# Budget-Aware Feature Selection for Reinforcement Learning

**Ofek Glick** [1], **Argaman Mordoch** [3], **Guy Azran** [2], **Sarah Keren** [2]
ofek.glick@campus.technion.ac.il
Mordocha@post.bgu.ac.il
guy.azran@campus.technion.ac.il
sarahk@technion.ac.il

[1]**Data and Decisions Faculty, Technion - Israeli Institute of Technology, Israel**
[2]**The Taub Faculty of Computer Science, Technion - Israeli Institute of Technology, Israel**
[3]**Department of Software and Information Engineering, Ben Gurion University in Be'er Sheva, Israel**

## Abstract

Real-world reinforcement learning (RL) agents often operate in high-dimensional environments where features, obtained from sensors or measurements, inform decision-making. However, utilizing all available features can result in significant operational costs. This creates a critical need for budget-sensitive feature selection methods that balance task performance with resource constraints. In this work, we formalize the problem of budget-aware feature selection in RL as a Constrained Markov Decision Process (CMDP) and propose an approach to allow an agent to dynamically toggling features during interaction with the environment. To address the dynamic feature selection challenge, we leverage Safe Reinforcement Learning (Safe RL) methodologies, enabling agents to learn policies that respect strict budgetary constraints during both learning and deployment. By incorporating feature selection into the agent's policy, the agent can learn to choose features independently, while balancing their costs and their relevance or redundancy for the task.

## 1 Introduction

In real-world reinforcement learning (RL) applications, agents often operate in high-dimensional state spaces where multiple input features inform decision-making. Consider a service robot tasked with delivering an apple to a person in a room. The robot has access to various sensors: a high-resolution color camera, a basic grayscale camera, radar sensors, and proximity detectors. While all these sensors provide potentially useful information, activating every sensor, particularly power-intensive ones, significantly increases the robot's energy consumption. Moreover, using all of the sensors at the same time could introduce redundancy which would increase computational costs but not necessarily improve performance. This raises a crucial question: which subset of sensors provides the most task-relevant information at the right time while staying within operational resource constraints?

The above scenario illustrates a fundamental challenge in practical RL deployments: feature selection under budget constraints. While modern RL algorithms can effectively leverage rich feature sets to learn optimal policies, each feature in real-world applications typically requires dedicated sensors or measurement systems. These hardware components introduce not just initial setup costs but also ongoing operational expenses in terms of power consumption, maintenance, and computa-

tional resources. Therefore, there exists a critical need for methods that can identify and utilize only the most essential features when they are most needed while maintaining acceptable performance.

Safe Reinforcement Learning (Safe RL) provides a promising framework to address this challenge by enabling agents to learn optimal policies while respecting predefined constraints, such as sensor cost budgets. Safe RL is a subfield of reinforcement learning that focuses on ensuring that an agent's actions remain within acceptable safety bounds during both learning and deployment.

In the context of this work, safety constraints are adhering to a strict budget for feature acquisition. By incorporating the Safe RL framework, it becomes possible to develop algorithms that not only seek to maximize reward but also guarantee that the agent operates efficiently within resource limits. This relevance is particularly critical in scenarios where exceeding budgetary or operational constraints could lead to significant consequences, such as hardware damage or operational failures.

## 2 Related Works

Our work sits at the intersection of feature selection and Safe Reinforcement Learning. Feature selection for RL has been studied in the previous. Hachiya & Sugiyama (2010) proposed a feature selection method that uses conditional mutual information, approximated via least-squares estimation and forward selection, to identify state features relevant to future rewards while discarding irrelevant ones. Nguyen et al. (2013) proposed to dynamically identify and retains features for predicting action effects by applying online multinomial logistic regression with group lasso regularization, which zeros out irrelevant features during model updates. Kishima & Kurashige (2012) proposed a two-phase pipeline to identify features which highly correlated with the reward function and classifying them as either important or unimportant features with respect to a task. This method showed improved results and also provided some explainability as to why features were chosen, but correlation isn't always a good predictor for high reward. To this end, we proposed to make the feature selection part of the policy itself, therefor the reward has direct effect on the feature selection via optimization. While those methods proved effective for identifying relevant features, they do not not take sensor costs into account. In contrast to prior work, our method is, to our knowledge, the first to formulate feature selection of the state-space as a Constrained Markov Decision Process (CMDP). Our approach operates as a wrapper of existing environments, making it a plug-n-play approach which allow it to be solvable via existing safe RL techniques.

## 3 Preliminaries

In reinforcement learning setting Sutton & Barto (2018), an Markov Decision Process (MDP) is represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where:

- $\mathcal{S}$: The state space, where each state $s_t \in \mathcal{S}$ at time $t$ is represented as a set of features $\{f_1, f_2, \ldots, f_n\}$ (e.g., the agent's sensory inputs).
- $\mathcal{A}$: The action space, which may include discrete, continuous, or hybrid actions.
- $\mathcal{P}(s_{t+1}|s_t, a_t)$: The state transition probability distribution, defining the probability of transitioning from state $s_t$ to $s_{t+1}$ given action $a_t \in \mathcal{A}$.
- $r(s_t, a_t)$: The reward function, which returns a scalar reward based on the agent's current state $s_t$ and action $a_t$.
- $\gamma \in [0, 1)$: The discount factor, which determines the importance of future rewards.

In reinforcement learning the goal is to find an optimal policy $\pi^*$ that maximizes the expected discounted cumulative reward. Formally, the problem can be defined as:

$$J(\pi) = \max_{\pi} V_\pi(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid S_0 = s\right]$$

A common approach to solving this problem is through model-free RL algorithms such as PPO Schulman et al. (2017b), TRPO Schulman et al. (2017a) or SAC Haarnoja et al. (2018), where we directly learn policies through interaction with the environment.

The typical setup is that an agent interacts with an environment by observing a state $s_t$, selecting an action $a_t$ from its action space $\mathcal{A}$, receiving a reward $r(s_t, a_t)$, and transitioning to the next state $s_{t+1}$. The action space can be discrete, continuous, or both. In discrete spaces, the agent chooses from a set of predefined actions, while in continuous spaces, it selects from a range of real values. In cases where the action space is hybrid, optimization becomes more complex as the agent must balance discrete and continuous decisions simultaneously. This is typically done by using multi-head actor architectures, such as the MERL (Multi-Head Reinforcement Learning) framework Fleet-Berliac & Preux (2020) and Hybrid Proximal Policy Optimization (H-PPO) Fan et al. (2019). The multi-head architecture features a shared parameter network with multiple output heads. Each head independently processes the final embedding layer to predict different task components, allowing the architecture to adaptively manage diverse tasks within a unified framework. This structure supports both continuous and discrete control settings by separating the responsibilities of each head while maintaining efficient shared learning across tasks.

### 3.1 Safe Reinforcement Learning (Safe RL)

Safe RL focuses on learning policies that maximize expected return while satisfying predefined constraints. These constraints can represent safety requirements, resource limitations, or operational budgets. Safe RL methods rely on Constrained Markov Decision Processes (CMDPs) Altman (1999) to formalize the problem.

A CMDP expands upon the MDP and is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, C, d)$, where $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the cost function, and $d$ is the allowable budget for the cumulative cost. The objective is to find a policy $\pi$ that maximizes the expected cumulative reward while ensuring that the expected cumulative cost remains within the budget:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma_r^t r(s_t, a_t) \mid \pi\right] \quad \text{subject to} \quad \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma_c^t C(s_t, a_t) \mid \pi\right] \leq d.$$

Constrained versions of model-free algorithms such as TRPO, PPO, and SAC have emerged. For example, CPO Achiam et al. (2017) expands the TRPO algorithm by explicitly adding constraints to the trust region update. Lagrangian relaxation methods have also been applied to existing algorithms Achiam & Amodei (2019). Works on environment augmentation techniques such as Sootla et al. (2022) incorporate constraints into the state space, enabling real-time adaptation to safety conditions. These approaches collectively focus on safe exploration and maintaining optimal performance under constraints.

## 4 Modeling and Solving Budget-Aware Feature Selection

Given an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, *Budget-Aware Feature Selection* (BAFS) in RL involves identifying a subset of relevant features from the high-dimensional state space in an online setting. As part of the agents learning, the agent must also recognize which features are contributing to the learning efficiency. In our setting, we assume the agent can dynamically toggle features "on" and "off" during interaction with the environment. This ability offers greater flexibility, as the agent can adaptively decide which features to use at different time steps. However, it also introduces additional complexity, as the agent must adapt to a new learning paradigm, specifically, when to activate or deactivate features as part of its policy given limited resources.

The objective in this problem setting is optimizing two policies concurrently, the original *environment interaction* policy, which guides the agent towards achieving it's goal, and find a *feature selec-*

*tion* policy which allows the agent to reduce the costs of feature usage while maintaining maximal rewards under the allocated budget.

We address the BAFS problem by extending the Constrained Markov Decision Process (CMDP) framework. This extended framework, termed *BAFS-CMDP*, is defined by the tuple:

$$(\mathcal{S}, \mathcal{A}_{ext}, \mathcal{P}, r, \gamma, C, d)$$

The elements of the BAFS-CMDP are consistent with the standard CMDP, except for the expanded action space $\mathcal{A}_{ext}$, which integrates both environment interaction and feature selection actions:

$$\mathcal{A}_{ext} = \mathcal{A}_{\text{env}} \cup \mathcal{A}_{\text{feat}}$$

- $\mathcal{A}_{\text{env}}$: The set of actions for interacting with the environment.
- $\mathcal{A}_{\text{feat}} : \{u_t \in \{0,1\}^n\}$: A set of feature-selection actions where each feature $f_i$ can be enabled or disabled by adjusting the corresponding value in $u_t$. This allows the agent to dynamically control which features are active at each time-step $t$.

Formally, we assume the agent is equipped with $n$ sensors. Let $f = (f_1, f_2, \ldots, f_n)$ denote a vector of all features where each feature represents as sensor. A feature configuration at time $t$ is represented by a binary vector $u_t \in \{0,1\}^n$, where the $i$-th entry indicates whether feature $i$ is active (1) or inactive (0) at time $t$.

The agent's decision-making is governed by two policies $\pi = (\pi_u, \pi_m)$ which we aim to learn

- $\pi_u$: A *feature selection* policy that chooses which subset of features to acquire (incurring feature usage costs).
- $\pi_m$: An *environment interaction* policy that selects the control action based on the (possibly reduced) observation.

Each feature $f_i$ from the feature vector $f$ is associated with a cost $c(f_i) \geq 0$. At any state and time-step $s_t$, if the agent selects a subset of features $u_t$, the total feature cost is given by:

$$C(s_t, u_t) = c(f)^T u_t.$$

The agent's **feature cost** accumulates over time, and a budget $d$ is imposed on the *expected* cumulative cost. Formally, we want to solve:

$$\max_{\pi=(\pi_u,\pi_m)} \quad \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \, r(s_t, a_t)\Big] \quad \text{subject to} \quad \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \, C(s_t, u_t)\Big] \;\leq\; d.$$

Here, $r(\cdot)$ is the reward function defined by the environment.

To handle the dual tasks of environment interaction and feature selection, we use a multi-head actor architecture for simultaneous training on both tasks. Following the architecture presented in Flet-Berliac & Preux (2020), The network initially shares parameters in the early layers to learn a common representation of the input space, to ensure common feature extraction and generalization. As the network progresses, it splits into separate heads, each dedicated to a specific task one for selecting relevant features and another for optimizing interaction with the environment.

For continuous action parameters, our network employs a Gaussian policy that learns both mean and variance of each action distribution and for the the discrete action parameters of the network models a discrete probability distribution over binary actions. The agent learns to activate or deactivate specific features, ensuring that it can effectively decide when a particular feature is relevant for decision-making.

## 5 Experiments

To evaluate our approach, we measure both overall agent performance as opposed to standard training, without feature selection and without feature cost considerations. Our goal in our experiments

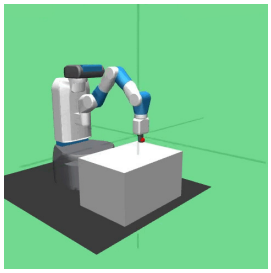


Figure 1: The Fetch Reach environment setup - the robot must reach the end the red dot, which is randomly placed in the agents workspace, with its end effector.

is to examine the resultant feature selection policy and determine whether using BAFS actually contributes to lowering feature cost usage and maintaining strong results.

We evaluate our approach on a continuous control task from Gymnasium Robotics de Lazcano et al. (2024) called Fetch Reach, an image from the environment can be seen in Figure 1. The task in the environment is for a manipulator to move the end effector to a randomly selected position in the robot's workspace. The observation space of the agent is comprised of 16 features: 10 features containing kinematic data of the end effector, including its global position (x, y, z), joint displacements of the gripper fingers, and linear velocities in different directions, 3 features specifying the target end effector position in Cartesian coordinates (x, y, z), 3 features representing the current position of the end effector.

Each feature was assigned a random cost drawn from a uniform distribution to represent the cost of using a certain feature at a given step. Using random costs allows simulating environments where different features have different costs. The reward in this environment is dense and represents the negative Euclidean distance between the achieved goal position and the desired goal. The action space is 3 dimensional, an action represents the Cartesian displacement dx, dy, and dz of the end effector. All RL algorithms, both safe and unsafe were based on the Omnisafe framework Ji et al. (2024), an infrastructural framework which provides a comprehensive and reliable toolkit for safe RL algorithms. All hyper-parameters and network architectures are based on the standard configurations that can be viewed on the Omnisafe home site.

To evaluate the agents ability to learn a feature selection policy, we first evaluate the BAFS method on a standard MDP (i.e. without constraints) with an expanded action space. The agent is trained using PPO algorithm Schulman et al. (2017b) for 1 million steps. We then evaluate the model for 20 evaluation episodes, each episode consists of 250 steps. For each evaluation episode we measure the average activation rate of each feature. The results, summarized in Figure 2, indicate that when feature costs are not considered, the agent appears to select most features for approximately 50% of the time, all the while still maintaining comparable reward to an agent that is required to use the entire observation space.

To further evaluate the feature selection policy learning we test our approach on a CMDP (i.e with constraints taken into account in the policy learning). We train three agents using the CPO algorithm Achiam et al. (2017) with budget constraints set at 80%, 50%, and 20% of the total budget. The total budget is defined as 100% of the sum of all feature costs multiplied by the predefined trajectory steps per epoch: $d_{\text{total}} = \sum_{t=0}^{T} C(s_t, \mathbf{1}_t)$. The results are summarized in Figure 3. Despite adding constraints to the policy, the achieved reward occasionally matches standard PPO (with or without feature selection) but lacks consistency. A higher budget does not guarantee full utilization of all features, even though using the entire observation space can improve performance. Additionally, decreasing the budget causes the agent to focus be more decisive in its feature selection policy as exemplified by the consist choice of features across many different trajectories. In Figure 4, we observe the most frequently activated features on average throughout a trajectory. Once budget constraints are introduced, the reinforcement learning algorithms exhibit a preference for specific

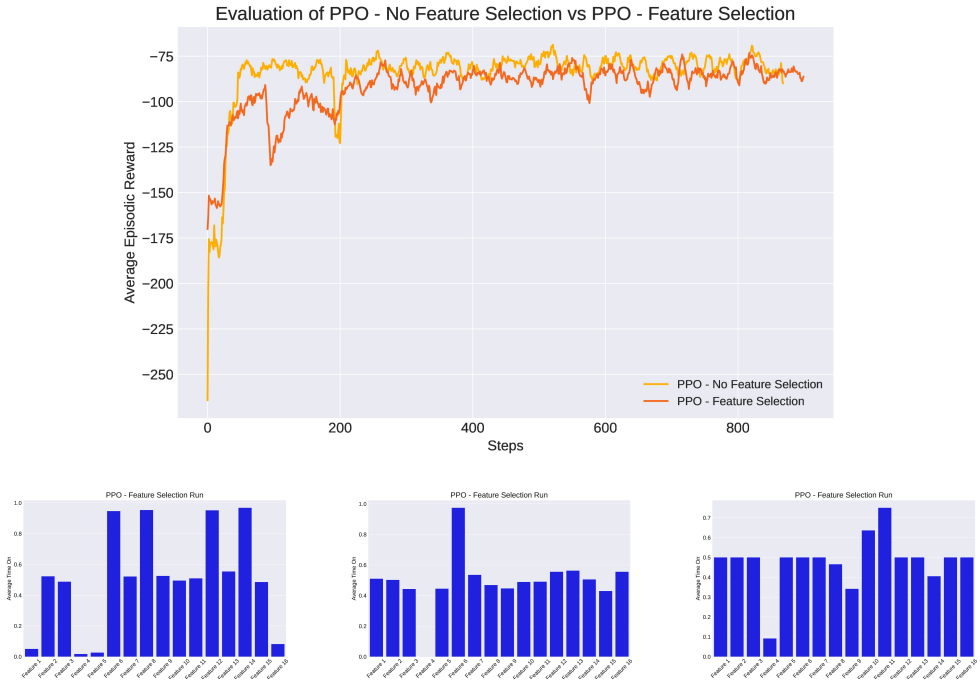



Figure 2: The average activation time of a feature during an evaluation trajectory indicates that most features operate only 50% of the time. Despite this, the agent achieves nearly the same accumulated return as in the full-feature run (without feature selection). This suggests that not all features are necessary at every step, implying either redundancy in feature usage or the agent's ability to selectively activate relevant features.

features throughout most of the run, rather than alternating between features approximately half of the time.

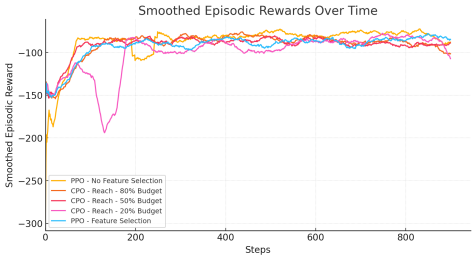
These results show promise at allowing agents to rely on their own policy learning to perform feature selection, and we hope would inspire future research in this under-explored field.

## 6 Limitations

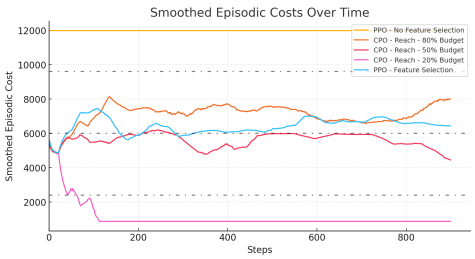
While our proposed method for Budget-Aware Feature Selection (BAFS) provides a structured approach to dynamically optimizing feature usage within reinforcement learning (RL) environments, it has several inherent limitations.

While feature selection reduces operational costs by employing a dedicated feature selection policy, introducing an additional policy alongside the primary task policy increases the overall dimensionality of the problem. The feature selection mechanism expands the action space, as the agent must now learn both when to activate or deactivate features in addition to selecting task-specific actions. This increase in complexity may negatively impact learning efficiency, particularly in environments with a significantly large number of features. In such cases where feature selection is arguably most beneficial, the added complexity may cause the actor network to learn degraded policies. The multi-headed actor architecture is designed to mitigate this challenge by dedicating separate network heads for feature selection and task policy optimization, and therefor act as a divide between the two goals, but its effectiveness is not always guaranteed, especially in high-dimensional environments.

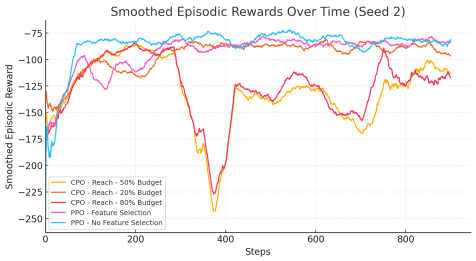
Additionally, it is a well-established challenge that safe reinforcement learning (RL) algorithms often experience constraint violations, as they are designed to adhere to constraints in *expectation*.




(a) Rewards Performance - Seed 1



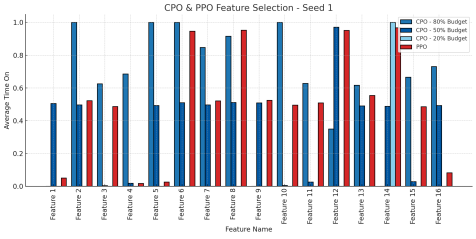

(b) Costs Performance - Seed 1



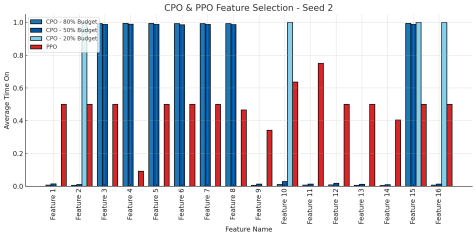

(c) Rewards Performance - Seed 2


(d) Costs Performance - Seed 2

Figure 3: Evaluation costs and rewards for different seeds. The three dashed lines in the costs graph represent the safety budget of 80%, 50% and 20%..




(a) Features Selected On Average - Seed 1




(b) Features Selected On Average - Seed 1

Figure 4: During the evaluation phase, we analyzed the trained feature selection policies by examining the activation ratio of each feature over episodes of 250 steps. Specifically, we measured the proportion of steps in which each feature was activated or de-activated. We compared the performance of PPO and CPO under varying budget constraints. Our findings indicate that while PPO tends to activate nearly all features approximately 50% of the time, as the budget decreases, this pattern shifts. When the budget is restricted to 20%, only a select few features remain consistently active throughout most of the episode.

This approach may be incompatible with real-world scenarios where violations of feature usage costs could be impractical or even lead to catastrophic consequences.

## 7 Conclusions and Future Work

This paper introduced Budget-Aware Feature Selection (BAFS), a novel framework that enables reinforcement learning (RL) agents to dynamically select relevant features while adhering to predefined budget constraints. By formulating BAFS as a CMDP, we leveraged Safe Reinforcement Learning methodologies to ensure efficient and decisive feature selection while remaining competitive in task performance. Empirical evaluation demonstrated that our method effectively reduces feature usage while maintaining performance comparable to standard RL approaches. This could

boast future research in feature selection methodologies in RL, where real-world costs play a more dominant consideration when equipping robotic agents with sensors. Additionally, we observed that feature selection strategies vary significantly across different budget constraints, which inspires future work to research different Safe RL algorithms in the BAFS framework.

Several promising directions remain for future work include incorporating dedicated feature selection signals or auxiliary losses for the feature selection task in addition to the reward signal to further refine selection policies. Extending our approach to higher-dimensional environments with larger state spaces would test its scalability and robustness in more complex domains. Furthermore, since our approach leverages safe reinforcement learning (RL) algorithms to promote a budget-aware feature selection policy, a more comprehensive analysis of safe RL techniques and their impact on the resulting feature selection policies would be valuable.

## References


Joshua Achiam and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019. URL https://api.semanticscholar.org/CorpusID:208283920.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization, 2017. URL https://arxiv.org/abs/1705.10528.

Eitan Altman. Constrained markov decision processes. 1999. URL https://api.semanticscholar.org/CorpusID:14906227.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

Zhou Fan, Rui Su, Weinan Zhang, and Yong Yu. Hybrid actor-critic reinforcement learning in parameterized action space, 2019. URL https://arxiv.org/abs/1903.01344.

Yannis Flet-Berliac and Philippe Preux. Merl: Multi-head reinforcement learning, 2020. URL https://arxiv.org/abs/1909.11939.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Hirotaka Hachiya and Masashi Sugiyama. Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In José Luis Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 474–489, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15880-3.

Jiaming Ji, Jiayi Zhou, Borong Zhang, Juntao Dai, Xuehai Pan, Ruiyang Sun, Weidong Huang, Yiran Geng, Mickel Liu, and Yaodong Yang. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *Journal of Machine Learning Research*, 25(285):1–6, 2024. URL http://jmlr.org/papers/v25/23-0681.html.

Yasutaka Kishima and Kentarou Kurashige. Reduction of state space on reinforcement learning by sensor selection. In *2012 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, pp. 138–143, 2012. DOI: 10.1109/MHS.2012.6492469.

Trung Nguyen, Zhuoru Li, Tomi Silander, and Tze Yun Leong. Online feature selection for model-based reinforcement learning. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 498–506, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/nguyen13.html.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017a. URL https://arxiv.org/abs/1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017b. URL https://arxiv.org/abs/1707.06347.

Aivar Sootla, Alexander I. Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David Mguni, Jun Wang, and Haitham Bou-Ammar. Saute rl: Almost surely safe reinforcement learning using state augmentation, 2022. URL https://arxiv.org/abs/2202.06558.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.