ACTION SHAPLEY: A TRAINING DATA SELECTION METRIC FOR TRAINING WORLD MODELS FOR REINFORCEMENT LEARNING

Rajat Ghosh, Debojyoti Dutta
Nutanix, Inc, San Jose, CA 95110
{rajat.ghosh, debojyoti.dutta}@nutanix.com

ABSTRACT

World models are central to model-based reinforcement learning, enabling agents to predict environment dynamics and reason about future outcomes. In real-world settings, however, training high-fidelity world models is often constrained by limited, noisy, and heterogeneous interaction data, making data selection a critical yet under-studied problem. To address this gap, we introduce **Action Shapley**, a principled data valuation metric for world model training for reinforcement learning. Action Shapley quantifies the marginal contribution of individual action–state trajectories to downstream control performance, enabling systematic selection of high-value training data. To address the exponential cost of Shapley value computation, we propose a randomized algorithm that exploits failure modes of learned world models to identify a cut-off cardinality, significantly reducing computation while preserving ranking fidelity. We evaluate Action Shapley across four real-world, partially observable control domains—including cloud resource management, database tuning, and Kubernetes workload control—using model-based RL agents. Across all domains, Action Shapley–based data selection improves data efficiency by up to 67% and often outperforms training on the full dataset in cumulative reward.

1 INTRODUCTION

World models predict environment dynamics for model-based reinforcement learning (RL), enabling planning, simulation, and policy optimization (Sutton & Barto, 2018). Recent work has focused on improving architectures, latent representations, and training objectives, including hierarchical dynamics (Gumbsch et al., 2023), multimodal latent models (Maytié et al., 2025), masked generative priors (Meo et al., 2024), generalization analysis (Fang et al., 2024), and dynamics decomposition. In contrast, comparatively little work has focused on *which* interaction data should be used to train world models. This data selection problem is a critical problem for real-world RL comparison to simulation-based RL.

Prior work on world models has shown that training predictive environment dynamics under surrogate objectives such as maximum likelihood makes learned models sensitive to redundant, noisy, or weakly informative transitions, as prediction errors can compound during multi-step rollout and planning (Ha & Schmidhuber, 2018; Hafner et al., 2019). Consequently, indiscriminate use of all collected interaction data can degrade model fidelity and increase sample complexity, highlighting data selection as a central design concern in scalable world model training rather than a mere implementation detail. In real-world settings, however, interaction data is expensive, noisy, and often collected under nonstationary and partially observable conditions, which has been repeatedly identified as a key challenge for practical reinforcement learning (Dulac-Arnold et al., 2019; Levine et al., 2020; Agarwal et al., 2020). We address the gap of training data selection for world model learning by introducing **Action Shapley**, a data valuation framework for reinforcement learning that assigns a value to each training action based on its marginal contribution to downstream agent performance.

We formalize the training dataset as a cooperative game in which individual action trajectories contribute to the cumulative reward achieved by an agent trained on the resulting world model, yielding

a principled criterion for ranking and selecting data by contribution rather than availability. Action Shapley is inspired by the Shapley value (Shapley, 1953b;a), which has been widely applied to data valuation in supervised learning (Ghorbani & Zou, 2019a) and model interpretability (Lundberg & Lee, 2017), as well as more recent work on Shapley-based credit assignment and contribution analysis in sequential and multi-agent decision-making settings (Li et al., 2021a; Chen et al., 2022). Extending this formulation to reinforcement learning introduces additional challenges due to non-i.i.d. data, feedback-driven data generation, and instability arising from the deadly triad of function approximation, bootstrapping, and off-policy learning (Sutton & Barto, 2018), which are further exacerbated in modern world-model-based RL systems that rely on learned latent dynamics and imagination-based rollouts (Hafner et al., 2020).

We evaluate Action Shapley on four real-world control problems drawn from cloud system management—virtual machine right-sizing (Derdus et al., 2019), load balancing (Mishra et al., 2020), database tuning (Bangare et al., 2016), and Kubernetes management (Rzadca et al., 2020)—which are representative of practical RL deployment settings (Yazdanov & Fetzer, 2013; Xu et al., 2019; Wang et al., 2021; Khaleq & Ra, 2021) where world models must be learned from limited and heterogeneous interaction data. Such settings are increasingly emphasized in recent work on real-world and systems-oriented reinforcement learning (Dulac-Arnold et al., 2019; Levine et al., 2020), where data sparsity (Kamthe & Deisenroth, 2018), noisy system dynamics (Dulac-Arnold et al., 2019), partial observability (Futoma et al., 2020), and irregular sampling (Yildiz et al., 2021) pose persistent challenges to reliable world model training and evaluation. Our contributions are threefold:

- We introduce **Action Shapley**, a Shapley-value–based data valuation metric for training world models in reinforcement learning, which quantifies the marginal contribution of individual training actions to downstream control performance.
- We propose an efficient randomized algorithm for computing Action Shapley that exploits world model failure modes to identify a cut-off cardinality, substantially reducing the computational cost of Shapley-based data valuation in practice.
- We empirically evaluate Action Shapley on four real-world, data-constrained control domains, demonstrating that principled data selection can reduce required training data by 20–67% while often improving downstream agent performance relative to training on all available data.

2 METHODOLOGY

In Reinforcement Learning (RL), an agent interacts sequentially with an environment to maximize cumulative reward, inducing trajectories $\tau = \langle s_0, a_0, r_1, s_1, a_1, r_2, \ldots \rangle$. Under the Markov Decision Process (MDP) assumption, each interaction is governed by a transition-reward distribution $p(s_t, r_t \mid s_{t-1}, a_{t-1})$. Model-based RL methods approximate this distribution using a training corpus $\mathcal{D}$ of transition tuples $(s_{t-1}, a_{t-1}, s_t, r_t)$. Since reward stochasticity is often low relative to transition dynamics, we model them jointly, treating each transition as an atomic training unit. To quantify the utility of these units, we introduce *Action Shapley*.

We define Action Shapley to measure the marginal contribution of each data point $k \in \mathcal{D}$ to an agent's performance. Contributions are quantified via a utility function $\mathcal{U}$, defined here as the cumulative reward achieved by the agent—an implicit function of the RL algorithm $\mathcal{A}$. We assume ϕ satisfies the standard Shapley axioms: nullity, symmetry, and linearity. Under these axioms, the Action Shapley value for a data point k is given by:

$$\phi_k = C_f \sum_{d \subseteq \mathcal{D} \setminus \{k\}} \frac{\mathcal{U}(d \cup \{k\}) - \mathcal{U}(d)}{\binom{n-1}{|d|}} \tag{1}$$

where $\mathcal{D} = \{k_1, \ldots, k_n\}$ is a multiset of n transitions and C_f is a normalization constant. The denominator weights the marginal contribution by the subset cardinality, allowing Action Shapley to account for data redundancy and subsume leave-one-out (LOO) valuation. If an agent fails to achieve its objective on a subset d, we set $\mathcal{U}(d) = \text{null}$. Computing Eq. 1 exactly is $\mathcal{O}(2^n)$, which is computationally prohibitive for large datasets.

We observe that a viable world model requires a threshold number of data points to succeed. To mitigate exponential complexity, we propose an algorithm (Algorithm 1) that leverages the concept of *cut-off cardinality* (θ). Algorithm 1 utilizes a top-down accumulator pattern, performing a breadth-first traversal of the power set tree starting from the full dataset $\mathcal{D}$.

The algorithm iterates through cardinalities in descending order. Within each cardinality, it checks for agent failure count error bound (ϵ). If failures reach a user-defined threshold ϵ, the algorithm terminates, setting the cut-off cardinality θ_k to the current level plus one. This early stopping identifies *indispensable* data points—those without which the agent consistently fails.

The computational effort P_{comp} of Algorithm 1 relative to exhaustive search is governed by the global cut-off cardinality $\theta_{k_{max}}$:

$$P_{comp} = 1 - \frac{2^{\theta_{k_{max}}}}{2^n} \tag{2}$$

Algorithm 1 Action Shapley Computation (ASC) for Training Action k

Require: Dataset $\mathcal{D}$, action k, RL algorithm $\mathcal{A}$, utility $\mathcal{U}$, error bound ϵ, constant C_f
Ensure: Action Shapley ϕ_k, cut-off cardinality θ_k

$flag \leftarrow 0, i \leftarrow n-1, sum \leftarrow 0, \theta_k \leftarrow 1$
while $i > 1$ **do**
  $mem \leftarrow 0$
  for all $d \in \mathcal{D}_i \setminus \{k\}$ **do**
    if $\mathcal{U}(d \cup \{k\}) =$ null **or** $\mathcal{U}(d) =$ null **then**
      $mem \leftarrow mem + 1$
      if $mem = \epsilon$ **then**
        $\theta_k \leftarrow i + 1, flag \leftarrow 1$
        break
      end if
    else
      $sum \leftarrow sum + C_f \frac{\mathcal{U}(d \cup \{k\}) - \mathcal{U}(d)}{\binom{n-1}{|d|}}$
    end if
  end for
  if $flag = 1$ **then**
    break
  else
    $i \leftarrow i - 1$
  end if
end while
return $\phi_k = sum, \theta_k$

While Algorithm 1 provides an efficient mechanism for identifying indispensable training actions via cut-off cardinality, its early stopping criterion introduces bias and does not, in general, yield an unbiased estimate of the Action Shapley value. To address this limitation, Algorithm 2 offers an unbiased Monte Carlo estimator by uniformly sampling subsets across cardinalities, at the cost of increased computation. We therefore combine the two approaches into a hybrid procedure: Algorithm 1 is first used to identify a cut-off cardinality θ_k, capturing the minimum subset size required for non-null utility when training action k is included, after which Algorithm 2 estimates the Action Shapley value via unbiased Monte Carlo sampling restricted to subset cardinalities $i \geq \theta_k$. This hybrid approach reduces computational cost by focusing estimation on regions of the power set where contributions are empirically significant, while retaining the interpretability of cut-off–based pruning and the statistical guarantees of unbiased Shapley estimation under mild assumptions.

3 EXPERIMENTS

We investigated the effectiveness of Action Shapley with four case studies. For case studies concerning VM right-sizing and load balancing, AWS EC2 instances have been deployed, incorporating the *Stress* tool to facilitate dynamic workload simulations. In the context of the database tuning case study, we have employed a Nutanix AHV-based virtualized infrastructure, utilizing the *HammerDB*

Algorithm 2 Unbiased Action Shapley Estimation

Require: $\mathcal{D}$ where $|\mathcal{D}| = n$, $k \in \mathcal{D}$, $\mathcal{A}, \mathcal{U}(\cdot; \mathcal{A})$, M is sample budget per cardinality
Ensure: Unbiased estimate $\hat{\phi}_k$

$\hat{\phi}_k \leftarrow 0$
for $i = 0$ **to** $n - 1$ **do**
  for $m = 1$ **to** M **do**
    Sample $d \subseteq \mathcal{D} \setminus \{k\}$ uniformly such that $|d| = i$
    if $\mathcal{U}(d \cup \{k\}) \neq$ null **and** $\mathcal{U}(d) \neq$ null **then**
      $\hat{\phi}_k \leftarrow \hat{\phi}_k + \frac{C_f}{M} \cdot \frac{\mathcal{U}(d \cup \{k\}) - \mathcal{U}(d)}{\binom{n-1}{i}}$
    end if
  end for
end for
return $\hat{\phi}_k$

tool for dynamic SQL load simulation. The case study focusing on Kubernetes management entails the utilization of a Nutanix Kubernetes (K8s) engine cluster, with *cassandra-stress* employed for workload simulations. In each case study, the training dataset is comprised of a collection of time series corresponding to different configurations, referred to as actions. These time series are measured over a 24-hour period at a 1-minute sampling rate. It is important to note that, across all case studies, the error bound (ϵ) is consistently fixed at 1. Further elaboration on the specifics of data collection is available in Appendix.

Action Shapley is agnostic to the specific instantiations of the RL algorithm and the environment model algorithm. For the environment model, we use a radial basis function (RBF) network (He et al., 2019), along with principal component analysis (PCA)-based pre-training (Genovese et al., 2019), for the $(state_{prev}, action_{prev}) \rightarrow (state_{cur}, reward_{cur})$ mapping. We choose two RL algorithms: SAC-PID (Yu et al., 2022) and PPO-PID (Shuprajhaa et al., 2022). Equation 3 shows the policy update recurrence (e = threshold − aggregated state statistic), the time step ((δt)), and learning parameters, $[k_p, k_i, kd]$.

$$a_i = a_{i-1} + [kp \quad ki \quad kd] \cdot \begin{bmatrix} e & (\delta t)e & \frac{e}{\delta t} \end{bmatrix} \tag{3}$$

For the VM right-sizing case study, we have five different data points: $\langle a_1 : (2, 2), a_2 : (2, 4), a_3 : (2, 8), a_4 : (4, 16), a_5 : (8, 32)\rangle$. Each pair represents $(\text{vCPU count}, \text{Memory Size})$. The training dataset consists of five time series of CPU usage, each with 1440 data points. While p50 is used for the aggregated state statistic, the goal of the RL agent is to bring the p50 CPU usage below 90%. The starting action for the RL loop is $(6, 14)$. The error bound, ϵ, is set to 1.

For the load balancing case study, we have five different data points: $\langle w_1 : (8, 16), w_2 : (8, 12), w_3 : (8, 2), w_4 : (1, 2), w_5 : (1, 16)\rangle$. Each pair represents $(\#\text{ of CPU workers}, \#\text{ of memory workers})$. The training dataset consists of five time series of CPU usage, each with 1440 data points. While p5 is used for the aggregated state statistic, the goal of the RL agent is to bring the p5 CPU usage below 70%. The starting action for the RL loop is $(5, 10)$. The error bound, ϵ, is set to 1.

For the database tuning case study, we have six different data points: $\langle p_1 : (1, 1), p_2 : (4, 4), p_3 : (6, 3), p_4 : (8, 4), p_5 : (8, 8), p_6 : (10, 10)\rangle$. Each pair represents $(\text{vCPU count}, \text{Memory Size})$. The training dataset consists of six time series of CPU usage, each with 1440 data points. While p90 is used for the aggregated state statistic, the goal of the RL agent is to bring the p90 CPU usage below 25%. The starting action for the RL loop is $(5, 2)$. The error bound, ϵ, is set to 1.

In the Kubernetes Management case study, we employ 15 different data points $\langle r_1 : (1 \times 10^6, 10), r_2 : (1 \times 10^6, 25), r_3 : (1 \times 10^6, 50), r_4 : (1 \times 10^6, 75), r_5 : (1 \times 10^6, 100), r_6 : (2 \times 10^6, 10), r_7 : (2 \times 10^6, 25), r_8 : (2 \times 10^6, 50), r_9 : (2 \times 10^6, 75), r_{10} : (2 \times 10^6, 100), r_{11} : (3 \times 10^6, 10), r_{12} : (3 \times 10^6, 25), r_{13} : (3 \times 10^6, 50), r_{14} : (3 \times 10^6, 75), r_{15} : (3 \times 10^6, 100)\rangle$. Each training configuration is denoted by a pair representing the tuple (write rate, thread count). The training dataset encompasses 15 time series data sets, each comprising 1440 data points, measuring response latency. The aggregated state statistic is determined by the p99.9 metric, and the primary

Table 1: Action Shapley for VM right-sizing case study. a_3, a_4, a_5 are indispensable actions.

	AS SAC-PID	AS PPO-PID
a_1	-3.96	-3.90
$\mathbf{a_2}$	**-1.84**	**-1.81**
a_3	ind.	ind.
a_4	ind.	ind.
a_5	ind.	ind.

Table 2: Action Shapley for load balancing case study

	AS SAC-PID	AS PPO-PID
w_1	0.57	0.57
w_2	1.02	1.10
$\mathbf{w_3}$	**8.31**	**8.27**
w_4	3.61	3.61
w_5	5.55	5.51

Table 3: Action Shapley for database tuning case study. p_1 is indispensable action.

	AS SAC-PID	AS PPO-PID
p_1	Ind.	ind.
p_2	1.11	1.12
$\mathbf{p_3}$	**3.42**	**3.37**
p_4	1.75	1.72
p_5	0.19	0.21
p_6	-0.23	-0.22

objective of the RL agent is to reduce the p99.9 latency to below 100ms. The RL loop commences with an initial action of $(2.9 \times 10^6, 95)$. The error bound, ϵ, is set to 1.

In each of the aforementioned case studies, the primary objective for the RL agent is to attain a pre-assigned threshold for a state statistic. To achieve this, we formulate the reward function for the RL agent as: $(\text{threshold} - \text{aggregated state statistic}(t))$ if threshold $<$ aggregated state statistic(t). This reward function is designed to prompt the agent to efficiently meet the specified threshold.

4 RESULTS

4.1 ACTION SHAPLEY VALUES FOR TRAINING DATA

Table 1 displays the two Action Shapley (AS) values corresponding to two RL algorithms SAC-PID and PPO-PID, and five distinct training data points, denoted as $\langle a_1, a_2, a_3, a_4, a_5 \rangle$, within the VM right-sizing case study. Notably, it was identified that the training data points a_3, a_4, and a_5 are indispensable. a_2 exhibits the highest Action Shapley values, specifically -1.84 and -1.81, respectively for SAC-PID and PPO-PID. The global cut-off cardinality, denoted as θ, is set at 4, resulting in $P_{comp} = 50\%$ as defined by Equation 2. It is noteworthy that both SAC-PID and PPO-PID demonstrate congruent Action Shapley values and identical cut-off cardinality values.

Table 2 displays the two Action Shapley (AS) values corresponding to two RL algorithms SAC-PID and PPO-PID, and five distinct training data points, denoted as $\langle w_1, w_2, w_3, w_4, w_5 \rangle$, within the load balancing case study. w_3 exhibits the highest Action Shapley values, specifically 8.31 and 8.27, respectively for SAC-PID and PPO-PID. The global cut-off cardinality, denoted as θ, is set at 3, resulting in $P_{comp} = 50\%$ as defined by Equation 2. It is noteworthy that both SAC-PID and PPO-PID demonstrate congruent Action Shapley values and identical cut-off cardinality values.

Table 3 displays the two Action Shapley (AS) values corresponding to two RL algorithms SAC-PID and PPO-PID, and six distinct training data points, denoted as $\langle p_1, p_2, p_3, p_4, p_5, p_6 \rangle$, within the database tuning case study. Notably, it was observed that the training data point p_1 is deemed indispensable, whereas the remaining five training data points are considered dispensable. p_3 exhibits the highest Action Shapley values, specifically 3.42 and 3.37, respectively for SAC-PID and PPO-PID. The global cut-off cardinality, denoted as θ, is set at 4, resulting in $P_{comp} = 75\%$ as defined by Equation 2. It is noteworthy that both SAC-PID and PPO-PID demonstrate congruent Action Shapley values and identical cut-off cardinality values.

Table 4: Action Shapley for K8s management case study.

	AS SAC-PID	AS PPO-PID
r_1	-0.7	0.68
r_2	0.53	0.54
r_3	0.61	0.62
r_4	-0.13	-0.12
r_5	0.12	0.11
r_6	-0.7	-0.7
r_7	-0.24	-0.25
r_8	0.65	0.65
r_9	0.42	0.42
r_{10}	0.08	0.07
r_{11}	-1.16	-1.17
r_{12}	-0.25	-0.24
$\mathbf{r_{13}}$	**0.77**	**0.77**
r_{14}	-0.31	-0.31
r_{15}	0.019	0.019

Table 4 displays the two Action Shapley (AS) values corresponding to two RL algorithms SAC-PID and PPO-PID, and 15 distinct training data points, denoted as

Table 5: This table summarizes the number of training data points and related parameters for four different case studies.

Case Study	VM Right-Sizing	Load Balancing	Database Tuning	K8s Management
No of Training Data Points	5	5	6	15
Cut-off Cardinality	4	3	4	5
No of Indispensable Data Points	3	0	1	0
No of Data Points to be Chosen	1	3	3	5
No of Possible Data Points	2	5	5	15

$\langle r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}, r_{15} \rangle$, within the K8s case study. Notably, r_{13} exhibits the highest Action Shapley values, specifically 0.77 for both SAC-PID and PPO-PID. The global cut-off cardinality, denoted as θ, is set at 5, resulting in $P_{comp} = 99.9\%$ as defined by Equation 2. It is noteworthy that both SAC-PID and PPO-PID demonstrate congruent Action Shapley values and identical cut-off cardinality values.

4.2 Validation of Action Shapley Based Training Data Selection Policy

In the preceding sections of this paper, we introduced a training data selection policy, which can be succinctly summarized as follows: the optimal training dataset should include the maximum number of training data points, up to the predefined global cut-off cardinality, with the highest average Action Shapley value. The selection process is further refined when indispensable data points are considered. For instance, Table 5 provides a demonstration in the context of the VM right-sizing case study. In this scenario, where we need to choose a single data point from 2 dispensable data points, there are $\binom{2}{1} = 2$ available options. In both the load balancing and database tuning cases, the task involves selecting 3 data points out of 5 dispensable data points, resulting in $\binom{5}{3} = 10$ potential choices for these specific case studies. Lastly, in the K8s management case study, where the selection involves 5 data points out of 15 dispensable data points, there are $\binom{15}{5} = 3,003$ possible choices.

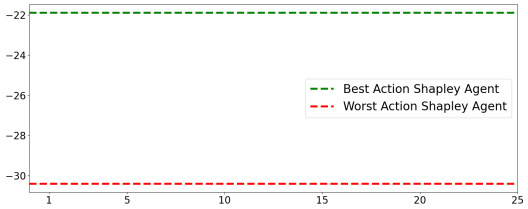

Figure 1: This plot compares the cumulative reward scores for the best Action Shapley vs the worst Action Shapley agents for the VM right-sizing case study.

Our empirical validation process involving four case studies consists of two steps. Firstly, we assess whether the best Action Shapley agent, generated from the training set with the highest average Action Shapley value and an element count matching the cut-off cardinality, achieves a higher cumulative reward compared to the worst Action Shapley agent. The worst Action Shapley agent is derived from the training set with the lowest average Action Shapley value and the same number of elements as the cut-off cardinality. Subsequently, we investigate whether the best Action Shapley agent consistently outperforms the majority of other agents. To answer this question, we conduct a series of 25 episodes, each involving multiple random selections. Each random selection is characterized by a training action set size equivalent to the cut-off cardinality. In light of the comparable Action Shapley values generated by both SAC-PID and SAC-PPO, we

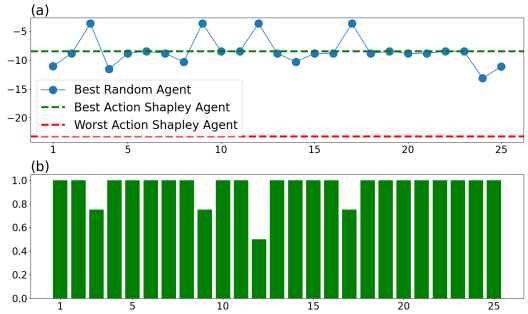

Figure 2: Action Shapley (AS) for the load balancing case study. (a) Comparison of cumulative rewards. (b) Fractions of agents based on 4 random training datasets with lower cumulative rewards than that of the best Action Shapley agent for 25 episodes.

choose to utilize SAC-PID-based Action Shapley values in this section for the sake of convenience and without sacrificing generality.

For the VM right-sizing case study, our options for training action sets are limited to two. Consequently, our validation process is distilled into a comparative analysis between the best Action Shapley agent and the worst Action Shapley agent. Illustrated in Figure 1, the cumulative reward score for the superior Action Shapley agent registers at -21.9, while that for the inferior counterpart is recorded at -30.4. This discernible discrepancy in cumulative rewards substantiates the efficacy of the proposed training action selection policy.

In the context of the load balancing case study, a judicious selection of three training actions from a pool of five is imperative. The cumulative rewards, as depicted in Figure 2(a), elucidate a noteworthy performance disparity between the best and worst Action Shapley agents, with respective scores of -8.49 and -23.2. Notably, the optimal Action Shapley agent exhibits comparable efficacy to the highest achiever among randomly chosen sets across 25 episodes. Figure 2(b) further presents the proportion of randomly assembled training action sets resulting in agents with cumulative rewards surpassing that of the premier Action Shapley agent. The discernible trend reveals a mere 5 out of 100 randomly derived agents outperforming the optimal Action Shapley agent. Consequently, a confident assertion can be made that this case study effectively validates the viability of our chosen training action selection policy.

For the tuning case study, we faced the task of selecting four training datasets out of a pool of six. As depicted in Figure 3(a), the cumulative reward for the best Action Shapley agent, standing at -2.42, surpasses that of the worst Action Shapley agent, which is -21. Additionally, the figure illustrates that the performance of the best Action Shapley agent is comparable to the top-performing agent derived from four randomly selected training action sets across each of the 25 episodes. Figure 3(b) presents the fraction of randomly chosen training action sets that yield an agent with a cumulative reward lower than that of the best Action Shapley agent. While 31 out of 100 random selections perform better than the best Action Shapley agent, it's crucial to note that this difference is marginal, as evidenced by the top subplot comparing the best Action Shapley performance with the top performers in each of the 25 episodes. Consequently, we assert that this case study serves as a validation of our training action selection policy.

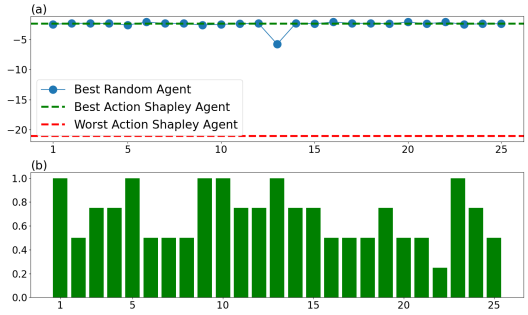

Figure 3: Validation of Action Shapley based selection policy for database tuning. (a) Comparisons of cumulative rewards among the best Action Shapley agent, the worst Action Shapley agent, and the best of 4 random training action sets for 25 different episodes. (b) Fractions of agents based on 4 random training datasets with lower cumulative rewards than that of the best Action Shapley agent for 25 episodes.

In our K8s management case study, we had a pool of 3,003 options for selecting the training action set. Figure 4(a) displays the cumulative reward, indicating that the best Action Shapley agent achieved a score of -499, surpassing the worst Action Shapley agent, which scored -621. Furthermore, it demonstrates that the best Action Shapley agent's performance is on par with the top-performing agent from 30 randomly selected sets across each of the 25 episodes. Figure 4(b) presents the proportions of random training action sets resulting in an agent with a cumulative reward lower than that of the best Action Shapley agent. Notably, in this figure, it's observed that 125 out of 625 random selections outperform the best Action Shapley agent in terms of cumulative reward. However, it is crucial to highlight that despite this, the top performers from these random selections exhibit performance levels comparable to the best Action Shapley agent.

4.3 Comparison to Baseline Results

All conducted case studies consistently validate the effectiveness of the proposed Action Shapley-based training data selection policy and computation algorithm. However, it is imperative to assess

the efficacy of Action Shapley by benchmarking it against a baseline study that assumes the utilization of all available training data points for training the environment model. Specifically, in the VM right-sizing case study, the baseline study incorporates 5 training data points instead of the specified cutoff cardinality of 4. In the load balancing case study, the baseline study utilizes 5 training data points, deviating from the specified cutoff cardinality of 3. Similarly, in the database tuning case study, the baseline study integrates 6 training data points rather than the stipulated cutoff cardinality of 4. Lastly, in the K8s management case study, the baseline study employs 15 training data points, exceeding the defined cutoff cardinality of 5. In summary, the proposed Shapley analytics consistently lead to substantial reductions in the number of required training data points, ranging between 20% and 67%. The second evaluative dimension considers the cumulative reward attained by both the baseline agent and the optimal Action Shapley agent. Notably, in three of the four case studies (VM right-sizing, load balancing, and K8s management), the optimal Action Shapley agent demonstrates a significant performance advantage over the baseline agent. Specifically, for VM right-sizing, the values are -37.7 compared to -21.9; for load balancing, -9.1 compared to -8.49; and for K8s management, -561.4 compared to -499. In these instances, the optimal Action Shapley agent consistently outperforms the baseline agent by a substantial margin, ranging from 7% to 42%. However, in the database tuning case study, a marginal 3.9% superiority in cumulative reward is observed for the baseline agent.

5 Related Work

The Shapley value, originating in cooperative game theory, provides a principled solution for fairly distributing total reward among contributors and has been widely applied across economics, voting, resource allocation, and bargaining problems (Shapley, 1953b; Roth, 1988). In machine learning, Shapley-inspired methods have become foundational for data valuation (Ghorbani & Zou, 2019a), feature attribution (Lundberg & Lee, 2017), and model explanation (Sundararajan & Najmi, 2020). These model explanation techniques fall under additive feature attribution methods, which also include LIME (Ribeiro et al., 2016), DeepLIFT (Shrikumar et al., 2016), and layer-wise relevance propagation (Bach et al., 2015). Recent work surveys both theoretical foundations and scalable approximations for Shapley computation, including deterministic and stochastic estimators (Qin et al., 2025), as well as efficient proxies such as DU-Shapley (Garrido-Lucero et al., 2024) and threshold-based estimators (Wang et al., 2023).

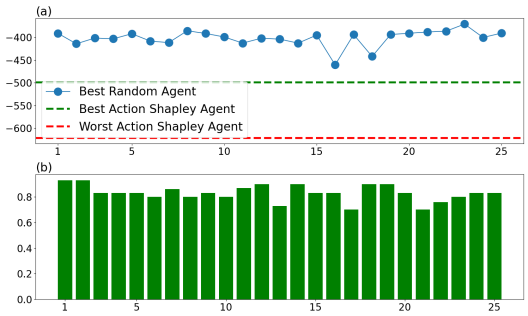

Figure 4: Validation of Action Shapley based selection policy for K8s management. (a) Comparisons of cumulative rewards among the best Action Shapley agent, the worst Action Shapley agent, and the best of 30 random training action sets for 25 different episodes. (b) Fractions of agents based on 30 random training datasets with lower cumulative rewards than that of the best Action Shapley agent for 25 episodes.

In the context of reinforcement learning, training data selection and sample efficiency remain significant challenges due to the deadly triad of bootstrapping, function approximation, and off-policy learning (Sutton & Barto, 2018). Prior work has approached sample efficiency, robustness, and exploration from diverse perspectives such as hindsight experience conditioning (Harutyunyan et al., 2019), return decomposition (Arjona-Medina et al., 2019), adaptive sampling and optimal sample selection (Rachelson et al., 2011), active learning for RL (Li et al., 2011), and statistical sample complexity bounds (Gheshlaghi Azar et al., 2013; Jin et al., 2018). While these efforts improve learning performance or theoretical guarantees, they do not provide a unified, valuation-based framework for selecting training transitions — particularly for model learning in world models.

Shapley values have also been studied for credit assignment and interpretation in RL. Recent work proposes scalable methods for computing Shapley explanations in reinforcement learning that handle temporal dependencies and off-policy data (Beechey & Özgür Şimşek, 2025), and cooperative

variants have been explored in multi-agent settings (Li et al., 2021b). However, these efforts focus on interpretability or attributions over policies and agents, not on data selection for environment model training.

In general machine learning, data valuation has seen rapid recent growth. Meta-learning approaches that jointly optimize data values and predictors have been proposed (Yoon et al., 2019). Extensions such as distributional Shapley values (Ghorbani & Zou, 2019b) and asymmetric Shapley values (Zheng et al., 2025) aim to capture structural and temporal dependencies in modern learning pipelines. Moreover, scalable data valuation methods based on gradients, influence functions, and surrogate utility approximations have been explored for large datasets (Koh & Liang, 2017; Pruthi et al., 2020).

Despite these advances, there is a lack of dedicated studies on applying Shapley-based data valuation specifically for reinforcement learning world model training. Technical challenges include the combinatorial complexity of Shapley computation, dynamic and temporally dependent data, data drift during environment interactions, and high training costs for model-based RL. Action Shapley addresses this gap by defining a principled value for each transition and proposing scalable estimation algorithms that balance bias, computational cost, and interpretability. Compared to classic techniques such as leave-one-out (LOO) valuation (Cook & Weisberg, 1982), Shapley-based approaches better account for redundancy and interaction effects among transitions.

6 Conclusion

In this work, we introduced Action Shapley, a principled data valuation framework for reinforcement learning that assigns credit at the level of individual training transitions. By framing each transition as an atomic training action and grounding valuation in Shapley axioms, Action Shapley provides a theoretically sound mechanism for quantifying the contribution of data to RL agent performance, with particular relevance to world model training.

We showed that naive computation of Action Shapley is intractable, and proposed two complementary algorithms: a fast, top-down cut-off–based heuristic that identifies indispensable data points and a Monte Carlo estimator that provides unbiased Shapley estimates. By combining these approaches, we obtain a practical hybrid method that balances computational efficiency, interpretability, and correctness. The notion of cut-off cardinality emerges as a key concept, capturing the minimum data mass required for successful environment modeling and enabling principled data selection under resource constraints.

Through empirical evaluation, we demonstrated that Action Shapley can effectively identify high-value transitions, reduce redundant data, and guide the construction of compact yet performant training datasets. These results suggest that data-centric perspectives, when coupled with cooperative game–theoretic foundations, offer a powerful lens for understanding and improving reinforcement learning systems.

While Action Shapley provides a principled and practical framework for data valuation in world model development for model-based RL, its computation remains nontrivial at scale. Although the proposed hybrid algorithm substantially reduces cost, further gains may be achieved through adaptive sampling, variance-aware allocation of Monte Carlo budgets across subset cardinalities, and confidence-bound–driven early stopping, particularly in large-scale or online settings.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

SL Bangare, S Gupta, M Dalal, and A Inamdar. Using node. js to build high speed and scalable backend database server. In *Proc. NCPCI. Conf*, volume 2016, pp. 19, 2016.

Daniel Beechey and Özgür Şimşek. Approximating shapley explanations in reinforcement learning, 2025. URL https://arxiv.org/abs/2511.06094.

Hugh Chen, Ian C. Covert, Scott M. Lundberg, and Su-In Lee. Algorithms to estimate shapley value feature attributions, 2022. URL https://arxiv.org/abs/2207.07605.

R Dennis Cook and Sanford Weisberg. Residuals and influence in regression, 1982.

K Derdus, V Omwenga, and P Ogao. Virtual machine sizing in virtualized public cloud data centres. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 5(4), 2019.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Qiaoyi Fang, Weiyu Du, Hang Wang, and Junshan Zhang. Towards unraveling and improving generalization in world models. *arXiv preprint arXiv:2501.00195*, 2024.

Joseph Futoma, Michael C Hughes, and Finale Doshi-Velez. Popcorn: Partially observed prediction constrained reinforcement learning. *arXiv preprint arXiv:2001.04032*, 2020.

Felipe Garrido-Lucero, Benjamin Heymann, Maxime Vono, Patrick Loiseau, and Vianney Perchet. Du-shapley: A shapley value proxy for efficient dataset valuation, 2024. URL https://arxiv.org/abs/2306.02071.

Angelo Genovese, Vincenzo Piuri, Konstantinos N Plataniotis, and Fabio Scotti. Palmnet: Gabor-pca convolutional networks for touchless palmprint recognition. *IEEE Transactions on Information Forensics and Security*, 14(12):3160–3174, 2019.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3): 325–349, 2013.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251. PMLR, 2019a.

Amirata Ghorbani and James Zou. Distributional shapley values. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019b.

Christian Gumbsch, Noor Sajid, Georg Martius, and Martin V Butz. Learning hierarchical world models with adaptive temporal abstractions from discrete latent dynamics. In *The Twelfth International Conference on Learning Representations*, 2023.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2555–2565, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. *Advances in neural information processing systems*, 32, 2019.

Qingfeng He, Himan Shahabi, Ataollah Shirzadi, Shaojun Li, Wei Chen, Nianqin Wang, Huichan Chai, Huiyuan Bian, Jianquan Ma, Yingtao Chen, et al. Landslide spatial modelling using novel bivariate statistical based naïve bayes, rbf classifier, and rbf network machine learning algorithms. *Science of the total environment*, 663:1–15, 2019.

Peter Jin, Kurt Keutzer, and Sergey Levine. Regret minimization for partially observable deep reinforcement learning. In *International conference on machine learning*, pp. 2342–2351. PMLR, 2018.

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International conference on artificial intelligence and statistics*, pp. 1701–1710. PMLR, 2018.

Abeer Abdel Khaleq and Ilkyeun Ra. Intelligent autoscaling of microservices in the cloud for real-time applications. *IEEE Access*, 9:35464–35476, 2021.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Fei Wu, and Jun Xiao. Shapley counterfactual credits for multi-agent reinforcement learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 934–942, 2021a.

Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Fei Wu, and Jun Xiao. Shapley counterfactual credits for multi-agent reinforcement learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 934–942, 2021b.

Lihong Li, Michael L Littman, Thomas J Walsh, and Alexander L Strehl. Knows what it knows: a framework for self-aware learning. *Machine learning*, 82(3):399–443, 2011.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Léopold Maytié, Roland Bertin Johannet, and Rufin VanRullen. Multimodal dreaming: A global workspace approach to world model-based reinforcement learning. *arXiv preprint arXiv:2502.21142*, 2025.

Cristian Meo, Mircea Lica, Zarif Ikram, et al. Masked generative priors improve world models sequence modelling capabilities. *arXiv preprint arXiv:2410.07836*, 2024.

Sambit Kumar Mishra, Bibhudatta Sahoo, and Priti Paramita Parida. Load balancing in cloud computing: a big picture. *Journal of King Saud University-Computer and Information Sciences*, 32 (2):149–158, 2020.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Lei Qin, Yingqiu Zhu, Shaonan Liu, Xingjian Zhang, and Yining Zhao. The shapley value in data science: Advances in computation, extensions, and applications. *Mathematics*, 13(10):1581, 2025.

Emmanuel Rachelson, François Schnitzler, Louis Wehenkel, and Damien Ernst. Optimal sample selection for batch-mode reinforcement learning. In *International Conference on Agents and Artificial Intelligence*, volume 2, pp. 41–50. SCITEPRESS, 2011.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, et al. Autopilot: workload autoscaling at google. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pp. 1–16, 2020.

L Shapley. Quota solutions op n-person games1. *Edited by Emil Artin and Marston Morse*, pp. 343, 1953a.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10): 1095–1100, 1953b.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

T Shuprajhaa, Shiva Kanth Sujit, and K Srinivasan. Reinforcement learning based adaptive pid controller design for control of linear/nonlinear unstable processes. *Applied Soft Computing*, 128: 109450, 2022.

Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International conference on machine learning*, pp. 9269–9278. PMLR, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jiachen T. Wang, Yuqing Zhu, Yu-Xiang Wang, Ruoxi Jia, and Prateek Mittal. Threshold knn-shapley: A linear-time and privacy-friendly approach to data valuation, 2023. URL `https://arxiv.org/abs/2308.15709`.

Junxiong Wang, Immanuel Trummer, and Debabrota Basu. Udo: universal database optimization using reinforcement learning. *arXiv preprint arXiv:2104.01744*, 2021.

Yue Xu, Wenjun Xu, Zhi Wang, Jiaru Lin, and Shuguang Cui. Load balancing for ultradense networks: A deep reinforcement learning-based approach. *IEEE Internet of Things Journal*, 6(6): 9399–9412, 2019.

Lenar Yazdanov and Christof Fetzer. Vscaler: Autonomic virtual machine scaling. In *2013 IEEE Sixth International Conference on Cloud Computing*, pp. 212–219. IEEE, 2013.

Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 12009–12018. PMLR, 2021.

Jinsung Yoon, Sercan O. Arik, and Tomas Pfister. Data valuation using reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Xinyi Yu, Yuehai Fan, Siyu Xu, and Linlin Ou. A self-adaptive sac-pid control approach based on reinforcement learning for mobile robots. *International journal of robust and nonlinear control*, 32(18):9625–9643, 2022.

Xi Zheng, Yinghui Huang, Xiangyu Chang, Ruoxi Jia, and Yong Tan. Rethinking data value: Asymmetric data shapley for structure-aware valuation in data markets and machine learning pipelines, 2025. URL `https://arxiv.org/abs/2511.12863`.

A APPENDIX

A.1 NOTATION

s	Environment state
a	Agent action
r	Reward received by an agent
$\mathcal{D}$	Training data set
$\mathcal{A}$	RL algorithm
$\mathcal{U}$	Evaluation function for resulting RL agent
ϕ_k	Action Shapley for a training data point k
C_f	Arbitrary constant for Action Shapley computation
n	Number of data points in a data set $\mathcal{D}$
ϵ	Error bound
mem	Local variabel for failure memoization
θ_k	Cut-off cardinality
$\theta_{k_{max}}$	Global cut-off cardinality
k_p	Proportional parameter in PID
k_i	Integral parameter in PID
k_d	Differential parameter in PID
$\binom{n}{r}$	n choose r
$\|d\|$	Number of elements in a set, $\{\mathrm{d}\}$
e	the difference between threshold and aggregated state statistic
δt	the time step size in reinforce learning loop

A.2 DATA COLLECTION FOR THE VM RIGHT-SIZING CASE STUDY

VM right-sizing entails a decision problem that recognizes minimum possible VM size without compromising a pre-assigned performance threshold. In this case study, we use CPU utilization as the state variable. The training action is represented by a pair of (# of CPU cores, memory size (GB)). The five training actions used are: $\{a_1 : (2, 2), a_2 : (2, 4), a_3 : (2, 8), a_4 : (4, 16), a_5 : (8, 32)\}$. Figure 5(a) shows training action samples and corresponding aggregated state statistics (i.e., 50-th percentile as mentioned in Table 6). The environment runs a rectangular workload, simulated using *stress* library. The simulated rectangular workload has a time period of 600 s: a high-stress, peak phase of 500 s is followed by a `sleep` phase of 100 s. The peak of the workload uses the stress command: {*sudo stress –io 4 –vm 2 –vm-bytes 1024M –timeout 500s*}. Essentially, the peak is running 4 *I/O* stressors and 2 VM workers spinning on malloc with 1024 MB per worker for 500 s. This workload runs for 24 hr. with a 1 minute sampling interval. Overall, five time series data for CPU utilization together represent the state space. The goal of the RL agent is to choose the most cost-efficient configuration to bring down the aggregated state statistic below a pre-assigned threshold of 90% without violating constraints. Until the goal is satisfied, the RL agent is penalized in every time step by the deviation of the aggregated state statistic from its set point. As shown in Table 6, two other parameters are: Initial Action $= (6, 14)$ and $\epsilon = 1$.

A.3 Data Collection for the Load Balancing Case Study

Load balancing deals with a decision problem that recognizes how to distribute traffic/loads to different servers. For this, it is important to estimate the dynamic capacity of a VM. In this case study, we use CPU utilization as the state variable. The action space is represented by a pair of (# of CPU workers, # of memory workers). The five training actions used are: $\{w_1 : (8, 16), w_2 : (8, 12), w_3 : (8, 2), w_4 : (1, 2), w_5 : (1, 16)\}$. Figure 5(b) shows training action samples and corresponding aggregated state statistics (i.e., 5-th percentile as mentioned in Table 6). The environment for this case study is an AWS t3a.medium EC2 VMs with 2 vCPUs, 4GB RAM. Each training workload has a time period of 600 seconds: a high-stress, peak phase of 500 seconds is followed by an inactive `sleep` phase of 100 seconds. Here are five different training workloads during high-stress phases: {*stress –cpu 8 –io 4 –vm 16 –vm-bytes 1024M –timeout 500s*}, {*stress –cpu 8 –io 4 –vm 12 –vm-bytes 1024M –timeout 500s*},{*stress –cpu 8 –io 4 –vm 2 –vm-bytes 1024M –timeout 500s*}, {*stress –cpu 1 –io 4 –vm 2 –vm-bytes 1024M –timeout 500s*}, and {*stress –cpu 1 –io 4 –vm 16 –vm-bytes 1024M –timeout 500s*}. The only differences among these workloads are the number of cpu workers and the number of memory workers.Each of these workloads is run for 3 hours with 5-minute sampling interval. Overall, the state space consists of five time series for CPU utilization metrics. The goal of the RL agent is to choose the highest possible workload intensity by packing more CPU and memory workers without violating a pre-assigned performance goal of keeping the 5-th percentile of CPU utilization below 70%. Until the goal is satisfied, the RL agent is penalized in every time step by the deviation of the aggregated state statistic from its set point. As shown in Table 6, two other parameters are: Initial Action $= (5, 10)$ and $\epsilon = 1$.

A.4 Data Collection for the Database Tuning Case Study

Two critical SQL database tuning parameters are the number of server CPU cores and server memory size (GB). The tuning of these parameters requires painstaking trial and error which can be replaced with RL for precision and efficiency. In this case study, we use CPU utilization as the state variable. The action space is represented by a pair of (# of CPU cores, memory size (GB)).The six training actions used are: $\{p_1 : (1, 1), p_2 : (4, 4), p_3 : (6, 3), p_4 : (8, 4), p_5 : (8, 8), p_6 : (10, 10)\}$. Figure 5(c) shows training action samples and corresponding aggregated state statistics (i.e., 90-th percentile as mentioned in Table 6). The goal of the RL agent is to bring the aggregated state statistics below 25%. Against each training action, a SQL workload is simulated using ***HammerDB***. The corresponding server utilization data is collected for a duration of 172 minutes at a sampling interval of 30 seconds. The workload is triggered by a HammerDB-hosted Tcl code which simulates *two* SQL warehouse building processes by *two* SQL virtual workers. Overall, we have six time series data against each database server configurations as the state space. Until the goal is satisfied, the RL agent is penalized in every time step by the deviation of the aggregated state statistic from its set point. As shown in Table 6, two other parameters are: Initial Action $= (5, 2)$ and $\epsilon = 1$.

A.5 Data Collection for the Kubernetes Management Case Study

Kubernetes management often deals with a decision problem that recognizes the optimal workload parameters for a given Kubernetes cluster to maintain certain service level agreement criteria. For the given case study, we use Cassandra database workload with the thread count and lines written as the parameters. The goal of the RL agent is to maintain the p99.9 latency below a pre-assigned threshold of 100 milliseconds. The initial action points are 2.9×10^6 for written line count and 95 for thread count. For training, we use 15 different pairs spanned by three written line counts: $\{1\times10^6, 2\times10^6, 3\times10^6\}$ and five thread counts of $\{10, 25, 50, 75, 100\}$. Figure 5(d) shows training action samples and corresponding aggregated state statistics (i.e., 99.9-th percentile as mentioned in Table 6). We set $\epsilon = 1$.

A.6 Validation of Training Action Selection Policy

We have proposed an optimal training action selection policy: the best possible training action set includes as many training actions as the global cut-off cardinality with the highest possible Action Shapley values. On the other hand, the worst possible training action set includes as many training actions as the global cut-off cardinality with the lowest possible Action Shapley values. We validate this policy for four different case studies.

Table 6: Salient details of three different case studies

Case Study	VM Right-Sizing	Load Balancing	Database Tuning	K8s Management
State	p50 CPU usage	p5 CPU usage	p90 CPU usage	p99.9 latency
Threshold	$\leq 90\%$	$\leq 70\%$	$\leq 25\%$	$\leq 100ms$
Initial Action	$(6, 14)$	$(5, 10)$	$(5, 2)$	$(2.9 \times 10^6, 95)$
ϵ	1	1	1	1

A.6.1 Validation of Training Action Selection Policy for the VM Right-Sizing Case Study

Based on the Action Shapley computation, we determine that the cut-off cardinality is equal to 4 with 5 total number of training actions. It indicates $P_{comp} = 50\%$. With a_3, a_4, a_5 being the indispensable actions, $\langle a_2, a_3, a_4, a_5 \rangle$ has the highest possible mean Action Shapley value. On the other hand, $\langle a_1, a_3, a_4, a_5 \rangle$ has the lowest possible mean Action Shapley value. As we hypothesize, the former action set produced an RL agent with higher cumulative award of -21.9 vs -37.75. The former has a lower convergence time too: 13 vs 16. In comparison to the baseline RL agent (cumulative reward of -30.4 and convergence time of 21), both best and worst agents performs better.

A.6.2 Validation of Training Action Selection Policy for the Load Balancing Case Study

Based on the Action Shapley computation, we determine the cut-off cardinality is equal to 3 with 5 totral number of training actions, indicating that $P_{comp} = 75\%$. $\langle w_3, w_4, w_5 \rangle$ has the highest possible mean Action Shapley value. On the other hand, $\langle w_1, w_2, w_4 \rangle$ has the lowest possible mean Action Shapley value. As we hypothesize, the former action set produced an RL agent with a higher cumulative award of -8.49 vs -23.2. The former has a lower convergence time too: 10 vs 29. In comparison to the baseline RL agent (cumulative reward of -9.1 and convergence time of 12), only the best agent performs better.

A.6.3 Validation of Training Action Selection Policy for the Database Tuning Case Study

Based on the Action Shapley computation, we determine the cut-off cardinality is equal to 4 from 6 total actions. This indicates $P_{comp} = 75\%$. With p_1 being the indispensable actions, $\langle p_1, p_2, p_3, p_4 \rangle$ has the highest possible mean Action Shapley value. On the other hand, $\langle p_1, p_2, p_5, p_6 \rangle$ has the lowest possible mean Action Shapley value. As we hypothesize, the former action set produced an RL agent with a higher cumulative award of -2.42 vs -21. The former has a lower convergence time too: 7 vs 16. In comparison to the baseline RL agent (cumulative reward of -2.33 and convergence time of 6), both the best and worst agent agents performs poorly. In fact, the baseline agent performs marginally better than the best agent chosen by the Action Shapley policy.

A.6.4 Validation of Training Action Selection Policy for the Kubernetes Management Case Study

Based on the Action Shapley computation, we determine that the cut-off cardinality is equal to 5 from 15 total actions. This indicates that $P_{comp} = 99.9\%$. $\langle r_2, r_3, r_8, r_9, r_{13} \rangle$ has the highest possible mean Action Shapley value. On the other hand, $\langle r_1, r_6, r_{11}, r_{12}, r_{14} \rangle$ has the lowest possible mean Action Shapley value. As we hypothesized, the former action set produced an RL agent with a higher cumulative award of -499 vs -621. The former has a higher convergence time too: 89 vs 51. It is counter-intuitive that the former set has a higher converge time despite with a higher mean Action Shapley value. It could be due to complex system dynamics. In comparison to the baseline RL agent (cumulative reward of -561.4 and convergence time of 52), only the best agent performs better.

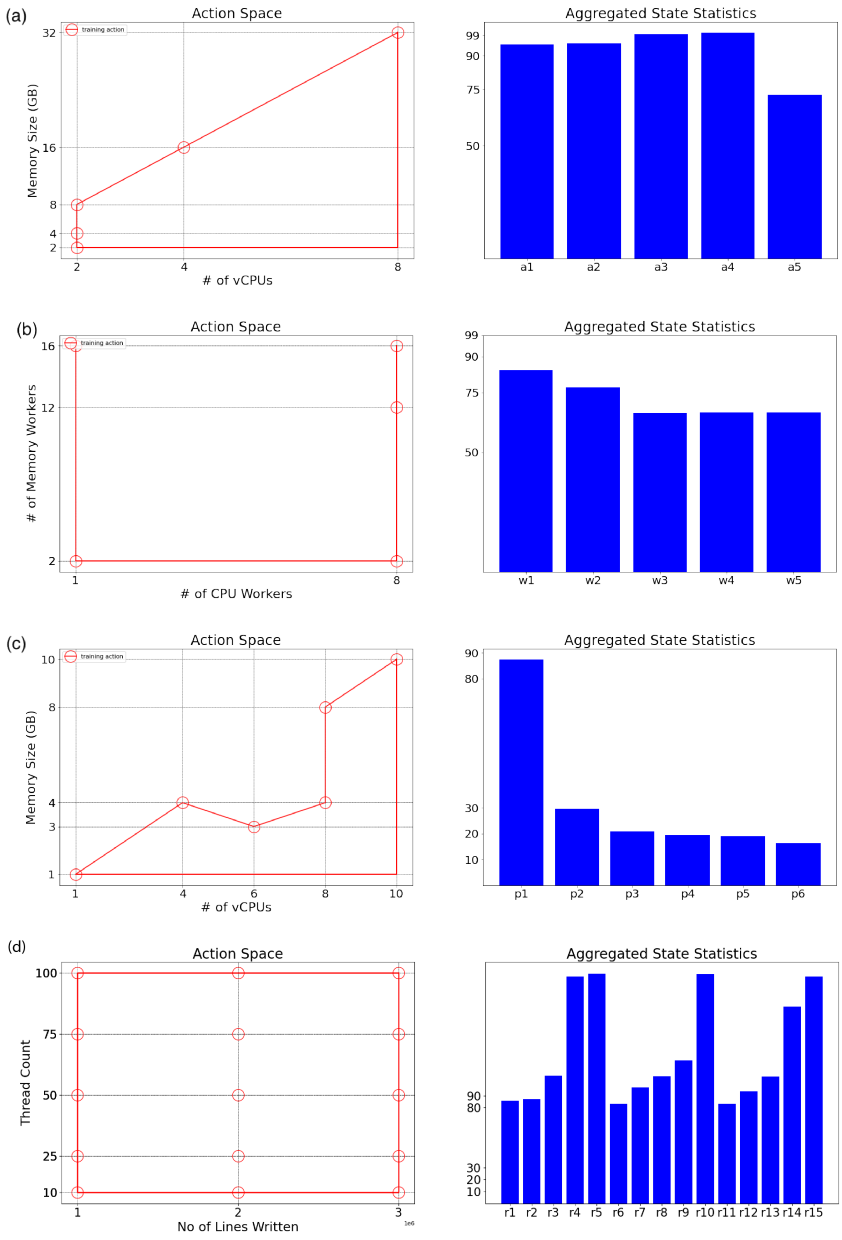

Figure 5: Different training actions and corresponding aggregated state statistics for three different case studies: (a) VM right-sizing case study. (b) Load balancing case study. (c) Database tuning case study. (d) Kubernetes management case study.

A.7 Cut-Off Cardinality

As we compute Action Shapley values for different training actions, we also compute the corresponding cut-off cardinality values. For each case study, we compute the corresponding global cardinality values as shown in Table 7. It shows for all case studies, the cut-off cardinality is lower than the number of training actions. It means the Action Shapley is more efficiency than the brute-force computation.

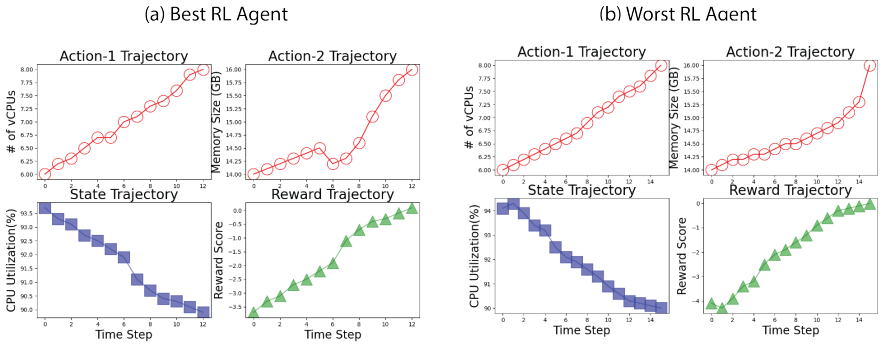

Figure 6: Action, state, and reward trajectories for the best and the worst RL agents for the VM right-sizing case study. (a) The best RL agent is produced by training actions with the highest possible Action Shapley values: $\langle a_2, a_3, a_4, a_5 \rangle$. The resulting cumulative reward is -21.9 and the convergence time is 13. (b) The worst RL agent is produced by training actions with lowest possible Action Shapley values: $\langle a_1, a_3, a_4, a_5 \rangle$. The resulting cumulative reward is -30.4 and the convergence time is 16.

.

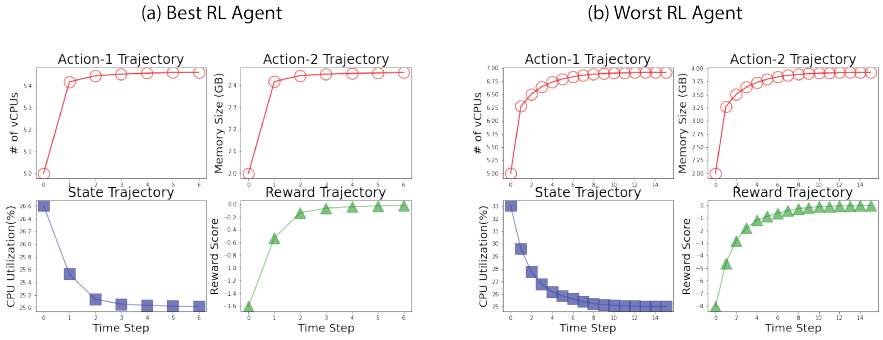

Figure 7: Action, state, and reward trajectories for the best and the worst RL agents for the database tuning case study. (a) The best RL agent is produced by training actions with the highest possible Action Shapley values: $\langle p_1, p_2, p_3, p_4 \rangle$. The resulting cumulative reward is -2.42 and the convergence time is 7. (b) The worst RL agent is produced by training actions with the lowest possible Action Shapley values: $\langle p_1, p_2, p_5, p_6 \rangle$. The resulting cumulative reward is -21 and the convergence time is 16.

.

B Training Data

B.1 VM Right-Sizing

B.2 Load Balancing

B.3 Database Tuning

C Baseline Results

The baseline study assumes the environment model for the RL agent is trained by all training actions. Table 8 shows cumulative rewards and convergence times for four case studies under the baseline condition. It shows cumulative rewards of -37.75, -9.1, -2.33, -561.4 and convergence times of 21, 12, 6, 52, for four different case studies, respectively.

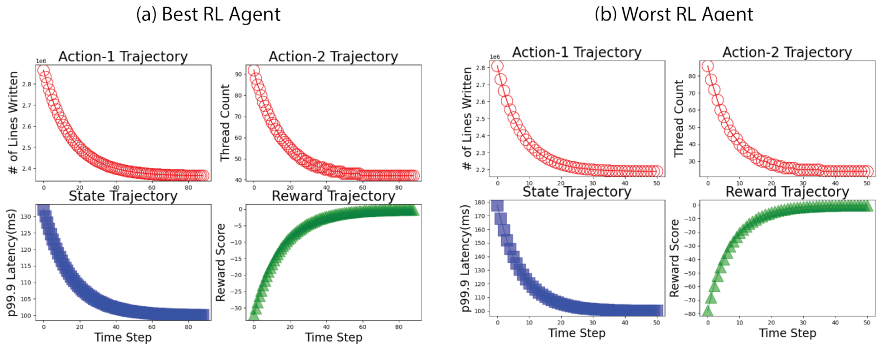

Figure 8: Action, state, and reward trajectories for the best and the worst RL agents for the Kubernetes management case study. (a) The best RL agent is produced by training actions with the highest possible Action Shapley values: $\langle r_2, r_3, r_8, r_9, r_{13} \rangle$. The resulting cumulative reward is -499 and the convergence time is 89. (b) The worst RL is produced by training actions with the lowest possible Action Shapley values: $\langle r_1, r_6, r_{11}, r_{12}, r_{14} \rangle$. The resulting cumulative reward is -621 and the convergence time is 51.

.

Table 7: Global cut-off cardinality values and computation efficiency metrics for different case studies

Case Study	**Global Cut-off**	**No. of Training Actions**	$\mathbf{P_{comp}}$
VM Right-Sizing	4	5	50%
Load Balacing	3	5	75%
Database Tuning	4	6	75%
kubernetes Management	5	15	99.9%

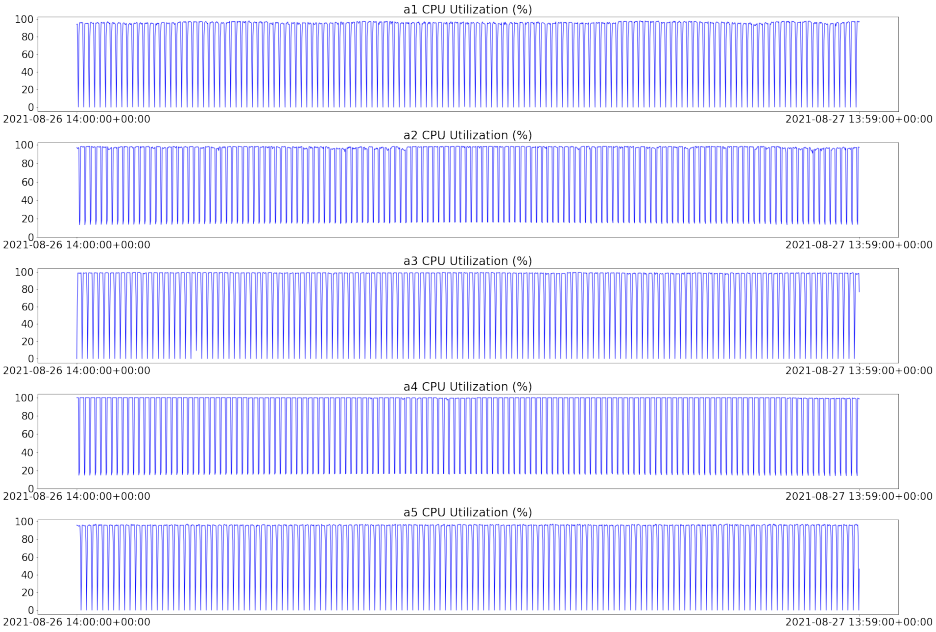

Figure 9: Training Data for VM Right-Sizing

.

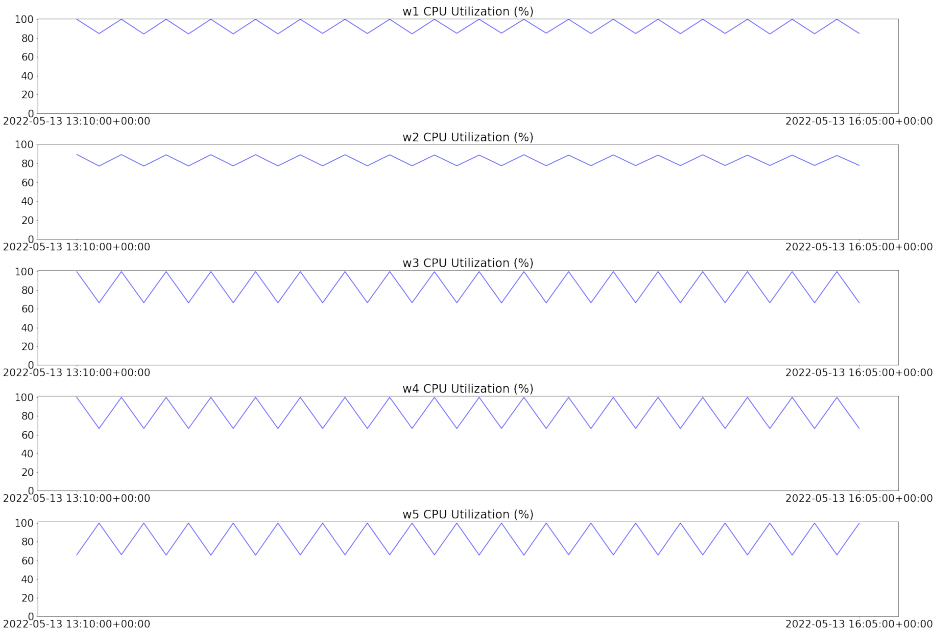

Figure 10: Training Data for Load Balancing

.

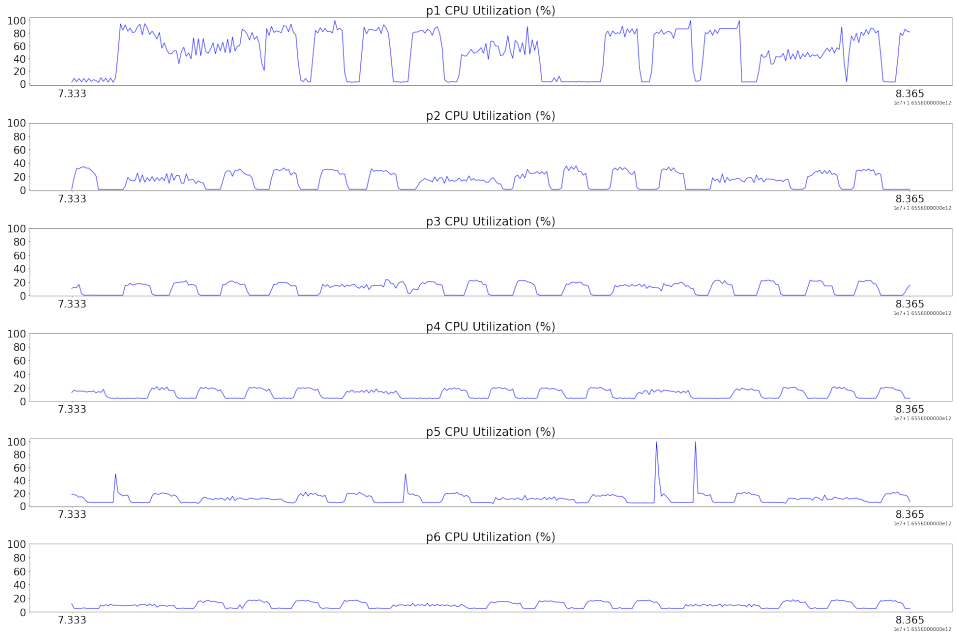

Figure 11: Training Data for Database Tuning

.

Table 8: Results from the Baseline Study

Case Study	Cumulative Reward	Convergence Time
VM Right-Sizing	-37.75	21
Load Balacing	-9.1	12
Database Tuning	-2.33	6
kubernetes Management	-561.4	52