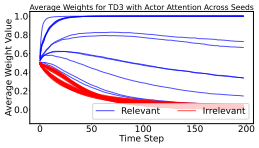
Figure 10: Masks for SAC



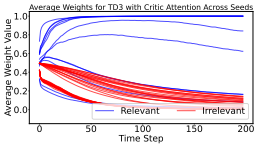
(a) finger-spin with actor loss



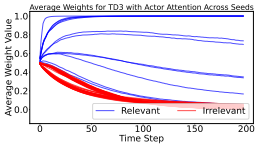
(b) finger-spin with critic loss



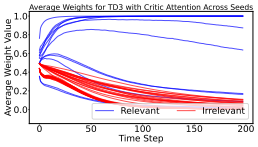
(c) finger-turn_easy with actor loss



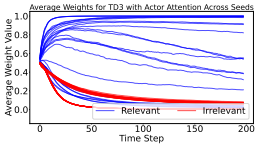
(d) finger-turn_easy with critic loss



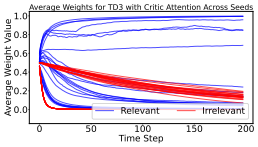
(e) finger-turn_hard with actor loss



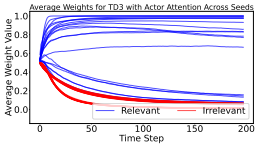
(f) finger-turn_hard with critic loss



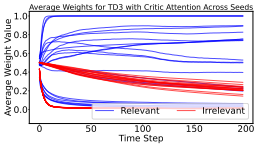
(g) fish-swim with actor loss



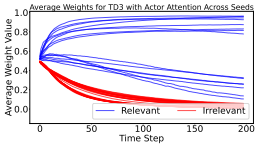
(h) fish-swim with critic loss



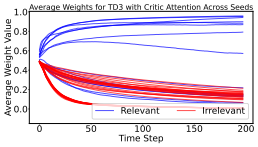
(i) fish-upright with actor loss



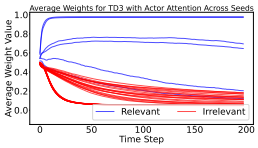
(j) fish-upright with critic loss



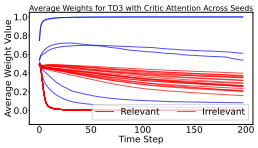
(k) hopper-stand with actor loss



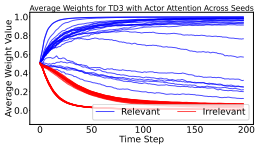
(l) hopper-stand with critic loss



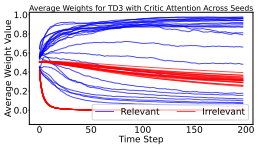
(m) reacher-hard with actor loss



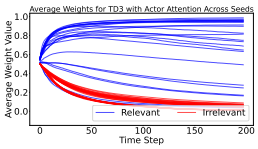
(n) reacher-hard with critic loss



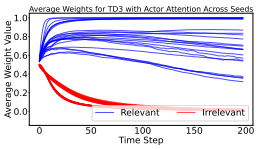
(o) swimmer-swimmer6 with actor loss



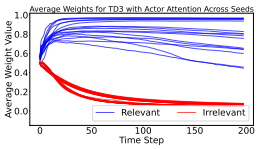
(p) swimmer-swimmer6 with critic loss



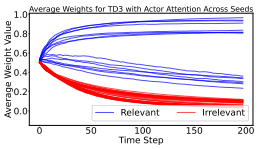
(q) walker-run with actor loss



(r) walker-walk with actor loss

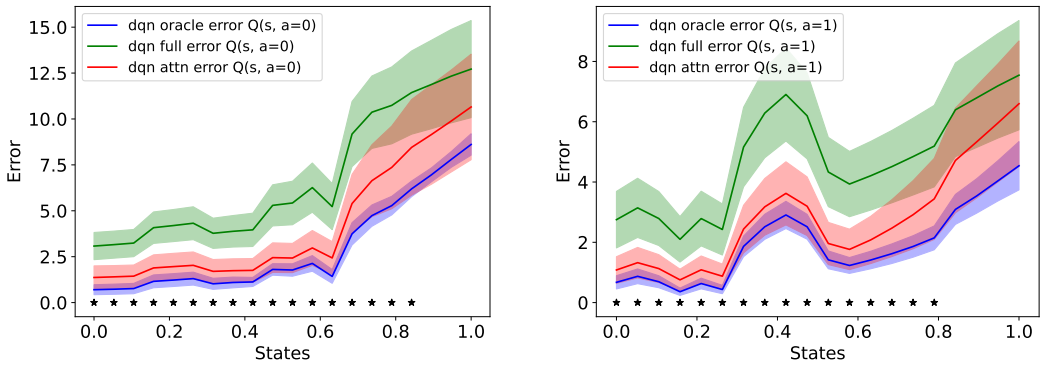


(s) cheetah-run with actor loss



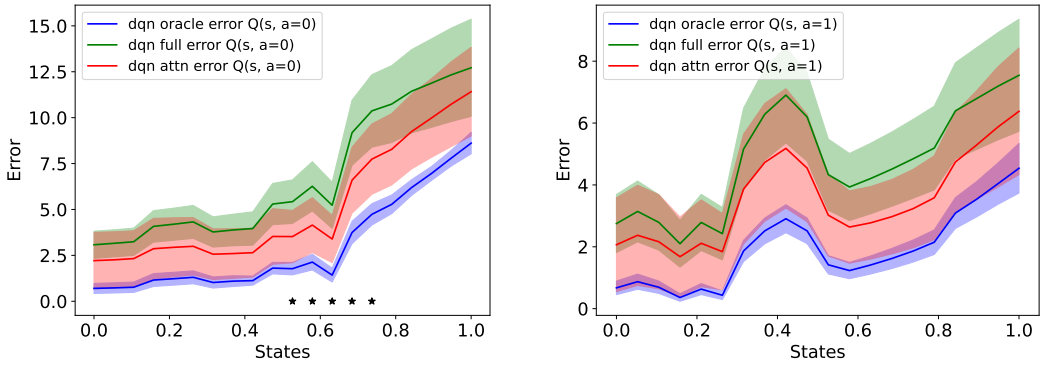
(t) hopper-hop with actor loss

Figure 11: Masks for TD3



action 0 action 1

Figure 12: Mean squared error of Q-value estimation for action 0 and 1 across oracle, full, and attention-based agents with sigmoid temperature 10.



action 0 action 1

Figure 13: Mean squared error of Q-value estimation for action 0 and 1 across oracle, full, and attention-based agents with sigmoid temperature 50.

C Hyperparameters

C.1 For SAC

Table 1: Key hyperparameters and architecture details of our SAC implementation.

Component	Setting
Environment	
Environment	DM Control Suite
Observation Dim	Original + 20 controllable + 20 uncontrollable distractors
Action Space	Continuous (Box)
Actor Network	
Architecture	MLP: 256-256, ReLU
Critic Networks (Q1/Q2)	
Architecture	MLP: 256-256, ReLU
Target Update	Soft update with $\tau = 0.005$
Training Setup	
Replay Buffer Size	5×10^6
Batch Size	256
Learning Starts	10,000 steps
Total Steps	1M
Policy LR	3×10^{-4}
Critic LR	1×10^{-3}
Optimizer	Adam
Max Grad Norm	10
Entropy Tuning	
α Tuning	Enabled (learned)
Distractors	
Controllable	Linear in action + bias
Uncontrollable	Random + bias
Evaluation & Logging	
Eval Episodes	50 (continuous)

C.2 PPO

Table 2: Key hyperparameters and architecture details of our PPO implementation.

Component	Setting
Environment	
Environment	DM Control Suite
Observation Dim	Original + 20 controllable + 20 uncontrollable distractors
Action Space	Continuous (Box)
Agent Architecture	
Policy/Value Network	Shared MLP: 64-64, `Tanh` activations
Training Setup	
Total Timesteps	3M
Rollout Length	2000 steps
Mini-batches	40
Update Epochs	10
Optimizer	Adam, LR = 3×10^{-4}
Annealed LR	Yes
Gradient Clipping	Max norm = 0.5
PPO Settings	
GAE Lambda	0.95
Discount Factor γ	0.99
Advantage Normalization	Enabled
Clip Coefficient	0.2
Clip Value Loss	Enabled
Entropy Coefficient	0.0
Value Loss Coef	0.5
Target KL	None
Distractor Variables	
Controllable	Linear in action + fixed bias
Uncontrollable	Random noise + fixed bias
Evaluation & Logging	
Eval	Returns from rollouts

C.3 TD3

Table 3: Key hyperparameters and architectural details of our TD3 implementation with shared attention.

Component	Setting
Environment	
Environment	DM Control Suite
Observation Dim	Original + 20 controllable + 20 uncontrollable distractors
Action Space	Continuous (Box)
Actor Network	
Architecture	MLP: 256-256, ReLU
Critic Networks (Q1, Q2)	
Architecture	MLP: 256-256, ReLU
Target Networks	Soft update with $\tau = 0.005$
Training Setup	
Total Timesteps	1M
Learning Starts	25K steps
Batch Size	256
Replay Buffer	Size = 10^6 transitions
Optimizer	Adam, LR = 3×10^{-4}
Gradient Clipping	Not used
TD3-Specific Settings	
Policy Delay	2 steps
Target Policy Noise	Std = 0.2, Clipped at 0.5
Exploration Noise	Std = 0.1 (added to actor output)
Distractor Variables	
Controllable	Linear in action + fixed bias
Uncontrollable	Random noise + fixed bias
Evaluation & Logging	
Eval Episodes	50 continuous episodes per checkpoint

D code

GitHub link:

https://github.com/nshmn/architectural-bias-for-state-abstraction