Where to Intervene: Action Selection in Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Deep reinforcement learning (RL) has gained widespread adoption in recent years but faces significant challenges, particularly in unknown and complex environments. Among these, *high-dimensional action selection* stands out as a critical problem. Existing works often require a sophisticated prior design to eliminate redundancy in the action space, relying heavily on domain expert experience or involving high computational complexity, which limits their generalizability across different RL tasks. In this paper, we address these challenges by proposing a general data-driven action selection approach with model-free and computational-friendly properties. Our method not only *selects minimal sufficient actions* but also *controls the false discovery rate* via knockoff sampling. More importantly, we seamlessly integrate the action selection into deep RL methods during online training. Empirical experiments validate the established theoretical guarantees, demonstrating that our method surpasses various alternative techniques in terms of both performances in variable selection and overall achieved rewards.

1 Introduction

Recent advances in deep reinforcement learning (RL) have attracted significant attention, with applications spanning numerous fields such as robotics, games, healthcare, and finance (Kober et al., 2013; Kaiser et al., 2019; Kolm & Ritter, 2020; Yu et al., 2021). Despite their ability to handle sequential decision-making, the practical utility of RL methods in real-world scenarios is often limited, especially in dealing with the high-dimensional action spaces (Sunehag et al., 2015; Kaiser et al., 2019; Sakryukin et al., 2020; Xiao et al., 2020). *High-dimensional action spaces* are prevalent in "black box" systems, characterized by overloaded actionable variables that are often *abundant and redundant*. Examples include precision medicine, where numerous combinations of treatments and dosages are possible (see e.g., Johnson et al., 2016; Liu et al., 2017); neuroscience, which involves various stimulation points and intensities (see e.g., Gershman et al., 2009); and robotics, particularly in muscle-driven robot control, where coordination of numerous muscles is required (see e.g., Schumacher et al., 2023). Nevertheless, these high-dimensional action spaces often contain many actions that are either ineffective or have negligible impact on states and rewards. Training RL models on the entire action space can result in substantial inefficiencies in both computation and data collection.

To handle high dimensionality, a promising approach is to employ automatic dimension reduction techniques to reduce the size of the action space, by selecting only *the essential minimum action set* necessary for effectively learning the environment and optimizing the policy based on the subspace. Having such a minimal yet sufficient action space can significantly enhance learning efficiency, as agents can thoroughly explore a more concise set of actions (Zahavy et al., 2018; Kanervisto et al., 2020; Jain et al., 2020). Moreover, a smaller action space can reduce computational complexity, a notable benefit in deep RL, where neural networks are used for function approximation (Sun et al., 2011; Sadamoto et al., 2020). In practical scenarios, eliminating superfluous actions saves the cost of extensive measurement equipment and thus allows a more comprehensive exploration of available actions. Yet, existing works often require a sophisticated prior design to eliminate redundancy in the action space (e.g., Synnaeve et al., 2019; Jiang et al., 2019; Farquhar et al., 2020; Luo et al., 2023), relying heavily on domain expert experience or involving high computational complexity, limiting their generalizability across different RL tasks.

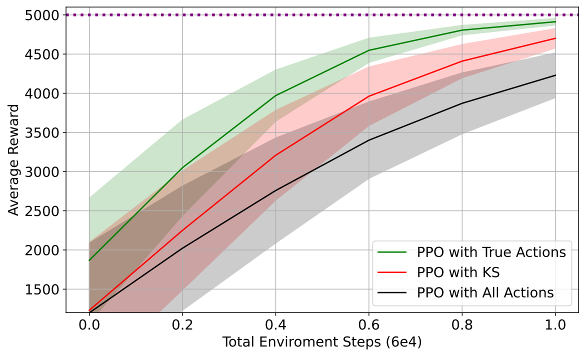


Figure 1: Average rewards under three proximal policy optimization (PPO) methods in a synthetic environment with 54 actions (among which only 4 actions influence states and rewards). The green line refers to the PPO trained based on the true influential actions, the red refers to the PPO with the estimated minimal sufficient actions by the proposed variable selection (KS), the black line represents the PPO with the entire redundant action space, and the dashed line is the optimal reward. The red line outperforms the black line indicating *the effectiveness of the variable selection* step.

In this paper, we propose a general data-driven action selection approach to identify the minimum sufficient actions in the high-dimensional action space. To handle the complex environments often seen in deep RL, we develop a novel variable selection approach called knockoff sampling (KS) for online RL, with theoretical guarantees of *false discovery rate control*, inspired by the model-free knockoff method (Candes et al., 2018). The effectiveness of this action selection method is demonstrated in Fig. 1. Here, a proximal policy optimization (PPO) method (Schulman et al., 2017) using variable selection outperforms the one trained on the entire action space and shows comparable performance to the PPO trained with the pre-known true minimal sufficient action. To remain computational-friendly, we design an adaptive strategy with a simple mask operation that seamlessly integrates this action selection method into deep RL methods during online training.

Our main **contributions** are fourfold:

• Conceptually, this work pioneers exploring high-dimensional action selection in online RL. We formally define *the sufficient action set* encompassing all influential actions, and *the minimal sufficient action set*, which contains the smallest number of actions necessary for effective decision-making.

• Methodologically, our method *bypasses the common challenge of creating accurate knockoff features in model-free knockoffs*. We use the established distribution of actions from the current policy network in online RL to precisely resample action values, producing exact knockoff features.

• Algorithm-wise, to *flexibly integrate arbitrary variable selection into deep RL* and eliminate the need to initialize a new RL model after the selection, we design a binary hard mask approach based on the indices of selected actions. This efficiently neutralizes the influence of non-chosen actions.

• Theoretically, to *address the issues of highly dependent data* in online RL, we couple our KS method with sample splitting and majority vote; under commonly imposed conditions, we theoretically show our method *consistently* identifies the minimal sufficient action set with false discovery rate control.

1.1 Related Works

Deep reinforcement learning has made significant breakthroughs in complex sequential decision-making across various tasks (Mnih et al., 2013; Silver et al., 2016; Schulman et al., 2017; Haarnoja et al., 2018). Yet, several considerable obstacles exist when dealing with high-dimensional spaces using deep RL. In terms of *high dimensional state space*, the state abstraction (Misra et al., 2020; Pavse & Hanna, 2023) has been studied to learn a mapping from the original state space to a much smaller abstract space to preserve the original Markov decision process. Yet, these methods such as bisimulation can be computationally expensive and challenging when the state space is very large or has complex dynamics (Ruan et al., 2015). Tied to our topic, it is *hard to utilize such abstraction-based methods to implement transformed actions*. This redirects us to *variable selection* on the redundant state space (see e.g., Kroon & Whiteson, 2009; Guo & Brunskill, 2017). Recently, Hao et al. (2021) combined LASSO with fitted Q-iteration to reduce states; following this context, Ma et al. (2023) employs the knockoff method for state selection but with discrete action spaces. However, all these works focus on the high dimensional state space in offline data, while our method aims to extract sufficient and necessary actions during online learning.

For RL with the high-dimensional action space, especially for continuous actions, some studies (Synnaeve et al., 2019; Farquhar et al., 2020) transformed the continuous control problem into the combinatorial action problem, by discretizing large action spaces into smaller subspaces. However, this transformation can lead to a significant loss of precision and hence produce suboptimal solutions (Lee et al., 2018; Tan et al., 2019). Other works (see e.g., Jiang et al., 2019; Luo et al., 2023) focused on muscle control tasks and used architectures reducing the action dimensionality before deploying RL methods. One recent study by Schumacher et al. (2023) combined differential extrinsic plasticity with RL to control high-dimensional large systems. Yet, all these works require specialized data collection, known joint ranges of actions, forced dynamics, or desired behaviors of policies, before implementing RL. In contrast, our method is entirely *data-driven without prior knowledge of environments* and thus can be generalized to tasks beyond muscle control. Some studies (Zahavy et al., 2018; Zhong et al., 2024) have also explored eliminating actions; however, their approaches are limited to discrete action spaces.

Variable selection, also known as feature selection, is a critical process to choose the most relevant variables representing the target outcome of interest, enhancing both model performance and interpretation. Over the past few decades, many well-known methods have been established, ranging from classical LASSO, Fisher score, and kernel dimension reduction (Tibshirani, 1996; Gu et al., 2012; Chen et al., 2017), towards deep learning (Liang et al., 2018; Balın et al., 2019; Lee et al., 2021). Yet, these works either *suffer from model-based constraints or lack theoretical guarantees.* **The model-X knockoff method** proposed by Candes et al. (2018) aims to achieve both goals via a general variable selection framework for black-box algorithms with guarantees of false discovery rate control. Due to its model-agnostic nature, the knockoff method has been extended to complement a wide range of variable selection approaches (Sesia et al., 2017; Ma et al., 2021; Liu et al., 2022). The main price or central challenge within the knockoff method lies in *the generation of faithful knockoff features.* Existing techniques either using model-specific methods (see e.g., Sesia et al., 2017; Liu & Zheng, 2018) that assume the underlying covariate distribution, or model-free approaches (see e.g., Jordon et al., 2018; Romano et al., 2020) that utilize deep generative models to obtain knockoffs without further assumptions on feature distribution. Owing to the blessing of online RL, our method bypasses this challenge through the known joint distribution of actions represented by the ongoing policy network, and thus can easily resample the action values to create exact knockoff features.

2 Problem Setup

2.1 Notations

Consider a Markov Decision Process (MDP) characterized by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, in which both the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are continuous. The state transition probability, denoted as $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$, is an unknown probability density function that determines the likelihood of transitioning to a next state $\mathbf{s}_{t+1} \in \mathcal{S}$, given the current state $\mathbf{s}_t \in \mathcal{S}$ and the action $\mathbf{a}_t \in \mathcal{A}$. The environment provides a reward, bounded within $[r_{\min}, r_{\max}]$, for each transition, expressed as $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$. The discount factor, represented by $\gamma \in (0, 1)$, influences the weighting of future rewards. We denote a generic tuple consisting of the current state, action, reward, and subsequent state as $(\mathbf{S}_t, \mathbf{A}_t, R_t, \mathbf{S}_{t+1})$. The Markovian property of MDP is that given the current state $\mathbf{S}_t$ and action $\mathbf{A}_t$, the current R_t and the next state $\mathbf{S}_{t+1}$, are conditionally independent of the past trajectory history. Consider $\mathbf{A}_t \in \mathbb{R}^p$ where p is very large indicating a *high dimensional action space.* We utilize $\rho_\pi(\mathbf{s}_t)$ and $\rho_\pi(\mathbf{s}_t, \mathbf{a}_t)$ to denote the state and state-action marginal distributions, respectively, of the trajectory distribution generated by a policy $\pi(\mathbf{a}_t \mid \mathbf{s}_t)$. The notation $J(\pi)$ is used to represent the expected discounted reward under this policy: $J(\pi) = \sum_t \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[\gamma^t r\left(\mathbf{s}_t, \mathbf{a}_t\right) \right].$

The goal of RL is to maximize the expected sum of discounted rewards above. This can be extended to a more general maximum entropy objective with the expected entropy of the policy over $\rho_\pi(\mathbf{s}_t)$.

2.2 Minimal Sufficient Action Set in Online RL

To address the high dimensional action space, we propose to utilize the variable selection instead of representation for practical usefulness. To achieve this goal, we first formally define the minimal sufficient action set. Denote the subvector of $\mathbf{A}_t$ indexed by components in G as $\mathbf{A}_{t,G}$ with an index set $G \subseteq \{1, 2, \dots, p\}$. Let $G^c = \{1, \dots, p\} \backslash G$ be the complement of G.

Definition 2.1. (Sufficient Action Set) We say G is the sufficient action (index) set in an MDP if

$$R_t \perp \mathbf{A}_{t,G^c} \mid \mathbf{S}_t, \mathbf{A}_{t,G}, \qquad \mathbf{S}_{t+1} \perp \mathbf{A}_{t,G^c} \mid \mathbf{S}_t, \mathbf{A}_{t,G}, \qquad \text{for all } t \geq 0.$$

The sufficient action set can be seen as a sufficient conditional set to achieve past and future independence. The sufficient action set may not be unique.

Definition 2.2. (Minimal Sufficient Action Set) We say G is the minimal sufficient action set in an MDP if it has the smallest cardinality among all sufficient action sets.

Unlike the sufficient action set, there is only one unique minimal sufficient action set to achieve conditional independence if there are no identical action variables in the environment. We also call G^c the redundant set when G is the minimal sufficient action (index) set. Here, to achieve such a minimal sufficient action set, one should also require the states $\mathbf{S}_t$ be the sufficient states, so there is no useless state (Ma et al., 2023) to introduce related redundant actions that possibly lead to ineffective exploration or data inefficiency. Without loss of generality, we assume sufficient states throughout this paper and focus on eliminating redundant actions in a high-dimensional action space. Our goal is to identify the minimal sufficient action set for online deep reinforcement learning to improve exploration.

2.3 Preliminary: Knockoff Variable Selection

Without making additional assumptions on the dependence among variables, in this work, we utilize the model-X knockoffs (Candes et al., 2018) for flexible variable selection, which ensures finite-sample control of the false discovery rate (FDR). We first briefly review *the model-X knockoffs (Candes et al., 2018) in the supervised regression setting with independent samples*, which will be leveraged later *as the base variable selector of our proposed method for dependent data in online RL setting.* Specifically, given n independent observations, consider $\boldsymbol{Y}$ as a n-dimensional response vector and $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_p)$ as an $n \times p$ matrix of covariates. The knockoff inference aims to identify significant covariates that influence the outcome while controlling FDR. Towards this goal, the model-X knockoff generates an $n \times p$ matrix $\widetilde{\boldsymbol{X}} = (\widetilde{\boldsymbol{x}}_1, \cdots, \widetilde{\boldsymbol{x}}_p)$ as *knockoff features* that have the similar properties as the collected covariates. This matrix is constructed by the joint distribution of $\boldsymbol{X}$ and satisfy:

$$\widetilde{\boldsymbol{X}} \perp \boldsymbol{Y} \mid \boldsymbol{X} \quad \text{and} \quad (\boldsymbol{X}, \widetilde{\boldsymbol{X}})_{\mathrm{swap}(\Omega)} \overset{d}{=} (\boldsymbol{X}, \widetilde{\boldsymbol{X}}), \tag{1}$$

for each subset Ω within the set $\{1, \cdots, p\}$, where swap(Ω) indicates the operation of swapping such that for each $j \in \Omega$, the j-th and $(j+p)$-th columns are interchanged. The notation $\overset{d}{=}$ signifies equality in distribution. After obtaining knockoff features, let $\widetilde{\mathcal{D}} = \{\boldsymbol{X}, \widetilde{\boldsymbol{X}}, \boldsymbol{Y}\}$ denote an augmented dataset and we can calculate the feature importance scores Z_j and $\widetilde{Z}_j$ for each variable $\boldsymbol{x}_j$ and its corresponding knockoff $\widetilde{\boldsymbol{x}}_j$. Define the function $f : \mathbb{R}^2 \to \mathbb{R}$ as an anti-symmetric function, meaning that $f(u, v) = -f(v, u)$ for all $u, v \in \mathbb{R}^2$, e.g., $f(u, v) = u - v$. Set $W_j = f(Z_j, \widetilde{Z}_j)$ in such a way that higher values of W_j indicate stronger evidence of the significance of $\boldsymbol{x}_j$ being influential covariate. The j-th variable is selected if its corresponding W_j is at least a certain threshold τ_α when the target FDR level is α. For example, the set of chosen variables can be represented as $\widehat{\mathcal{I}} = \{j : W_j \geq \tau_\alpha\}$, where

$$\tau_\alpha = \min \left\{ \tau > 0 : \frac{\# \{j \in [p] : W_j \leq -\tau\}}{\# \{j \in [p] : W_j \geq \tau\}} \leq \alpha \right\}. \tag{2}$$

3 Online Deep RL with Variable Selection

To identify the minimal sufficient action set in online deep RL, we integrate the action selection into RL to find truly influential actions during the training process. Its advantages are manifold. Firstly, its model-agnostic nature ensures compatibility across various RL architectures and algorithms. Moreover, its data-driven characteristic allows for straightforward application across diverse scenarios, thereby increasing practical utility. Crucially, the action selection boosts the explainability and reliability of RL systems by clearly delineating actions that contribute to model performance. In the following, we first introduce an action-selected exploration strategy for online deep RL in Section 3.1, followed by the model-free knockoff-sampling method for action selection in Section 3.2.

3.1 Action-Selected Exploration Algorithm

We propose an innovative action-selected exploration for deep RL. Suppose at a predefined time step $t = T_{vs}$, a set of actions $\widehat{G}$ is identified from the buffered data, where the cardinality of $\widehat{G}$ ($|\widehat{G}|$) is d, with $d \leq p$ indicting a possibly reduced dimension. A critical challenge arises in leveraging the insights gained from action selection for updating the deep RL models. The conventional approach of constructing an entirely new model based on the selected actions is not only time-consuming but also inefficient, particularly in dynamic, non-stationary environments where the requisite action sets are subject to frequent changes. Although our study primarily focuses on stationary environments, the inefficiency of model reinitialization post-selection remains a notable concern.

To seamlessly and efficiently integrate action selection results into deep RL, we propose to mask the non-selected actions and remove their influence once a hard mask is constructed, and thus *is flexible to integrate with arbitrary variable selection method.* Specifically, in continuous control tasks, deep RL algorithms utilize a policy network π_θ to sample a certain action $\mathbf{a}$ given current state $\mathbf{s}$, namely $\mathbf{a} \sim \pi_\theta(\cdot \mid \mathbf{s})$. Here, we use the Gaussian policy as an illustrative example, but it can be flexibly generalized to other distributions. Assume the policy network is parameterized by a multivariate Gaussian with the diagonal covariance matrix as:

$$\mathbf{a} \sim \mathcal{N}\left(\mu(\mathbf{s}), \operatorname{diag}(\sigma(\mathbf{s}))^2\right),$$

where μ and σ are parameterized functions to output mean and standard deviations. Each time we obtain an action from the policy network. The updates of the policy network π_θ and the action-value function Q_ϕ usually involve sampled actions $\mathbf{a}_t$ and $\log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$ which is the log density of sampled actions. Our strategy is using *a binary mask* to set them as a certain constant value during the forward pass and it will block the gradient when doing backpropagation and also remove influence when fitting a function. Given a selected action set $\widehat{G}$, we focus on integrating this selection into the model components $Q_\phi(\mathbf{a}, \mathbf{s})$ and $\pi_\theta(\mathbf{a} \mid \mathbf{s})$. To facilitate this, we define a selection vector $\mathbf{m} = (m_1, \cdots, m_p) \in \{0,1\}^p$, where $m_i = 1$ if $i \in \widehat{G}$ and 0 otherwise. This vector enables the application of a selection mask to both the Q_ϕ and π_θ as follows.

For Q_ϕ, we use the hard mask to remove the influence of non-selected actions during Q function fitting,

$$Q_\phi^m(\mathbf{a}, \mathbf{s}) = Q_\phi(\mathbf{m} \odot \mathbf{a}, \mathbf{s}), \tag{3}$$

where $\odot$ is the element-wise product. The adoption of action selection significantly reduces the dimensionality of the input action space, thereby reducing bias in the Q function fitting.

For π_θ, considering the necessity of updating the policy network via policy gradient, we integrate a hard mask into the logarithm of the policy probability. The modified log probability is formulated as

$$\log \pi_\theta^m(\mathbf{a} \mid \mathbf{s}) = \mathbf{m} \cdot \left(\log \pi_\theta(a_1 \mid \mathbf{s}), \ldots, \log \pi_\theta(a_p \mid \mathbf{s})\right), \tag{4}$$

where $\cdot$ is the dot product. This masking of the log probability helps mitigate the likelihood of encountering extremely high entropy values, thereby facilitating a more stable and efficient training process. We demonstrate the integration of action selection into deep RL as detailed in Algorithm 1.

Remark 3.1. Here, we focus on the case that actions are parameterized as diagonal Gaussian which are conditionally independent given states. However, our method can be easily extended to the correlated actions, with details provided in Appendix D.

Algorithm 1 Action-Selected Exploration in Reinforcement Learning

```
Require: FDR rate $\alpha$, majority voting ratio $\Gamma$, max steps $T$, variable selection step $T_{vs}$
  Begin: Initialize the selection set $\widehat{G} = \{\}$, policy $\pi_\theta$, value function parameter $\phi$, augmented replay buffer $\mathcal{D}$
  while steps smaller than $T$ do
    Sample $\mathbf{a}_t \sim \pi_\theta(\cdot \mid \mathbf{s}_t)$
    Sample knockoff copy $\widetilde{\mathbf{a}}_t \sim \pi_\theta(\cdot \mid \mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim \mathrm{Env}(\mathbf{a}_t, \mathbf{s}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{s}_t, \mathbf{a}_t, \widetilde{\mathbf{a}}_t, r_t, \mathbf{s}_{t+1}\}$
    if $t = T_{vs}$ then
      Utilize a variable selection algorithm (optional: Knockoff-Sampling in Algorithm 2) on $\mathcal{D}$ to obtain
      the estimated minimal sufficient action set $\widehat{G}$
      Generate a mask $\mathbf{m}$ based on $\widehat{G}$ to prune RL networks based on equation 3 and equation 4
    end if
    if it's time to update then
      update $\phi$ and $\theta$ based on the specific RL algorithm used
    end if
  end while
```

Algorithm 2 Knockoff-Sampling Variable Selection

```
Require: FDR rate $\alpha$, majority voting ratio $\Gamma$, data buffer $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \widetilde{\mathbf{a}}_t, r_t, \mathbf{s}_{t+1})\}_{t=1}^{T_{vs}}$
  Split $\mathcal{D}$ into non-overlapping sets $\{\mathcal{D}_k\}_{k=1}^{K}$ and let $\mathbf{y}_t = (r_t, \mathbf{s}_{t+1})$ as the response vector
  for $k = 1, \ldots K$ do
    for $i$-th dimension in $\{\mathbf{y}_t\}_{t=1}^{T_{vs}}$ do
      Apply a machine learning algorithm to all $(\mathbf{s}_t, \mathbf{a}_t, \widetilde{\mathbf{a}}_t, \mathbf{y}_t) \in \mathcal{D}_k$ to construct feature importance statistics
      $Z_{j,i}$ and $\widetilde{Z}_{j,i}$ for the $j$-th action and its knockoff copy, respectively, for each $j \in [p]$.
    end for
    for each $j \in [p]$ do
      Set $Z_j = \max_i Z_{j,i}, \quad \widetilde{Z}_j = \max_i \widetilde{Z}_{j,i}, \quad$ and $\quad W_j = f\left(Z_j, \widetilde{Z}_j\right)$
    end for
    Utilize the threshold $\tau_\alpha$ defined in equation 2, and get $\widehat{G}_k = \{j \in [p] : W_j \geq \tau_\alpha\}$
  end for
  return $\widehat{G} := \left\{ j \in \{1, \ldots, p\} : \sum_{k=1}^{K} \mathbb{I}\left(j \in \widehat{G}_k\right) \geq K\Gamma \right\}$
```

Remark 3.2. In scenarios where the algorithm exclusively employs the state-value function $V_\phi(\mathbf{s})$, the use of the mask operation is unnecessary. Our empirical studies suggest that, even without masking, the model maintains robust performance. This implies that updates to the policy network may hold greater significance than those to the critic in certain contexts.

3.2 Knockoff-Sampling for Action Selection

Despite the large volume of variable selection (VS) methods (see e.g., Tibshirani, 1996; Gu et al., 2012; Chen et al., 2017; Liang et al., 2018; Balın et al., 2019; Lee et al., 2021), these works either *suffer from model-based constraints or lack theoretical guarantees.* The traditional VS often identifies unimportant actions, leading to a high false discovery rate and further causing performance degeneration, as shown in Fig. 2. To provide a general action selection approach for deep RL with false discovery rate control, we propose a novel knockoff-sampling (KS) method that handles dependent data in the online setting with a model-agnostic nature as follows.

Suppose now we have a data buffer with the size M, collected from N trajectories where each trajectory has length T_j for $j = 1, \ldots N$ and $\sum_{j=1}^{N} T_j = M$. Each time we obtain an action from the policy network,

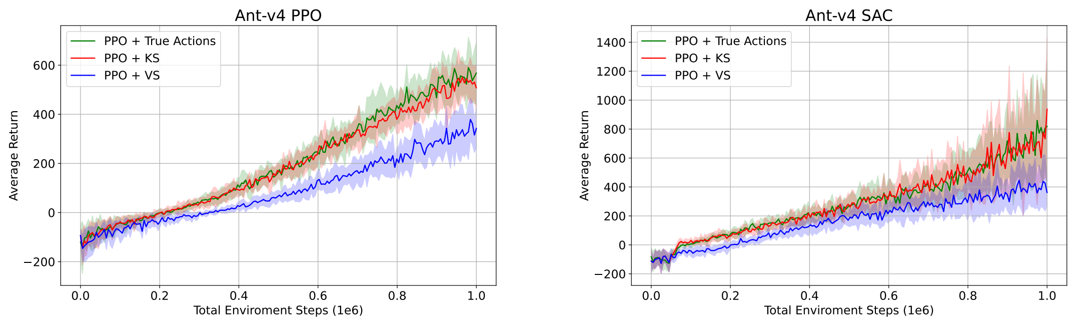


Figure 2: Learning curves in the Ant-v4 environment reveal that the Knockoff Sampling (KS) method outperforms the traditional Variable Selection (VS) method. When implemented with either the Proximal Policy Optimization (PPO) or Soft-Actor-Critic (SAC) algorithm, KS achieves performance comparable to that of the true actions by automatically setting optimal thresholds to filter out redundant actions, whereas VS often selects useless actions, leading to a high false discovery rate.

we also resample a knockoff copy conditional on the same state $\widetilde{\mathbf{a}}_t \sim \pi_\theta(\cdot \mid \mathbf{s}_t)$, and append it to the buffer. The transition tuples thus is redefined as $(\mathbf{S}_t, \mathbf{A}_t, \widetilde{\mathbf{A}}_t, R_t, \mathbf{S}_{t+1})$. Note that steps within each trajectory are temporally dependent. To *address the issues of highly dependent data* in online RL, we couple our method with sample splitting and majority vote following Ma et al. (2023). The proposed KS method consists of three steps as summarized in Algorithm 2: 1. Sample Splitting; 2. Knockoff-Sampling Variable Selection; 3. Majority Vote. We detail each step below.

1. **Sample Splitting:** We first split all transition tuples $(\mathbf{S}_t, \mathbf{A}_t, \widetilde{\mathbf{A}}_t, R_t, \mathbf{S}_{t+1})$ into K non-overlapping sub-datasets. This process results in a segmentation of the dataset $\mathcal{D}$ into distinct subsets $\mathcal{D}_k$ for $k \in [K]$. We combine response variables and denote $\mathbf{Y}_t = (R_t, \mathbf{S}_{t+1})$ to simplify the notation, based on the target outcomes in Definition 2.1. Here, each sequence $(\mathbf{S}_t, \mathbf{A}_t, \widetilde{\mathbf{A}}_t, \mathbf{Y}_t)$ is assigned to $\mathcal{D}_k$ if $t \mod K = k-1$. Subsequent to this division, any two sequences located within the same subset $\mathcal{D}_k$ either originate from the same trajectory with a temporal separation of no less than K or stem from different trajectories. If the system adheres to β-mixing conditions (Bradley, 2005), then a careful selection (Berbee, 1987) can allow us to assert that transition sequences within each subset $\mathcal{D}_k$ is approximately independent.

2. **Knockoff-Sampling Variable Selection:** For each data subset $\mathcal{D}_k$, we select a minimal sufficient action set using the model-X knockoffs as the base selector. Unlike the knockoff method detailed in Section 2.3 that either constructs knockoff features based on second-order machines or estimates the full distribution, we directly sample a knockoff copy of actions from the policy network, i.e., $\widetilde{\mathbf{a}}_t \sim \pi_\theta(\cdot \mid \mathbf{s}_t)$. This helps us to *bypass the common challenge of creating accurate knockoff features in model-free knockoffs*. We theoretically validate that the sampled knockoffs in online RL meet the swapping property equation 1 in Section 4. For every single dimension i of the outcome vector $\mathbf{Y}_t = (R_t, \mathbf{S}_{t+1})$, we use a general machine learning method (e.g., LASSO, random forest, neural networks) to provide variable importance scores $Z_{j,i}$ and $\widetilde{Z}_{j,i}$ for the j-th dimension of actions and its knockoff copy, respectively. By the maximum score $Z_j = \max_i Z_{j,i}$ and $\widetilde{Z}_j = \max_i \widetilde{Z}_{j,i}$, the selected action set $\widehat{G}_k$ is then obtained following the same procedure in Section 2.3.

3. **Majority Vote:** To combine the results on the whole K folds, we calculate the frequency of subsets where the j-th action is chosen, i.e., $\widehat{p}_j = \sum_{k=1}^{K} \mathbb{I}(j \in \widehat{G}_k)/K$, and establish the ultimate selection of actions $\widehat{G} = \{j : \widehat{p}_j \geq \Gamma\}$, with Γ being a predetermined cutoff between 0 and 1.

4 Theoretical Results

Without loss of generality, we assume that the data buffer $\mathcal{D}$ consisting of N i.i.d. finite-horizon trajectories, each of length T, which can be summarized as NT transition tuples. We first define two properties to establish theoretical results.

Definition 4.1. (**Flip Sign Property for Augmented Data**) property on the augmented data matrix $\mathcal{D}_k = \left[\mathbf{A}_k, \tilde{\mathbf{A}}_k, \mathbf{S}_k, \mathbf{Y}_k\right]$ if for any $j \in [p]$ and $\Omega \subset [p]$,

$$W_i \left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right]_{\text{swap}(\Omega)}, \mathbf{S}_k, \mathbf{Y}_k \right) = \begin{cases} -W_i \left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right], \mathbf{S}_k, \mathbf{Y}_k \right), & \text{if } j \in \Omega, \\ W_i \left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right], \mathbf{S}_k, \mathbf{Y}_k \right), & \text{otherwise,} \end{cases}$$

where $\mathbf{A}_k, \tilde{\mathbf{A}}_k \in \mathbb{R}^{(NT/K)\times p}$ denote the matrices of the actions and their knockoffs, $\mathbf{S}_k \in \mathbb{R}^{(NT/K)\times d}$ denote the matrice of state, $\mathbf{Y}_k \in \mathbb{R}^{(NT/K)\times d+1}$ denotes the response matrix, and $\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right]_{\text{swap}(\Omega)}$ is obtained by swapping all j-th columns in $\mathbf{A}_k, \tilde{\mathbf{A}}_k$ for $j \in \Omega$.

The above flip sign property is a common property that needs to be satisfied in knockoff-type methods. We show our method automatically satisfies this property in Lemma F.3 of Appendix.

Definition 4.2. (**Stationarity and Exponential β-Mixing**) The process $\{(\mathbf{S}_t, \mathbf{A}_t, R_t)\}_{t\geq 0}$ is stationary and exponentially β-mixing if its β-mixing coefficient at time lag k is of the order ρ^k for some $0 < \rho < 1$.

This exponential β-mixing condition has been assumed in the RL literature (see e.g., Antos et al., 2008; Dai et al., 2018) to derive the theoretical results for the dependent data. Such a condition quantifies the decay in dependence as the future moves farther from the past to achieve the dependence of the future on the past. Based on the above definitions, we establish the following false discovery control results of our method.

Theorem 4.3. *Set the number of sample splits $K = k_0 \log(NT)$ for some $k_0 > -\log^{-1} \rho$ where ρ is defined in Definition 4.2. Assume that the following assumption hold: the process $\{(\mathbf{S}_t, \mathbf{A}_t, R_t)\}_{t\geq 0}$ is stationary and exponentially β-mixing.*

Then $\widehat{G}_k$ obtained by Algorithm 2 with the standard knockoffs controls the modified FDR (mFDR),

$$mFDR \leq \alpha + O\left\{K^{-1}(NT)^{-c}\right\},$$

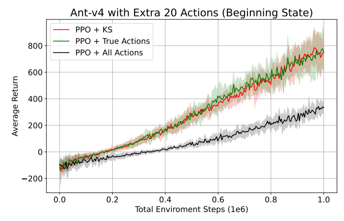
where the constant $c = -k_0 \log(\rho) - 1 > 0$.

The proof can be mainly divided into two parts, firstly we show valid mFDR control can be achieved when data are independent, then for dependent data satisfying the β-mixing condition, the upper band can be relaxed as the cost of dependence. Finally, a combination of the two would provide the final upper bound on mFDR control. The detailed proof is in Appendix F. This theorem provides a theoretical guarantee for controlling the modified false discovery rate in our proposed method.

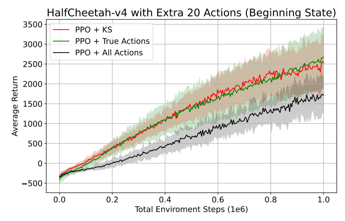
5 Experiments

Experiment Setup We aim to answer whether the variable selection is helpful for deep RL training when the action dimension is high and redundant. We conduct experiments on standard locomotion tasks in MuJoCo (Todorov et al., 2012) and treatment allocation tasks calibrated from electronic health records (EHR), the MIMIC-III dataset (Johnson et al., 2016). The environment details are in Table B.1 of Appendix. Here, we focus on two representative actor-critic algorithms, Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Soft-Actor-Critic (SAC) (Haarnoja et al., 2018). We adopt the implementation from Open AI Spinning up Framework (Achiam, 2018). For SAC, the implementation involves fitting both Q_ϕ and π_θ, and we use a mask on both components. For PPO, it fits V_θ and π_θ, hence we only combine the mask with π_θ. Tables A.1 and A.2 summarize the hyperparameters we used. We set the FDR rate $\alpha = 0.1$ and voting ratio $\Gamma = 0.5$ in all settings. All the experiments are conducted in the server with 4× NVIDIA RTX A6000 GPU.

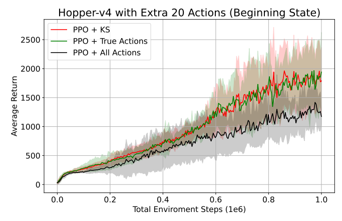
Semi-synthetic MuJoCo Environments We choose three tasks: Ant, HalfCheetah, and Hopper. To increase the dimension of action space, we artificially add extra p actions to the raw action space and consider two scenarios $p = 20$ and 50. For each setting, we run experiments over 2×10^5 and 10^6 steps for SAC and PPO, respectively, averaged over 10 training runs. The running steps for SAC and PPO are set adaptively to obtain better exploration for each method and save computation costs, as the main goal is to show how action selection can improve sample efficiency rather than compare these two methods. For each evaluation point, we run 10 test trajectories and average their reward as the average return. Besides RL algorithm



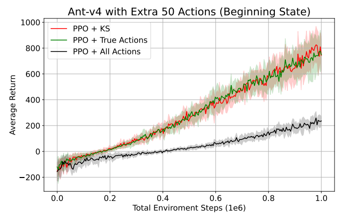
(a) PPO Ant $p = 20$ (Initial)



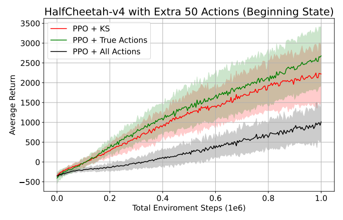
(b) PPO Cheetah $p = 20$ (Initial)



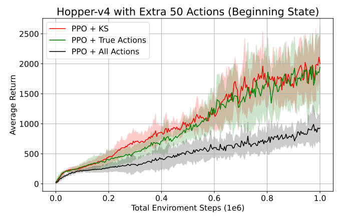
(c) PPO Hopper $p = 20$ (Initial)



(d) PPO Ant $p = 50$ (Initial)

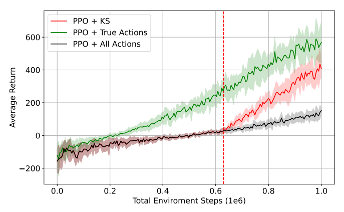


(e) PPO Cheetah $p = 50$ (Initial)

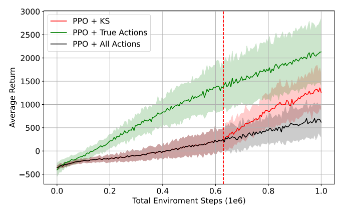


(f) PPO Hopper $p = 50$ (Initial)

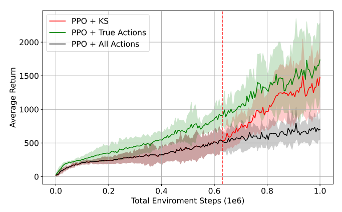
Figure 3: Learning curves for PPO in the MujoCo environments with different approaches during the initial stage. In all experiments, our knockoff sampling (KS) method not only performs comparably to the true actions but also consistently delivers higher rewards than using all actions.



(a) PPO Ant $p = 50$ (Middle)



(b) PPO Cheetah $p = 50$ (Middle)



(c) PPO Hopper $p = 50$ (Middle)

Figure 4: Learning curves for PPO in the MujoCo environments during the middle stage where the red line indicates the time point we utilize the proposed KS. After identifying the essential action set, the policy can be more efficient and achieve higher rewards than continuing training on all actions.

performance, we also evaluate variable selection performance in terms of True Positive Rate (TPR), False Positive Rate (FPR), and FDR.

Action Selection in the Initial Stage of Training We utilize action selection in the beginning stage of the training. For both methods, we utilize the first 4000 samples for variable selection and then use the selection results to build a hard mask for action in deep RL models. We compare our knockoff sampling (KS) method with the baseline of selecting all actions (All) to evaluate the impact of integrating a masking mechanism with a selection strategy in deep RL. We also provide the experimental results with only ground-truth actions selected (True) as a reference. To reduce the computational complexity, we choose LASSO (Tibshirani, 1996) as our base variable selection algorithm for KS. Here, selecting all actions (All) and ground-truth actions (True) are the cases where RL models trained on whole action space and minimal sufficient action space, respectively. Hence, the model corresponding to ground-truth action has smaller parameters than all other methods because its initialization is based on the minimal sufficient action set. The results are shown in Fig. 3 for PPO, Fig. B.1 for SAC, and Table 2 for all numerical details. Due to space constraints, we mainly present the PPO figures in the main text. In all cases, we find that KS-guided models outperform those trained on the whole action space in terms of average return and much lower FDR and FPR, with larger improvement gains as p increases. This empirically validates our theory of FDR control with the proposed KS method, demonstrating that action selection can enhance learning efficiency during the initial stages of RL training where action space is high and redundant.

Table 1: In the treatment allocation environments, we report the average reward and standard deviation (Std) at both the midpoint (50%) and the end (100%) of the training process, where the proposed KS performs the best with the highest reward. Both KS and All show significantly lower variance with improving performance over time. In contrast, Lattice and gSDE show high variance and degraded performance.

Method	Average Reward (Std)	
	50%	100%
KS	**14.0**(0.8)	**15.3**(1.2)
All	13.6(1.6)	14.4(1.0)
Lattice	12.7(1.7)	10.8(5.9)
gSDE	9.7(5.3)	9.3(5.3)

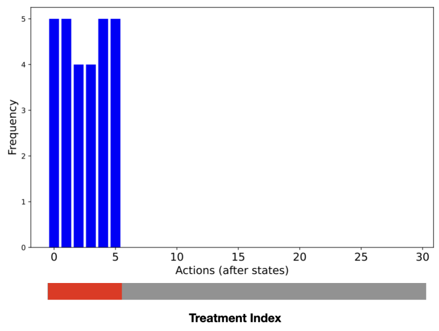


Figure 5: In the treatment allocation environments, our KS methods identify the most relevant treatments and the policy can be more efficient to achieve higher rewards than continuing training on all actions.

Action Selection in the Middle Stage of Training To show whether action selection can be used in the middle stage of training to remedy the inefficiency brought by exploring the whole action space, we conduct experiments where in the first half of the training steps the models are trained on the whole action space, and in the middle of the stage, we utilize action selection and build hard masks for them and then continue training for the rest of the steps. We compare our KS method with selecting all actions (All) and ground-truth actions (True) similarly. The results in Fig. 4 and Fig. B.2 reveal a notable pattern: agents initially struggle to learn effectively, but mid-stage variable selection significantly improves their performance, with models trained on the correct actions. This demonstrates the effectiveness of mid-stage variable selection in enhancing learning outcomes.

Treatment Allocation for Sepsis Patients We test our method with PPO and utilize the first 1000 samples for action selection in the initial stage. We also include two additional baselines: Lattice (Chiappa et al., 2024), and gSDE (Raffin et al., 2022). Lattice and gSDE also use all actions but additionally incorporate temporally correlated Gaussian noise into the training. We run experiments over 5×10^4 time steps and average results over 5 runs. For each evaluation point, we run 5 test trajectories and average their reward as the average return. The results are shown in Table 1 and Fig. 5. Lattice and gSDE exhibit high variance and instability in this environment, likely due to over-exploration. In contrast, our method consistently delivers stable and superior performance compared to the others. The analysis in Fig. 5 reveals that our methods effectively identify sepsis-influencing treatments that are relevant to key body factors, demonstrating the potential for real-world medical applications.

Action Selection is Fast and Lightweight With just a few thousand data points and a lightweight machine learning algorithm like random forest or LASSO, the whole action selection process outlined in Algorithm 2, completes in under 20 seconds—including knockoff threshold determination. This is significantly faster and less computationally intensive than the RL training part. Even when incorporating more sophisticated feature selection methods, the additional computational overhead remains *negligible* compared to the time required for RL training. Moreover, for the RL agent's deep neural network, only a few lightweight masking parameters are introduced, which have minimal effect on both training and inference speed. Yet, these in turn substantially enhance policy optimization.

Additional Experiments We conduct additional experiments to visualize the action distribution during training, both with and without masking. The results indicate that masking promotes a more focused and potentially more effective learning process. Furthermore, we increase the PPO step size from 10^6 to 4×10^6, demonstrating that our method achieves both high efficiency and improved performance. Additionally, we investigate whether network capacity plays a critical role in addressing high-dimensional action problems. However, we find that merely increasing network capacity does not necessarily simplify the learning process. Detailed results can be found in Appendix C.

Table 2: Results on the PPO and SAC for three Mujoco tasks: Ant, HalfCheetah, and Hopper. Action selection is utilized at the beginning stage of RL training. The final reward is the performance evaluation for the agent after training. The best-performing results between KS and All are highlighted in bold.

Env	RL Algo.	p	Selection	Ant			
				TPR (↑)	FDR (↓)	FPR (↓)	Reward (↑)
Ant	PPO	0	True	1.00	0.0	0.00	567.77
		20	KS	1.00	**0.01**	**0.01**	**507.90**
			All	1.00	0.71	1.00	202.65
		50	KS	1.00	**0.00**	**0.00**	**572.39**
			All	1.00	0.86	1.00	151.66
	SAC	0	True	1.00	0.00	0.00	817.95
		20	KS	1.00	**0.01**	**0.01**	**937.74**
			All	1.00	0.71	1.00	12.61
		50	KS	1.00	**0.00**	**0.00**	**731.73**
			All	1.00	0.86	1.00	−208.04
HalfCheetah	PPO	0	True	1.00	0.0	0.00	2130.55
		20	KS	1.00	**0.01**	**0.01**	**2237.08**
			All	1.00	0.77	1.00	1356.46
		50	KS	1.00	**0.00**	**0.00**	**1932.27**
			All	1.00	0.89	1.00	619.67
	SAC	0	True	1.00	0.00	0.00	6640.05
		20	KS	1.00	**0.00**	**0.00**	**6607.55**
			All	1.00	0.77	1.00	5631.20
		50	KS	1.00	**0.00**	**0.00**	**6873.95**
			All	1.00	0.89	1.00	4748.24
Hopper	PPO	0	True	1.00	0.0	0.00	1736.65
		20	KS	1.00	**0.00**	**0.00**	**1540.83**
			All	1.00	0.87	1.00	1205.12
		50	KS	1.00	**0.00**	**0.00**	**1710.82**
			All	1.00	0.94	1.00	703.08
	SAC	0	True	1.00	0.00	0.00	2511.00
		20	KS	1.00	**0.00**	**0.00**	**2165.81**
			All	1.00	0.87	1.00	398.75
		50	KS	1.00	**0.00**	**0.00**	**2424.09**
			All	1.00	0.94	1.00	137.67

6 Conclusion, Limitation, and Future Work

In this work, we address the high-dimensional action selection problem in online RL. We formally define the objective of action selection by identifying a minimal sufficient action set. We innovate by integrating a knockoff-sampling variable selection into broadly applicable deep RL algorithms. Empirical evaluations in synthetic robotics and treatment allocation environments demonstrate the enhanced efficacy of our approach. Yet, a notable constraint of our method is its singular application during the training phase, coupled with the potential risk of overlooking essential actions with weak signals. Inadequate action selection could degrade the agent's performance. Intriguing future research includes extending our methodology to incorporate multiple and adaptive selection stages. This adaptation could counterbalance initial omissions in action selection. Additionally, formulating an effective termination criterion for this process represents another compelling research direction.

References

Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning, 71:89–129, 2008.

Muhammed Fatih Balın, Abubakar Abid, and James Zou. Concrete autoencoders: Differentiable feature selection and reconstruction. In International conference on machine learning, pp. 444–453. PMLR, 2019.

Rina Foygel Barber and Emmanuel J Candès. Controlling the false discovery rate via knockoffs. 2015.

Henry Berbee. Convergence rates in the strong law for bounded mixing sequences. Probability theory and related fields, 74(2):255–270, 1987.

Richard C Bradley. Basic properties of strong mixing conditions. a survey and some open questions. 2005.

Emmanuel Candes, Yingying Fan, Lucas Janson, and Jinchi Lv. Panning for gold:'model-x'knockoffs for high dimensional controlled variable selection. Journal of the Royal Statistical Society Series B: Statistical Methodology, 80(3):551–577, 2018.

Jianbo Chen, Mitchell Stern, Martin J Wainwright, and Michael I Jordan. Kernel feature selection via conditional covariance minimization. Advances in Neural Information Processing Systems, 30, 2017.

Alberto Silvio Chiappa, Alessandro Marin Vargas, Ann Huang, and Alexander Mathis. Latent exploration for reinforcement learning. Advances in Neural Information Processing Systems, 36, 2024.

Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. Sbeed: Convergent reinforcement learning with nonlinear function approximation. In International conference on machine learning, pp. 1125–1134. PMLR, 2018.

Gregory Farquhar, Laura Gustafson, Zeming Lin, Shimon Whiteson, Nicolas Usunier, and Gabriel Synnaeve. Growing action spaces. In International Conference on Machine Learning, pp. 3040–3051. PMLR, 2020.

Samuel J Gershman, Bijan Pesaran, and Nathaniel D Daw. Human reinforcement learning subdivides structured action spaces by learning effector-specific values. Journal of Neuroscience, 29(43):13524–13531, 2009.

Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. arXiv preprint arXiv:1202.3725, 2012.

Zhaohan Daniel Guo and Emma Brunskill. Sample efficient feature selection for factored mdps. arXiv preprint arXiv:1703.03454, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning, pp. 1861–1870. PMLR, 2018.

Botao Hao, Yaqi Duan, Tor Lattimore, Csaba Szepesvári, and Mengdi Wang. Sparse feature selection makes batch reinforcement learning more sample efficient. In International Conference on Machine Learning, pp. 4063–4073. PMLR, 2021.

Vishal Jain, William Fedus, Hugo Larochelle, Doina Precup, and Marc G Bellemare. Algorithmic improvements for deep reinforcement learning applied to interactive fiction. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 4328–4336, 2020.

Yifeng Jiang, Tom Van Wouwe, Friedl De Groote, and C Karen Liu. Synthesis of biologically realistic human motion using joint torque actuation. ACM Transactions On Graphics (TOG), 38(4):1–12, 2019.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. Scientific data, 3(1):1–9, 2016.

James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Knockoffgan: Generating knockoffs for feature selection using generative adversarial networks. In International conference on learning representations, 2018.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. arXiv preprint arXiv:1903.00374, 2019.

Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. Action space shaping in deep reinforcement learning. In 2020 IEEE conference on games (CoG), pp. 479–486. IEEE, 2020.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11):1238–1274, 2013.

Petter N Kolm and Gordon Ritter. Modern perspectives on reinforcement learning in finance. Modern Perspectives on Reinforcement Learning in Finance (September 6, 2019). The Journal of Machine Learning in Finance, 1(1), 2020.

Mark Kroon and Shimon Whiteson. Automatic feature selection for model-based reinforcement learning in factored mdps. In 2009 International Conference on Machine Learning and Applications, pp. 324–330. IEEE, 2009.

Changhee Lee, Fergus Imrie, and Mihaela van der Schaar. Self-supervision enhanced feature selection with correlated gates. In International Conference on Learning Representations, 2021.

Kyowoon Lee, Sol-A Kim, Jaesik Choi, and Seong-Whan Lee. Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling. In International conference on machine learning, pp. 2937–2946. PMLR, 2018.

Faming Liang, Qizhai Li, and Lei Zhou. Bayesian neural networks for selection of drug sensitive genes. Journal of the American Statistical Association, 113(523):955–972, 2018.

Jingyuan Liu, Ao Sun, and Yuan Ke. A generalized knockoff procedure for fdr control in structural change detection. Journal of Econometrics, 2022.

Ying Liu and Cheng Zheng. Auto-encoding knockoff generator for fdr controlled variable selection. arXiv preprint arXiv:1809.10765, 2018.

Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. Deep reinforcement learning for dynamic treatment regimes on medical registry data. In 2017 IEEE international conference on healthcare informatics (ICHI), pp. 380–385. IEEE, 2017.

Shuzhen Luo, Ghaith Androwis, Sergei Adamovich, Erick Nunez, Hao Su, and Xianlian Zhou. Robust walking control of a lower limb rehabilitation exoskeleton coupled with a musculoskeletal model via deep reinforcement learning. Journal of neuroengineering and rehabilitation, 20(1):1–19, 2023.

Shiyang Ma, James Dalgleish, Justin Lee, Chen Wang, Linxi Liu, Richard Gill, Joseph D Buxbaum, Wendy K Chung, Hugues Aschard, Edwin K Silverman, et al. Powerful gene-based testing by integrating long-range chromatin interactions and knockoff genotypes. Proceedings of the National Academy of Sciences, 118(47): e2105191118, 2021.

Tao Ma, Hengrui Cai, Zhengling Qi, Chengchun Shi, and Eric B Laber. Sequential knockoffs for variable selection in reinforcement learning. arXiv preprint arXiv:2303.14281, 2023.

Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In International conference on machine learning, pp. 6961–6971. PMLR, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

Brahma S Pavse and Josiah P Hanna. Scaling marginalized importance sampling to high-dimensional state-spaces via state abstraction. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pp. 9417–9425, 2023.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In Conference on Robot Learning, pp. 1634–1644. PMLR, 2022.

Yaniv Romano, Matteo Sesia, and Emmanuel Candès. Deep knockoffs. Journal of the American Statistical Association, 115(532):1861–1872, 2020.

Sherry Ruan, Gheorghe Comanici, Prakash Panangaden, and Doina Precup. Representation discovery for mdps using bisimulation metrics. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 29, 2015.

Tomonori Sadamoto, Aranya Chakrabortty, and Jun-ichi Imura. Fast online reinforcement learning control using state-space dimensionality reduction. IEEE Transactions on Control of Network Systems, 8(1): 342–353, 2020.

Andrey Sakryukin, Chedy Raïssi, and Mohan Kankanhalli. Inferring dqn structure for high-dimensional continuous control. In International Conference on Machine Learning, pp. 8408–8416. PMLR, 2020.

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

Pierre Schumacher, Daniel Haeufle, Dieter Büchler, Syn Schmitt, and Georg Martius. DEP-RL: Embodied exploration for reinforcement learning in overactuated and musculoskeletal systems. In The Eleventh International Conference on Learning Representations, 2023. URL `https://openreview.net/forum?id=C-xa_D3oTj6`.

Matteo Sesia, Chiara Sabatti, and Emmanuel J Candès. Gene hunting with knockoffs for hidden markov models. arXiv preprint arXiv:1706.04677, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489, 2016.

Xueqing Sun, Tao Mao, Laura Ray, Dongqing Shi, and Jerald Kralik. Hierarchical state-abstracted and socially augmented q-learning for reducing complexity in agent-based learning. Journal of Control Theory and Applications, 9:440–450, 2011.

Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. arXiv preprint arXiv:1512.01124, 2015.

Gabriel Synnaeve, Jonas Gehring, Zeming Lin, Daniel Haziza, Nicolas Usunier, Danielle Rothermel, Vegard Mella, Da Ju, Nicolas Carion, Laura Gustafson, et al. Growing up together: Structured exploration for large action spaces. 2019.

Huachun Tan, Hailong Zhang, Jiankun Peng, Zhuxi Jiang, and Yuankai Wu. Energy management of hybrid electric bus based on deep reinforcement learning in continuous state and action space. Energy Conversion and Management, 195:548–560, 2019.

Robert Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B: Statistical Methodology, 58(1):267–288, 1996.

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, pp. 5026–5033. IEEE, 2012.

Baicen Xiao, Qifan Lu, Bhaskar Ramasubramanian, Andrew Clark, Linda Bushnell, and Radha Poovendran. Fresh: Interactive reward shaping in high-dimensional state spaces using human feedback. arXiv preprint arXiv:2001.06781, 2020.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. ACM Computing Surveys (CSUR), 55(1):1–36, 2021.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. Advances in neural information processing systems, 31, 2018.

Dianyu Zhong, Yiqin Yang, and Qianchuan Zhao. No prior mask: Eliminate redundant action for deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 17078–17086, 2024.

A Implementation Details

We adopt the implementation from Open AI Spinning up Framework (Achiam, 2018). Tables A.1 and A.2 show the hyperparameters for the RL algorithms we used in our experiments. We set the FDR rate $\alpha = 0.1$ and voting ratio $r = 0.5$ for our knockoff method in all settings. All the experiments are conducted in the server with 4× NVIDIA RTX A6000 GPU.

Table A.1: PPO Hyperparameters

Parameter	Mujoco	EHR
optimizer	Adam	Adam
learning rate π	$3.0 \cdot 10^{-4}$	$3.0 \cdot 10^{-3}$
learning rate V	$1.0 \cdot 10^{-3}$	$1.0 \cdot 10^{-3}$
learning rate schedule	constant	constant
discount (γ)	0.99	0.99
number of hidden layers (all networks)	2	2
number of hidden units per layer	$[64, 32]$	$[64, 32]$
number of samples per minibatch	256	100
number of steps per rollout	1000	100
non-linearity	ReLU	ReLU
gSDE		
initial log σ	0	0
Full std matrix	Yes	Yes
Lattice		
initial log σ	0	0
Full std matrix	Yes	Yes
Std clip	(0.001,1)	(0.001,1)

Table A.2: SAC Hyperparameters

Parameter	Mujoco
optimizer	Adam
learning rate π	$3.0 \cdot 10^{-4}$
learning rate Q	$3.0 \cdot 10^{-4}$
learning rate schedule	constant
discount (γ)	0.9
replay buffer size	$1 \cdot 10^{6}$
number of hidden layers (all networks)	2
number of hidden units per layer	$[256, 256]$
number of samples per minibatch	256
non-linearity	ReLU
entropy coefficient (α)	0.2
warm-up steps	$1.0 \cdot 10^{4}$

B More Experimental Results and Analyses

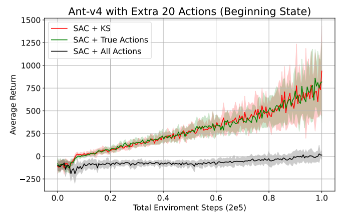
We list the dimension of action and state in terms of environments we used in Table B.1.

B.1 MuJoCo

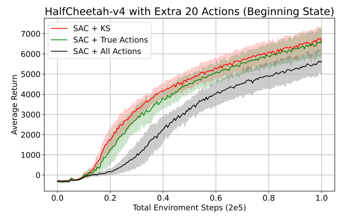
The results for the initial stage are shown in Fig. B.1 and Table 2. For different environments, action selection difficulty varies and Hopper is the easiest one where the proposed KS method can correctly select the minimal

Table B.1: Summary of Environments

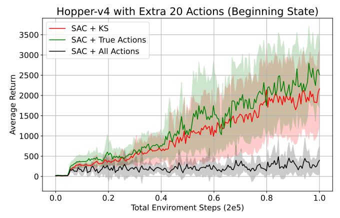
Env	Dimension of Action	Dimension of State
Ant	8	27
HalfCheetah	6	17
Hopper	3	11
EHR	20	46



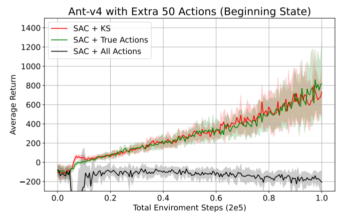
(a) SAC Ant $p = 20$



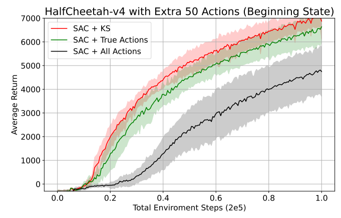
(b) SAC HalfCheetah $p = 20$



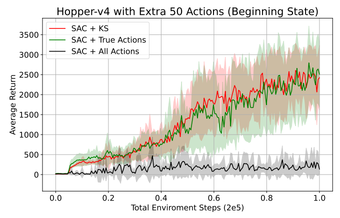
(c) SAC Hopper $p = 20$



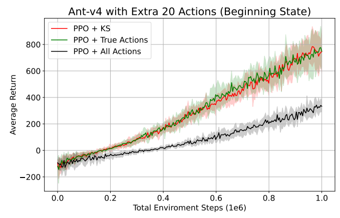
(d) SAC Ant $p = 50$



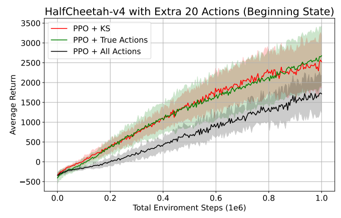
(e) SAC HalfCheetah $p = 50$



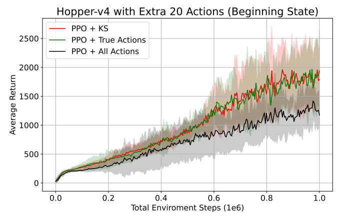
(f) SAC Hopper $p = 50$



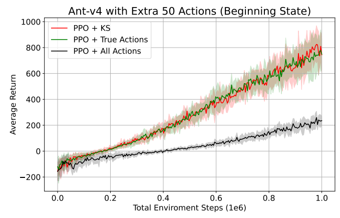
(g) PPO Ant $p = 20$



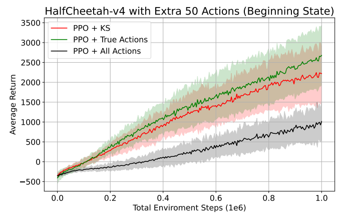
(h) PPO HalfCheetah $p = 20$



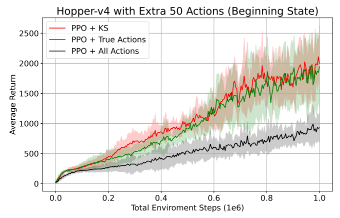
(i) PPO Hopper $p = 20$



(j) PPO Ant $p = 50$



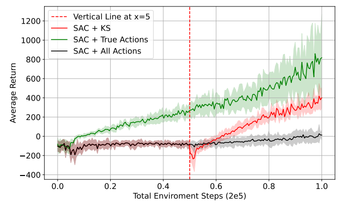
(k) PPO HalfCheetah $p = 50$



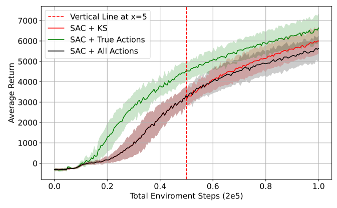
(l) PPO Hopper $p = 50$

Figure B.1: Results of SAC and PPO when using different variable selection approaches during the initial stage.

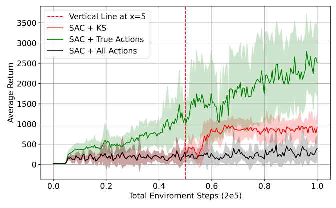
sufficient action set. Also, the patterns of the results for PPO and SAC are similar. One observation is that KS can select all the sufficient actions with all TPRs equal to 1. It is shown that KS is efficient in selecting only the minimal sufficient action set in almost all scenarios, which also empirically validates our theory of FDR control under the proposed method.



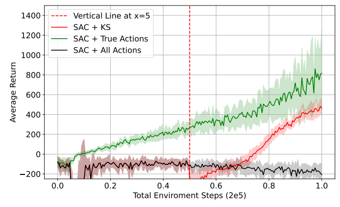
(a) SAC Ant $p = 20$



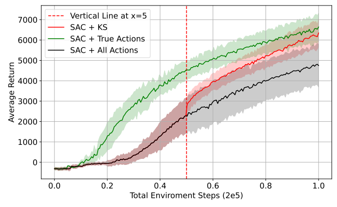
(b) SAC HalfCheetah $p = 20$



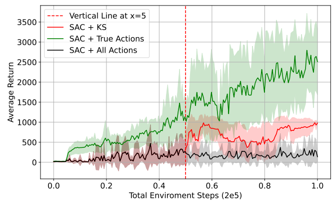
(c) SAC Hopper $p = 20$



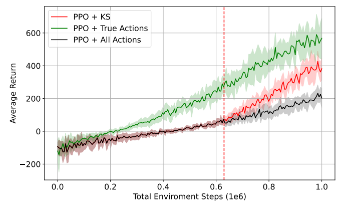
(d) SAC Ant $p = 50$



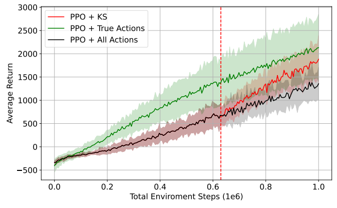
(e) SAC HalfCheetah $p = 50$



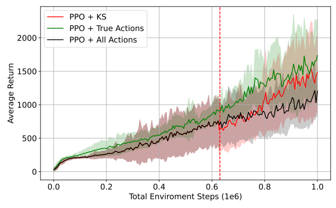
(f) SAC Hopper $p = 50$



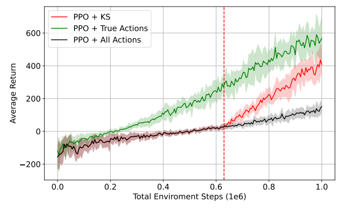
(g) PPO Ant $p = 20$



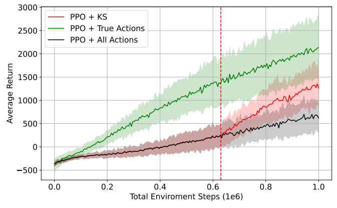
(h) PPO HalfCheetah $p = 20$



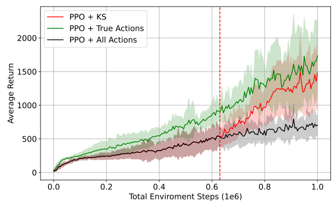
(i) PPO Hopper $p = 20$



(j) PPO Ant $p = 50$



(k) PPO HalfCheetah $p = 50$



(l) PPO Hopper $p = 50$

Figure B.2: Learning curves in the MujoCo environments with our method during the middle stage where the red line indicates the time we utilize KS. After identifying a less redundant action set, the policy can be more efficient and achieve higher rewards than continuing training on all actions.

B.2 Treatment Allocation for Sepsis Patients

We collect 250000 data points from the MIMIC-III Clinical Database. Then we utilize a long short-term memory (LSTM) to model the state transition. The observed state information encompasses a broad spectrum of clinical and laboratory variables for assessing patient health and outcomes in a medical setting. It includes demographic information (gender, age), physiological metrics (weight, vital signs such as heart rate, blood pressure, respiratory rate, oxygen saturation, temperature), and neurological status (Glasgow Coma Scale). Additionally, it captures details about the patient's readmission status, mechanical ventilation use, and severity

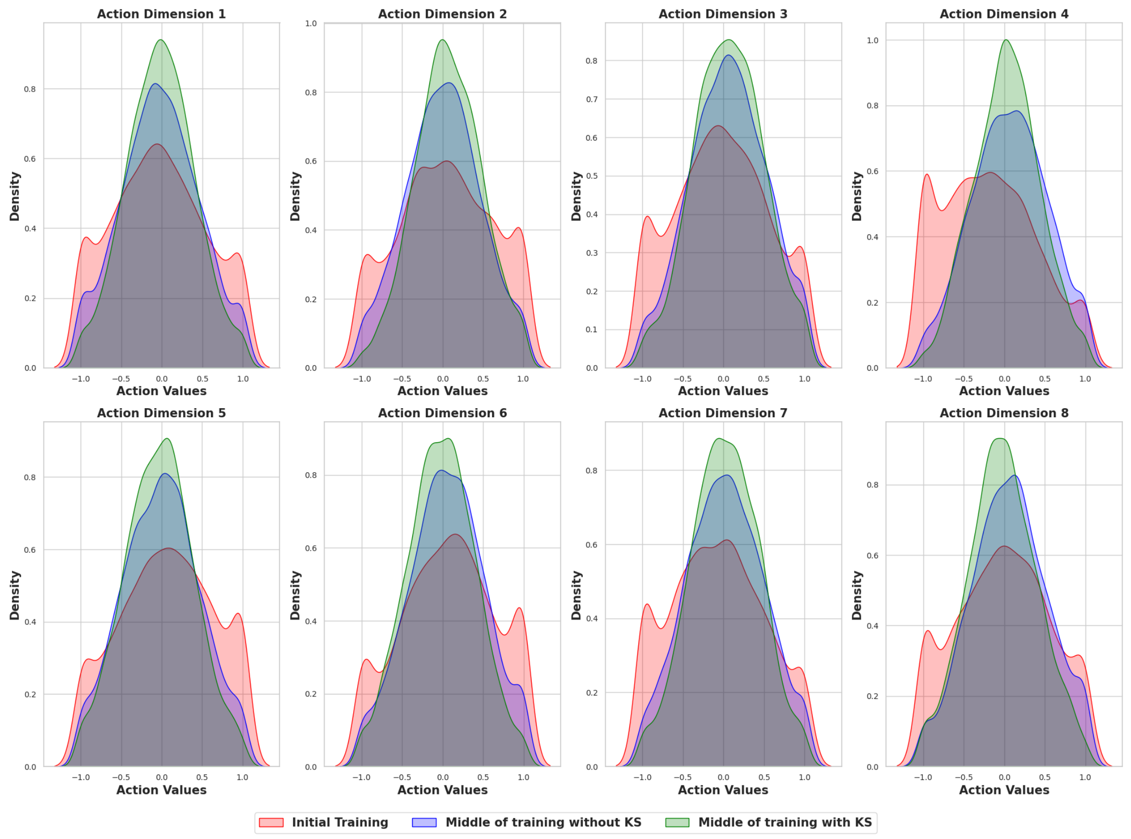


Figure C.1: The distributions of actions in 3 stages: initial training, middle of training without KS, and middle of training with KS. It can be seen that with KS the actions have slightly less variance than other methods, which could be a positive indicator of a more focused and potentially more effective learning process.

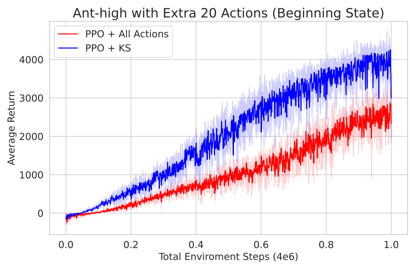
scores such as SOFA and SIRS. Comprehensive lab tests cover a variety of blood chemistry components, including electrolytes, liver enzymes, and arterial blood gases.

The action includes treatments such as the median and maximum doses of vasopressors administered to manage blood pressure and perfusion, alongside vasopressors and intravenous fluids. The reward is calculated based on whether the patient's SOFA Score has improved. Termination of an episode is achieved based on the patient's mortality rate reaching the minimum (SOFA Score being 0) or the patient's mortality rate reaching the maximum (SOFA score being 24).

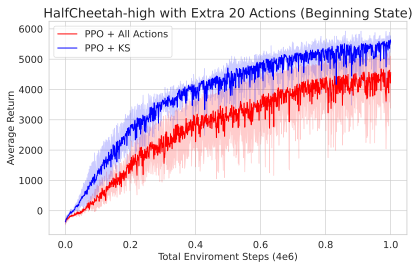
C Supporting Analyses

We conducted additional experiments regarding the distribution of actions that were sampled over the training period. The new results are summarized in Figure C.1. Based on the results together with existing figures in the main text, we can conclude that there exist changes regarding the distribution of actions that are sampled during different periods of training which could be a positive indicator of a more focused and potentially more effective learning process.

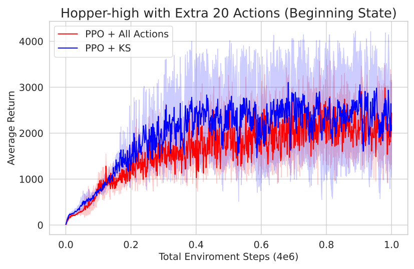
Since we only use steps for PPO training which might be able to converge in the end, we further add the steps to 4e6. The new results are summarized in Figure C.2. Based on the results together with existing figures in the main text, we can conclude that the benefit of the proposed framework is both sample efficiency and performance.



(a) Learning Curves

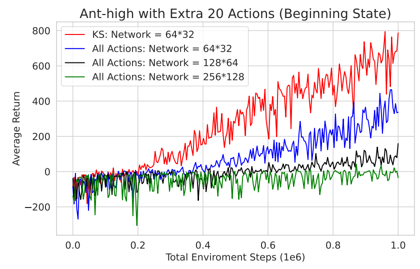


(b) Learning Curves

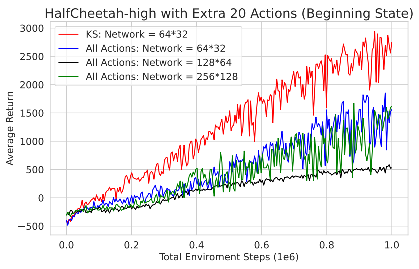


(c) Learning Curves

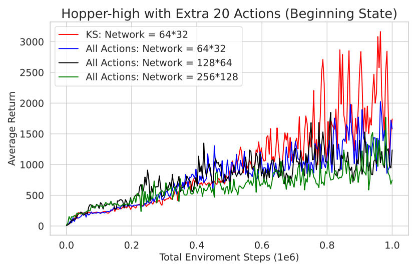
Figure C.2: Learning curves in the MujoCo environments with $4e6$ steps.



(a) Learning Curves



(b) Learning Curves



(c) Learning Curves

Figure C.3: Learning curves in the MujoCo environments with different network sizes.

We also conducted additional experiments by increasing the network size. The new results under MujoCo with different network sizes are summarized in Figure C.3, where we can conclude that the network capacity has a certain influence on the performance of All Action. However, simply increasing the network capacity unnecessarily makes learning easier. In addition, the proposed method consistently performs better than All Action no matter how we increase the network size, which indicates the performance difference results from the redundancy of action space. In addition, we admit there are other factors that affect how the agent learns with a larger action dimension, such as regularization techniques, albeit with limited influence compared with the redundancy of action space.

D Extend to Correlated Actions

Now assume the policy network is parameterized by a multivariate Gaussian with covariance matrix as:

$$\mathbf{a} \sim \mathcal{N}\left(\mu\left(\mathbf{s}\right), \Sigma\left(\mathbf{s}\right)\right),$$

where μ and Σ are parameterized functions to output the mean and covariance matrix.

Since the actions are correlated, we couldn't mask the individual $\log \pi_\theta\left(a_i \mid \mathbf{s}\right)$. To solve this problem, we can transform the non-selected actions to be conditional independent first.

Given a selected action set $\widehat{G}$, we define a selection vector $\mathbf{m} = (m_1, \cdots, m_p) \in \{0,1\}^p$, where $m_i = 1$ if $i \in \widehat{G}$ and 0 otherwise. Then we mask the covariance matrix as follows:

$$\Sigma^m(\mathbf{s})_{ij} \begin{cases} \Sigma(\mathbf{s})_{ij} & \text{if } m_i = 1, \text{and } m_j = 1, i \neq j \\ 0 & \text{if } m_i = 0, \text{or } m_j = 0, i \neq j. \end{cases}$$

The masking only changes non-selected actions to be independent and removes its influence on selected actions which still keeps the covariance structure of selected actions. Then we can easily mask the log density of non-selected actions.

E Machine Learning Algorithm for Calculating Importance Scores

In Knockoff, the computation of the importance scores is very flexible since many machine learning algorithms can be used. The only requirement for the ML method is to satisfy a fairness constraint, so that that swapping $\tilde{X}_j$ with X_j would have the only effect of swapping Z_j with $\tilde{Z}_j$, which is usually true in standard tabular machine learning algorithms, like regression, decision tree. One example is that when we fit a linear regression model, then the coefficients of the variables can be seen as importance scores.

F Technical Proofs

F.1 Preliminary Results

Before we prove Theorem 1, we first provide a preliminary lemma of our procedure that can enable the flip-sign property of W-statistics. This property can be used to prove Theorem 1 when observations are independent. Now we focus on one data split $\mathcal{D}_k$ and assume the data are independent.

Lemma F.1. *$\mathbf{A}_k$ and $\mathbf{S}_k$ are a action and a state matrix. For any subset $\Omega \subset \{1, \ldots, p\}$, and $\tilde{\mathbf{A}}_k$ obatined by resampling, we have*

$$\left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right]_{\mathrm{swap}(\Omega)}, \mathbf{S}_k\right) \overset{d}{=} \left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right], \mathbf{S}_k\right),$$

where $\mathrm{swap}(\Omega)$ *represents swapping the j-th entry of $\mathbf{A}_k$ and $\tilde{\mathbf{A}}_k$ for all $j \in \Omega$.*

The proof of Lemma F.1 is based on the property of constructed variables where $\mathbf{A}_k$ and $\tilde{\mathbf{A}}$ have the same marginal distribution and the whole joint distribution is symmetrical in terms of $\mathbf{A}_k$ and $\tilde{\mathbf{A}}$.

In the following, we show the exchangeability holds jointly on actions, states, and rewards when swapping null variables.

Lemma F.2. *Let $\mathcal{H}_0 \subseteq \{1, \ldots, p\}$ be the indices of the null variables, for any subset $\Omega \subset \mathcal{H}_0$*

$$\left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right]_{\mathrm{swap}(\Omega)}, \mathbf{S}_k, \mathbf{Y}_k\right) \overset{d}{=} \left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right], \mathbf{S}_k, \mathbf{Y}_k\right),$$

where $\mathbf{Y}_k$ is a response including the next state and reward.

Proof: Based on the exchangeability proved in Lemma F.1, we can directly utilize the proof of Lemma 3.2 in Candès et al. (2018). We just need to extend the derivation by conditional on $\mathbf{S}_k$ and show equivalence by swapping action variables in Ω one by one. We omit further details of the proof.

Lemma F.3.

$$W_i\left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right]_{\mathrm{swap}(\Omega)}, \mathbf{S}_k, \mathbf{Y}_k\right) = W_i\left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right], \mathbf{S}_k, \mathbf{Y}_k\right) \cdot \begin{cases} -1, & \textit{if } i \in \Omega \\ +1, & \textit{otherwise} \end{cases}.$$

Proof: We require the method constructing W to satisfy a fairness requirement so that swapping two variables would have the only effect of swapping corresponding feature importance scores. The fairness constraint is satisfied with many general machine learning algorithms, like LASSO and random forest. Once the fairness constraint is satisfied, W will be anti-symmetric and the equal above automatically holds.

Lemma F.4. *Assume the flip-coin property in Lemma F.3 is satisfied, on data $\mathcal{D}_k$, the selection $\widehat{G}_k$ obtained from applying knockoff method in Algorithm 2 controls modified FDR (mFDR), e.g.*

$$mFDR\left(\widehat{G}_k\right) \leq \alpha.$$

Proof: For statistics W calculated, we denote $W_{\mathrm{swap}(\Omega)}$ to be the W-statistics computed after the swap w.r.t. $\Omega \subset \{1, \ldots, p\}$. Now consider a sign vector $\epsilon \in \{\pm 1\}^p$ independent of $\mathbf{W} = [W_1, \ldots, W_p]^\top$, where $\epsilon_i = 1$ for all non-null state variables and $\mathbb{P}(\epsilon_i = 1) = 1/2$ are independent for all null state variables. Then for such ϵ, denote $\Omega := \{i : \epsilon_i = -1\}$, which is a subset of $\mathcal{H}_0$ by the assumption (and recall that $\mathcal{H}_0$ is the collection of all null variables). By Lemma F.3 we know

$$(W_1 \cdot \epsilon_1, \ldots, W_p \cdot \epsilon_p) = W_{\text{swap}(\Omega)}.$$

For convenience, we also use h to denote a measurable mapping function from a data set to its W-statistics, i.e., on $[\mathbf{s}_k, \tilde{\mathbf{s}}_k, \mathbf{a}_k, \mathbf{y}_k]$,

$$\mathbf{W} = h\left(\mathbf{A}_k, \tilde{\mathbf{A}}_k, \mathbf{S}_k, \mathbf{Y}_k\right).$$

Then we can get:

$$\begin{aligned}\mathbf{W}_{\text{swap}(\Omega)} &= h\left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right]_{\text{swap}(\Omega)}, \mathbf{S}_k, \mathbf{Y}_k\right) \\ &\overset{d}{=} h\left(\left[\mathbf{A}_k, \tilde{\mathbf{A}}_k\right], \mathbf{S}_k, \mathbf{Y}_k\right) = \mathbf{W},\end{aligned}$$

where the second equality (in distribution) is due to Lemma F.2 and h is measurable. The rest of the proof will be the same as that for Theorems 1 and 2 in Barber & Candès (2015).

F.2 Proof of Theorem 4.3

Using Lemma F.4, we can show that if the data points in $\mathcal{D}_k$ are independent, then $mFDR$ can be controlled. Now we want to weaken the independence assumption to stationarity and exponential β-mixing assumption in 4.2. Based on Lemma F.4, the following proof is essentially the same as Theorem 1 in Ma et al. (2023). We will omit those steps for brevity.