CAE: Repurposing the Critic as an Explorer in Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Exploration remains a fundamental challenge in reinforcement learning, as many existing methods either lack theoretical guarantees or fall short in practical effectiveness. In this paper, we propose CAE, *i.e.*, the Critic as an Explorer, a lightweight approach that repurposes the value networks in standard deep RL algorithms to drive exploration, without introducing additional parameters. CAE leverages multi-armed bandit techniques combined with a tailored scaling strategy, enabling efficient exploration with provable sub-linear regret bounds and strong empirical stability. Remarkably, it is simple to implement, requiring only about 10 lines of code. For complex tasks where learning reliable value networks is difficult, we introduce CAE+, an extension of CAE that incorporates an auxiliary network. CAE+ increases the parameter count by less than 1% while preserving implementation simplicity, adding roughly 10 additional lines of code. Extensive experiments on MuJoCo, MiniHack, and Habitat validate the effectiveness of CAE and CAE+, highlighting their ability to unify theoretical rigor with practical efficiency.

1 Introduction

Exploration in reinforcement learning (RL) remains a fundamental challenge, particularly in environments with complex dynamics or sparse rewards. Although algorithms such as DQN (Mnih et al., 2015), PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018), DDPG (Lillicrap et al., 2016), TD3 (Fujimoto et al., 2018), IMPALA (Espeholt et al., 2018), and DSAC (Duan et al., 2021; 2023) have demonstrated impressive performance on tasks like Atari games (Mnih et al., 2013; 2015), StarCraft (Vinyals et al., 2019), Go (Silver et al., 2017), *etc.*, they often depend on rudimentary exploration strategies. Common approaches, such as ϵ-greedy or injecting noise into actions, are typically inefficient and struggle in scenarios with delayed or sparse rewards.

For decades, exploration with proven optimality in tabular settings has been available (Kearns & Singh, 2002). More recently, methods with provable regret bounds have been developed for scenarios involving function approximation, including linear functions (Osband et al., 2016; 2019; Jin et al., 2018; 2020; Kamyar & Animashree, 2018; Agarwal et al., 2020), kernels (Yang et al., 2020), and neural networks (Yang et al., 2020). However, while linear and kernel-based approaches make strong assumptions about the RL functions, provable methods based on neural networks suffer from prohibitive computational costs, specifically $O(n^3)$, where n is the number of parameters in the RL network. Moreover, some studies (Ash et al., 2022; Ishfaq et al., 2024a) propose algorithms with theoretical guarantees under the linearity assumption and attempt to extend them directly to deep RL without further proof. Other works (Ishfaq et al., 2021; 2024b) provide provable bounds for deep RL, but they are either practically burdensome or rely on unknown sampling errors.

More practical approaches to exploration rely on heuristics, giving rise to several practically successful methods, including Pseudocount (Bellemare et al., 2016), ICM (Pathak et al., 2017), RND (Burda et al., 2019b), RIDE (Raileanu & Rocktäschel, 2020), NovelD (Zhang et al., 2021a), AGAC (Flet-Berliac et al., 2021), and E3B (Henaff et al., 2022; 2023). These approaches typically introduce internally generated bonuses to incentivize exploration of novel states based on predefined metrics. For instance, RND uses the prediction error of a randomly initialized target network as the bonus. Despite their practical success, such

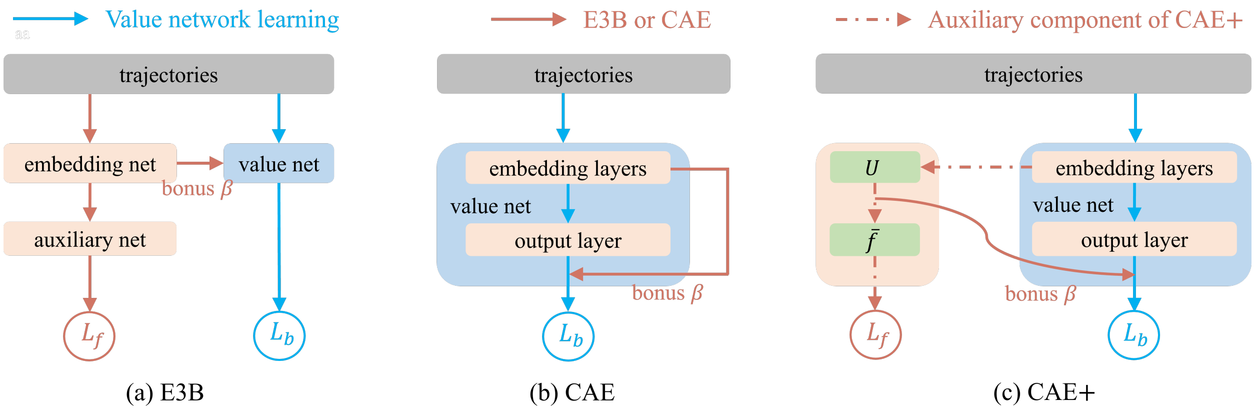


Figure 1: Comparison of existing methods, such as E3B, with CAE and CAE+. L_b represents the Bellman loss to update the value function, while L_f refers to the loss of the auxiliary network. E3B requires additional networks to generate exploration bonuses, while CAE utilizes the embedding layers of the value network for bonuses. CAE+ extends CAE by incorporating a compact auxiliary network $f = \bar{f} \circ \boldsymbol{U}$ to enhance performance in very sparse reward environments, with a minor increase in parameters.

methods often introduce bias into the environment's external reward signal and lack theoretical guarantees. In contrast, potential-based reward shaping methods, such as Liberty (Yiming et al., 2023) and EME (Yiming et al., 2024), offer stronger theoretical underpinnings, but are difficult to implement in practice. Moreover, all of the aforementioned methods typically require training auxiliary networks in addition to the standard value and policy networks in deep RL, leading to significantly increased computational overhead.

In this work, we aim to combine the strengths of both theoretically grounded and empirically effective exploration methods. Provably efficient approaches are fundamentally rooted in the theory of Multi-Armed Bandits (MAB) (Li et al., 2010; Chu et al., 2011; Agrawal & Goyal, 2013; Wen et al., 2015; Zhang et al., 2021b; Zhou et al., 2020). Building on this foundation, we hypothesize that advanced techniques from **neural MAB** can be effectively adapted for exploration in **deep RL**. Recent studies (Zahavy & Mannor, 2019; Riquelme et al., 2018; Xu et al., 2022) suggest that decoupling deep representation learning from exploration strategies holds promise for achieving efficient exploration in neural MAB settings.

Motivated by these insights, we propose CAE. Unlike existing methods that train additional embedding networks to generate exploration bonuses, CAE leverages the embedding layers of the value networks in the RL algorithms and employs MAB techniques to produce exploration bonuses. To ensure the practical stability of CAE, we adopt an appropriate scaling strategy (Welford, 1962; Elsayed et al., 2024) to process the bonuses. Consequently, CAE introduces no additional parameters beyond those in the original algorithms, showcasing that RL algorithms inherently possess strong exploration capabilities if their learned networks are effectively leveraged. Moreover, CAE is simple to implement, requiring only about 10 lines of code. A comparison between CAE and existing methods is in Figure 1.

For tasks with complex dynamics and sparse rewards, learning effective value networks is challenging, impeding exploration based on them. Accordingly, we propose an extended version CAE+, as illustrated in Figure 1. CAE+ integrates a small auxiliary network to facilitate the learning process. The structure of the auxiliary network is carefully designed to prevent severe coupling between the environment dynamics and the returns, thereby further enhancing the performance of CAE+. Remarkably, this addition increases the parameter count by less than 1% and requires only about 10 extra lines of code, thus preserving the simplicity and lightweight nature.

Our experiments cover a diverse set of benchmarks, *i.e.*, MuJoCo, MiniHack, and Habitat, representing dense-reward, sparse-reward, and reward-free environments. CAE improves the performance of state-of-the-art baselines such as PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018), TD3 (Fujimoto et al., 2018), and DSAC (Duan et al., 2021; 2023). Additionally, CAE+ demonstrates robust performance, consistently outperforming E3B Henaff et al. (2022; 2023), the state-of-the-art exploration method for MiniHack and

Table 1: Method comparison. **Linear function** indicates whether the theoretical guarantees apply to RL algorithms with linear value function approximation. **Network-based function** specifies whether the proof extends to RL algorithms using network-based function approximation. **Toy tasks** denotes whether the method can be effectively applied to very simple deep RL tasks. **Practical complex tasks** assesses whether the method remains computationally feasible when deployed in realistic, large-scale deep RL settings.

Method	Provable		Empirical with Deep RL	
	Linear function	Network-based function	Toy tasks	Practical complex tasks
LSVI-UCB (Jin et al., 2020)	✓	✗	✗	✗
NN-UCB (Yang et al., 2020)	✓	✓	✗	✗
OPT-RLSVI (Andrea et al., 2020)	✓	✗	✗	✗
LSVI-PHE (Ishfaq et al., 2021)	✓	✓	✓	✗
BDQN (Kamyar & Animashree, 2018)	✓	✗	✓	✓
ACB (Ash et al., 2022)	✓	✗	✓	✓
LMCDQN (Ishfaq et al., 2024a)	✓	✗	✓	✓
CAE	✓	✓	✓	✓
CAE+	✓	✓	✓	✓

Habitat, across all evaluated tasks. These results highlight the superior reliability and effectiveness of CAE and CAE+ in diverse RL scenarios.

In summary, we make three key contributions. First, we propose lightweight CAE and CAE+, which enable the use of linear MAB techniques for exploration in deep RL. By adopting a scaling strategy and carefully designing the small auxiliary network, we ensure both practical stability and functionality in environments with dense and sparse rewards. Second, our theoretical analysis demonstrates that any deep RL algorithm with CAE or CAE+ achieves a sub-linear regret bound over episodes. Finally, experiments on MuJoCo, MiniHack, and Habitat validate the effectiveness of CAE and CAE+, showcasing their superior performance.

2 Related Work

Multi-armed bandits. MAB algorithms address the exploration-exploitation dilemma by making sequential decisions under uncertainty. LinUCB (Li et al., 2010) assumes a linear relationship between arm contexts and rewards, enabling efficient exploration via uncertainty quantification in the estimated parameter space and ensuring a sub-linear regret bound (Chu et al., 2011). To relax the linearity assumption, KernelUCB (Valko et al., 2013; Chowdhury & Gopalan, 2017) and NegUCB (Li et al., 2024) transform contexts into high-dimensional spaces and apply LinUCB to the mapped contexts. Neural-UCB (Zhou et al., 2020) and Neural-TS (Zhang et al., 2021b) leverage neural networks to model the complex relationships between contexts and rewards. However, their computational complexity of $O(n^3)$, where n denotes the number of network parameters, limits their scalability in real-world applications. Neural-LinTS (Riquelme et al., 2018) and Neural-LinUCB (Xu et al., 2022) mitigate this limitation by decoupling representation learning from exploration, improving the practicality of neural MAB.

Provable exploration in RL. Provably efficient exploration methods (Kearns & Singh, 2002; Osband et al., 2016; 2019; Jin et al., 2018; 2020; Agarwal et al., 2020; Cai et al., 2020; Daniil et al., 2022; 2023) often face empirical limitations or are primarily theoretical, lacking applicability in deep RL. Some studies (Alessio & Filippo, 2024; Kamyar & Animashree, 2018; Ash et al., 2022; Ishfaq et al., 2024a) propose methods with theoretical guarantees under a tabular or linearity assumption and directly extend them to deep RL settings. Other works (Ishfaq et al., 2021; 2024b) offer provable bounds for deep RL, while they are either practically burdensome or rely on unknown sampling errors. A comparative analysis of representative provably efficient exploration methods is in Table 1.

Practical exploration in deep RL. Practically successful methods (Bellemare et al., 2016; Pathak et al., 2017; Burda et al., 2019a; Raileanu & Rocktäschel, 2020; Burda et al., 2019b; Zhang et al., 2021a; Flet-Berliac et al., 2021; Henaff et al., 2022; 2023; Jarrett et al., 2023; Yuan et al., 2023) typically employ exploration bonuses to encourage agents to visit novel states. However, these approaches often lack rigorous theoretical foundations. In contrast, methods inspired by potential-based reward shaping (Andrew et al., 1999), such as Liberty (Yiming et al., 2023) and EME (Yiming et al., 2024), offer stronger theoretical grounding, but are challenging to implement in practice. Moreover, both classes of methods generally require training a substantial number of additional parameters. In comparison, CAE and CAE+ utilize MAB techniques, assisted by embedding layers within the RL value networks, providing empirical benefits with minimal additional parameters. Figure 1 illustrates the differences between various exploration methods, while Table 2 summarizes their additional networks and parameters.

Table 2: Comparison of exploration methods on MiniHack. **Networks**: additional networks beyond those in the base RL algorithm IMPALA (Espeholt et al., 2018), which contains $25,466,652$ parameters; **# Params**: the number of additional parameters introduced by the exploration method. Networks in **bold** represent those with significant parameters, while those in gray indicate substantially fewer parameters.

Method	Networks	# Params	Params ↑
ICM (Pathak et al., 2017)	**Embedding** + Forward dynamics + Inverse dynamics	$16,074,512$ $+2,110,464+527,371$	73%
RND (Burda et al., 2019b)	**Embedding**	$16,074,512$	63%
RIDE (Burda et al., 2019b)	**Embedding** + Forward dynamics + Inverse dynamics	$16,074,512$ $+2,110,464+527,371$	73%
NovelD (Zhang et al., 2021a)	**Embedding**	$16,074,512$	63%
E3B (Henaff et al., 2022; 2023)	**Embedding** + Inverse dynamics	$16,074,512$ $+527,371$	65%
CAE	-	-	0
CAE+	Inverse dynamics	$199,819$	0.8%

3 Methodology

Unless otherwise specified, bold uppercase symbols denote matrices, while bold lowercase symbols represent vectors. $\boldsymbol{I}$ refers to an identity matrix. Frobenius norm and l_2 norm are denoted by $\|\cdot\|_2$. Mahalanobis norm of $\boldsymbol{x}$ based on $\boldsymbol{A}$ is $\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}$. For integer $K > 0$, the set of integers $\{1, 2, ..., K\}$ is denoted by $[K]$.

3.1 Preliminary

An episodic Markov Decision Process is formally defined as a tuple $(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, where $\mathcal{S}$ denotes the state space and $\mathcal{A}$ is the action space. Integer $H > 0$ indicates the duration of each episode. Functions $\mathbb{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ are the Markov transition and reward functions, respectively. During an episode, the agent follows a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. At each time step $h \in [H]$ in the episode, the agent observes the current state $s_h \in \mathcal{S}$ and selects an action $a_h \sim \pi(\cdot|s_h)$ to execute, then the environment transits to the next state $s_{h+1} \sim \mathbb{P}(\cdot|s_h, a_h)$, yielding an immediate reward $r_h = r(s_h, a_h)$. At time step h, the action-value $Q(s_h, a_h)$ approximates the accumulative return obtained by taking action a_h at state s_h and subsequently following policy π:

$$Q(s_h, a_h) \approx \sum_{t=h}^{H} \gamma^{t-h} r_t \tag{1}$$

where $0 \leq \gamma \leq 1$ is the discount parameter.

Many algorithms have been developed to learn the optimal policy π^* for the agent to select and execute actions at each time step h in the episode, thus ultimately maximizing the accumulative return $\sum_{h=1}^{H} \gamma^{h-1} r_h$. Notable algorithms include DQN (Mnih et al., 2015), PPO (Schulman et al., 2017), SAC (Haarnoja et al., 2018), TD3 (Fujimoto et al., 2018), IMPALA (Espeholt et al., 2018), DSAC (Duan et al., 2021; 2023), and others. A common component of these algorithms is using a network to approximate the action-value function[1] Q under a specific policy as Equation 2, where $\phi(\cdot, \cdot|\boldsymbol{W})$ is the embedding layers, $\boldsymbol{\theta}$ and $\boldsymbol{W}$ are trainable parameters of the network.

[1] In some algorithms, the state-value function, rather than the action-value function, is learned. However, this does not affect the implementation and conclusion of our method, as will be seen in subsection 3.2.

$$Q(s,a) = \boldsymbol{\theta}^{\mathsf{T}} \phi(s,a|\boldsymbol{W}) \tag{2}$$

Bellman loss as Equation 3 is often employed to update the action-value function. Using the most recent action-value function, the policy can be updated in various ways, depending on the specific algorithm. Since CAE focuses on leveraging Equation 2 for efficient exploration while preserving the core techniques of existing RL algorithms, we introduce CAE within the context of DQN for simplicity. However, it can be easily adapted to other RL algorithms.

$$L_B = \left(Q(s_h, a_h) - \mathbb{E}_{s_{h+1} \sim \mathbb{P}(\cdot|s_h, a_h)} \left[r_h + \gamma \cdot \max_{a_{h+1}} Q(s_{h+1}, a_{h+1}) \right] \right)^2 \tag{3}$$

3.2 CAE: the Critic as an Explorer

For a state-action pair (s,a), the approximated action-value $Q(s,a)$ is subject to an uncertainty term $\beta(s,a)$, arising from the novelty or limited experience with the particular state-action pair. Similar to MAB problems, it is essential to account for this uncertainty when utilizing the latest approximated action-value function. Incorporating the uncertainty term encourages exploration, ultimately improving long-term performance. Consequently, the action-value function adjusted for uncertainty is as Equation 4, where $\alpha \geq 0$ is the exploration coefficient. Notably, in some literature, *uncertainty* is also referred to as a *bonus*, and we use these terms interchangeably when unambiguous.

$$Q(s,a) = \boldsymbol{\theta}^{\mathsf{T}} \phi(s,a|\boldsymbol{W}) + \alpha\beta(s,a) \tag{4}$$

However, defining $\beta(s,a)$ remains challenging. Provably efficient methods often attempt to address this by either assuming a linear value function (Jin et al., 2018) or requiring $O(n^3)$ computation time (Yang et al., 2020) in terms of the number of parameters n in the value network. Both of these approaches have drawbacks, *i.e.*, linearity fails to capture the complexity of tasks while the $O(n^3)$ computational cost is impractical.

To overcome these limitations, we draw inspiration from Neural-LinUCB (Xu et al., 2022) and Neural-LinTS (Riquelme et al., 2018), which effectively decouple representation learning from exploration strategies. Building on this idea and following the standard value network structure in Equation 2, CAE decomposes the action-value function into two distinct components.

- Network $\phi(s,a|\boldsymbol{W})$ extracts the embedding of the state-action pair (s,a);
- $Q(s,a) = \boldsymbol{\theta}^{\mathsf{T}} \phi(s,a|\boldsymbol{W})$ is a linear function of the embedding $\phi(s,a|\boldsymbol{W})$ with parameter $\boldsymbol{\theta}$.

Consequently, after appropriate modifications, MAB theory under the linearity assumption can be adapted to work with the embeddings $\phi(s,a)$ for $\forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}$. Simultaneously, the action-value function retains its representational capacity through the embedding layers $\phi(s,a)$, ensuring promising practical performance. While various MAB techniques can be adapted to the embeddings, we illustrate CAE using the two most representative ones. Other techniques can be utilized similarly, showcasing that CAE is a generalizable framework rather than a fixed method.

Upper Confidence Bound (UCB) is an optimistic exploration strategy in MAB. It defines the uncertainty term as Equation 5, where $\boldsymbol{A}$ denotes the Gram matrix, initialized as $\boldsymbol{A} = \lambda \boldsymbol{I}$ with λ being the ridge regularization parameter. After each time step executing action a under state s, $\boldsymbol{A}$ is updated according to Equation 6, where $\phi(\cdot,\cdot)$ represents the latest embedding layers.

$$\beta(s,a) = \sqrt{\phi(s,a)^{\mathsf{T}} \boldsymbol{A}^{-1} \phi(s,a)} \tag{5}$$

$$\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{\phi}(s,a)\boldsymbol{\phi}^{\mathsf{T}}(s,a) \tag{6}$$

Thompson Sampling is a randomized exploration strategy that samples the value function, adjusted for uncertainty, from a posterior distribution. It defines the uncertainty term as Equation 7, where the Gram matrix $\boldsymbol{A}$ is initialized and updated in the same manner as in UCB.

$$\boldsymbol{\Delta\theta} \sim N(0, \boldsymbol{A}^{-1}); \quad \beta(s,a) = (\boldsymbol{\Delta\theta})^{\mathsf{T}}\phi(s,a) \tag{7}$$

As the value network undergoes continuous updates, exploration based on the ever-changing embedding layers $\phi(\cdot,\cdot)$ can become highly unstable, significantly impairing practical performance. Inspired by existing scaling strategies (Welford, 1962; Elsayed et al., 2024), we adopt an appropriate one for the generated uncertainty at each time step, ensuring both stability and practical functionality, as detailed in Algorithm 1. This scaling strategy normalizes the generated uncertainty at each time step using the running standard deviation, which, despite its simplicity, has a profound impact on the performance of CAE. The critical importance of this design is further highlighted through ablation studies presented in Appendix E.

Adapting CAE to General RL Algorithms Depending on the RL algorithm employed, we may sometimes learn the state- instead of the action-value network. As a result, the value network can only derive state embeddings rather than state-action pair embeddings. Even when learning the action-value network, it may still output only state embeddings if it is designed to take states as input and produce values for multiple actions. In such cases, the embedding of the next state is utilized as a proxy for the current state-action embedding when computing uncertainty.

Algorithm 1 Scaling strategy for the uncertainty

```
Input: Uncertainty $b$, running mean $\mu$, running variance $\nu^2$, and running count of samples $\mathcal{N}$
Update the sample count $\mathcal{N} \leftarrow \mathcal{N} + 1$
Compute $\delta = b - \mu$
Update the running mean $\mu \leftarrow \mu + \frac{\delta}{\mathcal{N}}$
Update the running variance $\nu^2 \leftarrow \nu^2 + \frac{\delta \times (b-\mu)}{\mathcal{N}}$
Output: Scaled uncertainty $\frac{b}{\nu}$, and updated $\mu$, $\nu^2$, and $\mathcal{N}$
```

3.3 CAE+: Enhancing CAE with Minimal Overhead

For tasks with complex dynamics or very sparse rewards, learning high-quality value networks is particularly challenging, which in turn hinders exploration reliant on them. Accordingly, we propose CAE+, an extension of CAE that incorporates a lightweight auxiliary network, introducing less than 1% additional parameters.

Specifically, we utilize the Inverse Dynamics Network (IDN) (Pathak et al., 2017; Raileanu & Rocktäschel, 2020; Henaff et al., 2022) to enhance the learning of the embedding layers contained in the value network. This is achieved by a compact network f that infers the distribution $p(a_h)$ over taken actions given consecutive states s_h and s_{h+1}, which is trained by maximum likelihood estimation as Equation 8.

$$L_f = -\log p(a_h|s_h, s_{h+1}) \tag{8}$$

To introduce minimal additional parameters, we utilize the embedding layers as follows:

- If the value network is an action-value network with embedding layers $\phi(s,a)$, a constant default value $\ddot{a}$ is assigned to the action input, while the actual states are used; then the resulting outputs are treated as state embeddings.
- If the value network is a state-value network with embedding layers $\phi(s)$, the actual states are directly fed in to generate their embeddings.

These state embeddings are subsequently transformed by a linear layer $\boldsymbol{U}$, followed by a small network $\bar{f}$, which processes the transformed consecutive embeddings to infer the action. Equation 9 illustrates this procedure for the case of embedding layers $\phi(s,a)$.

$$p(a_h|s_h, s_{h+1}) = f(\phi(s_h, \ddot{a}), \phi(s_{h+1}, \ddot{a})) = \bar{f}(\boldsymbol{U}\phi(s_h, \ddot{a}), \boldsymbol{U}\phi(s_{h+1}, \ddot{a})) \tag{9}$$

Although incorporating the IDN loss can accelerate the learning of the embedding layers by leveraging knowledge of environment dynamics, the resulting strong coupling with the dynamics restricts the flexibility of exploration. In CAE+, to address this limitation, we adopt the transformerd embeddings $\boldsymbol{U}\phi(s, a)$ instead of the original ones $\phi(s, a)$ to calculate the uncertainty. Consequently, by replacing $\phi(s, a)$ with $\boldsymbol{U}\phi(s, a)$, Equations 5 and 7, which are used in CAE to generate uncertainty, are modified into Equation 10, with the Gram matrix $\boldsymbol{A}$ updated according to Equation 11. Complete procedure of CAE+ is in Algorithm 2.

$$\beta(s, a) \triangleq \begin{cases} \sqrt{\phi^\top(s, a)\boldsymbol{U}^\top \boldsymbol{A}^{-1}\boldsymbol{U}\phi(s, a)} & \text{UCB} \\ (\boldsymbol{\Delta\theta})^\top \boldsymbol{U}\phi(s, a) \mid_{\boldsymbol{\Delta\theta} \sim \mathcal{N}(0, \boldsymbol{A}^{-1})} & \text{Thompson Sampling} \end{cases} \tag{10}$$

$$\boldsymbol{A} \leftarrow \boldsymbol{A} + \boldsymbol{U}\phi(s, a)\phi^\top(s, a)\boldsymbol{U}^\top \tag{11}$$

In CAE+, the structure of the network f offers several advantages. First, the transformation $\boldsymbol{U}$ decouples environment dynamics from returns, mitigating interdependencies that could hinder flexibility and thereby enhancing empirical performance, as evidenced by the ablation studies in section 5. Second, since $\boldsymbol{U}$ is a simple linear transformation, it preserves the theoretical guarantees of UCB- and Thompson Sampling-based exploration strategies, ensuring both rigor and stability in practice. Third, by projecting $\phi(s, a)$ into a lower-dimensional embedding with $\bar{d} < d$, the approach not only reduces the number of additional parameters but also lowers the computational complexity of uncertainty estimation at each time step, *i.e.*, from $O(d^3)$ to $O(\bar{d}^3)$, making the method more efficient.

Speed Up CAE+ with Rank-1 Update According to Algorithm 2, the Gram matrix $\boldsymbol{A}$ needs to be inverted at each step, which is cubic in dimension. *Alternatively*, we can use the Sherman-Morrison matrix identity (Sherman & Morrison, 1950; Henaff et al., 2022) to perform rank-1 updates of $\boldsymbol{A}^{-1}$ in quadratic time as Equation 12.

$$\boldsymbol{A}^{-1} \leftarrow \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{U}\boldsymbol{\phi}(s, a)\boldsymbol{\phi}^\top(s, a)\boldsymbol{U}^\top(\boldsymbol{A}^{-1})^\top}{1 + \boldsymbol{\phi}^\top(s, a)\boldsymbol{U}^\top\boldsymbol{A}^{-1}\boldsymbol{U}\boldsymbol{\phi}(s, a)} \tag{12}$$

4 Theoretical Analysis

Under the optimal policy π^*, assume the corresponding action-value function Q^* is structured as in Equation 2 and parameterized by $\boldsymbol{\theta}^*$ and $\boldsymbol{W}^*$. In Algorithm 2, the policy executed in episode $m \in [M]$ is π_m, with its action-value function denoted as Q^{π_m}. Cumulative regret of Algorithm 2 is as Definition 1.

Definition 1. ***Cumulative Regret.*** *After M episodes of interactions with the environment, the cumulative regret of CAE or CAE+ is defined as Equation 13, where u_1^m is the optimal action at state s_1^m generated by policy π^* while a_1^m is that selected by the executed policy π_m.*

$$\mathrm{R}_M = \sum_{m=1}^{M} Q^*(s_1^m, u_1^m) - Q^{\pi_m}(s_1^m, a_1^m) \tag{13}$$

Cumulative regret measures the gap between the optimal return and the actual return accumulated over M episodes. As discussed earlier, CAE draws inspiration from Neural-LinUCB (Xu et al., 2022) and Neural-LinTS (Riquelme et al., 2018). While Neural-LinUCB is supported by theoretical guarantees, Neural-LinTS has so far only been validated empirically. In this work, we complete the regret analysis for Neural-LinTS and subsequently derive the regret bound of CAE. Before stating Theorem 1, we introduce the standard

Algorithm 2 CAE+ with action-value network

```
Input: Ridge parameter $\lambda > 0$, exploration parameter $\alpha \geq 0$, episode length $H$, episode number $M$
Initialize: Gram matrix $\boldsymbol{A} = \lambda \boldsymbol{I}$, initial policy $\pi(\cdot)$ and value function $Q(\cdot,\cdot)$, network $f(\cdot|\boldsymbol{U})$, running mean $\mu = 0$, running variance $\nu^2 = 0$, running count $\mathcal{N} = 0$
for episode $m = 1$ to $M$ do
    Receive the initial state $s_1^m$ from the environment
    for step $h = 1, 2, ..., H$ do
        Conduct action $a_h^m \sim \pi(s_h^m)$ and observe the next state $s_{h+1}^m$ and receive immediate reward $r_h^m$
        Generate bonus $\beta(s_h^m, a_h^m)$ by Equation 10
        Provide $\beta(s_h^m, a_h^m)$, $\mu$, $\nu^2$, and $\mathcal{N}$ as inputs to Algorithm 1 to obtain scaled bonus $\beta_h^m$ and updated $\mu$, $\nu^2$, and $\mathcal{N}$
        Reshape the reward $\tilde{r}_h^m = r_h^m + \alpha\beta_h^m$
        Update the Gram matrix $\boldsymbol{A}$ by Equation 11
    end for
    Sample a batch $\mathcal{B} = \{s_h, a_h, s_{h+1}, \tilde{r}_h\}, h \in [1, H-1]$
    Calculate the IDN loss $L_f$ by Equation 8 and the Bellman loss $L_B$ by Equation 3 on batch $\mathcal{B}$
    Update value function $Q(\cdot,\cdot)$ and network $f$ jointly by minimizing the combined loss $\min(L_f + L_B)$
    Update the policy $\pi(\cdot)$ based on the latest value function $Q(\cdot,\cdot)$
end for
```

assumptions in the literature of *deep representation and shallow exploration* (Xu et al., 2022) in Neural MAB as Assumption 1, Assumption 2, and Assumption 3.

Assumption 1. *Assume that $\|(s;a)\|_2 = 1$ for $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$, and that the entries of $(s;a)$ satisfy Equation 14, where D represents the dimension of $(s;a)$.*

$$(s;a)_j = (s;a)_{j+\frac{D}{2}} \tag{14}$$

Notably, even if the original state-action pairs $(s;a)$ do not satisfy this assumption, they can be preprocessed by transforming them into $(s;a;s;a)$ and applying an appropriate scaling to ensure the assumption holds.

Assumption 2. *For $\forall s_1, s_2 \in \mathcal{S}$ and $\forall a_1, a_2 \in \mathcal{A}$, there is a constant $l_{Lip} > 0$, such that:*

$$\left\|\nabla_{\boldsymbol{W}}\phi(s_1, a_1|\boldsymbol{W}_h^1) - \nabla_{\boldsymbol{W}}\phi(s_2, a_2|\boldsymbol{W}_h^1)\right\|_2 \leq l_{Lip} \left\|(s_1;a_1) - (s_2;a_2)\right\|_2 \tag{15}$$

Assumption 3. *The neural tangent kernel $\boldsymbol{H}$ of the action-value network is positive definite.*

Neural tangent kernel $\boldsymbol{H}$ is defined in accordance with a recent line of research (Jacot et al., 2018; Arora et al., 2019; Xu et al., 2022) and is essential for the analysis of overparameterized neural networks. A detailed discussion of these assumptions is deferred to Appendix D.2, where we show that they are mild and commonly adopted in the literature. In the following, we present the regret guarantee of CAE.

Theorem 1. *Suppose Assumption 1, Assumption 2, and Assumption 3 hold, $\|\boldsymbol{\theta}^*\|_2 \leq 1$ and $\|(s_h;a_h)\|_2 \leq 1$. For any $\sigma \in (0,1)$, assume the number of parameters ι in each of the L layers of $\phi(\cdot,\cdot)$ satisfies $\iota = \mathrm{poly}(L, d, \frac{1}{\sigma}, \log\frac{M|\mathcal{A}|}{\sigma})$, where $|\mathcal{A}|$ means the action space size and $\mathrm{poly}(\cdot)$ means a polynomial function depending on the incorporated variables. Let:*

$$\alpha = \sqrt{2(d \cdot \log(1 + \frac{M \cdot \log|\mathcal{A}|}{\lambda}) - \log\sigma)} + \sqrt{\lambda}$$
$$\eta \leq C_1(\iota \cdot d^2 M^{\frac{11}{2}} L^6 \cdot \log\frac{M\,|\mathcal{A}|}{\sigma})^{-1}$$

then with probability at least $1 - \sigma$, it holds that:

$$\mathrm{R}_M \leq C_2\alpha H\sqrt{Md\log(1 + \frac{M}{\lambda d})} + C_4 H\sqrt{MH\log\frac{2}{\sigma}} + \frac{C_3 H L^3 d^{\frac{5}{2}} M\sqrt{\log(\iota + \frac{1}{\sigma} + \frac{M|\mathcal{A}|}{\sigma})}\,\|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{\boldsymbol{H}^{-1}}}{\iota^{\frac{1}{6}}}$$

where C_1, C_2, C_3, C_4 are constants; $\boldsymbol{q}$ and $\tilde{\boldsymbol{q}}$ are the target value vector and the estimated value vector of the action-value network, respectively. More discussions of these notations are in Appendix B and Appendix D.

Specifically, we assume $\|\boldsymbol{\theta}^*\|_2 \leq 1$ and $\|(s_h; a_h)\|_2 \leq 1$ to make the bound scale-free. From this theorem, we can conclude that the upper bound of the cumulative regret grows sub-linearly with the number of episodes M, *i.e.*, $\widetilde{O}(\sqrt{M})$ where $\widetilde{O}(\cdot)$ hide constant and logarithmic dependence of M, indicating that the executed policy improves over time. Since MAB techniques are applied to the linear layer on top of the embedding layers, the cumulative regret naturally consists of two components, i.e., the exploration regret from the linear layer, and the error induced by the network's estimation, which appears as the last term in the regret bound. It involves a trade-off between M and ι. Moreover, as the estimation error $|\boldsymbol{q} - \tilde{\boldsymbol{q}}|_{\boldsymbol{H}^{-1}}$ decreases over time, this term typically becomes negligible.

5 Experiment

Benchmarks. In our experiments, we evaluate CAE and CAE+ on MuJoCo, MiniHack, and Habitat, which are characterized by dense rewards, sparse rewards, and reward-free setting, respectively.

Baselines. In our experiments, we evaluate our approach against **nine** baselines, *i.e.*, SAC (Haarnoja et al., 2018), PPO (Schulman et al., 2017), TD3 (Fujimoto et al., 2018), DSAC (Duan et al., 2021; 2023), ICM (Pathak et al., 2017), RND (Burda et al., 2019b), RIDE (Raileanu & Rocktäschel, 2020), NovelD (Zhang et al., 2021a), and E3B (Henaff et al., 2022; 2023).

For **MuJoCo** tasks, we evaluate SAC, PPO, TD3, and DSAC, both with and without CAE. Notably, the other baselines are excluded from the MuJoCo experiments, as they are rarely applied to dense reward settings. Moreover, a fair comparison is infeasible due to the lack of publicly available implementations. Since CAE introduces no additional parameters, it is meaningful to assess whether it can improve existing RL algorithms without increasing training overhead.

For **MiniHack and Habitat** tasks, we adopt IMPALA (Espeholt et al., 2018) and PPO as the base RL algorithms, respectively, following the standard configurations in the open-source codebases. We compare CAE and CAE+ against ICM, RND, RIDE, NovelD, and E3B. Since E3B achieves state-of-the-art performance on both MiniHack and Habitat, we report only the results of E3B. For results of ICM, RND, RIDE, and NovelD, refer to the E3B paper (Henaff et al., 2022; 2023).

All the experiments are based on open-source codebases from E3B, CleanRL (Huang et al., 2022), DSAC, and Habitat-lab (Savva et al., 2019; Andrew et al., 2021; Xavi et al., 2023). Core code and hyperparameters are provided in Appendix A and E, respectively. Experiments were conducted on an Ubuntu 22.04 LTS system equipped with a 13th Gen Intel Core i9-13900KF CPU and an NVIDIA RTX 4090 GPU.

5.1 MuJoCo tasks with Dense Rewards

Table 3: Experimental results after $1e6$ interaction steps on MuJoCo-v4 tasks, except for the *Humanoid* task, whose results are evaluated after $4e6$ interaction steps. *RPI* represents the **R**elative **P**erformance **I**mprovement achieved by CAE. Refer to Appendix E for experimental figures.

Alg. / Env	PPO	PPO + CAE	*RPI %*	TD3	TD3 + CAE	*RPI %*	SAC	SAC + CAE	*RPI %*
Swimmer	99 ± 11.5	$\mathbf{107 \pm 6.47}$	8.08	78 ± 15.4	$\mathbf{130 \pm 14.2}$	66.7	61 ± 35.2	$\mathbf{161 \pm 26.9}$	164
Hopper	$\mathbf{2503 \pm 786.6}$	2453 ± 673.2	−2.00	3044 ± 574.0	$\mathbf{3244 \pm 226.3}$	6.57	2908 ± 600.8	$\mathbf{3188 \pm 485.2}$	9.63
Walker2d	3405 ± 842.0	$\mathbf{3554 \pm 928.0}$	4.38	3764 ± 234.4	$\mathbf{4251 \pm 567.1}$	12.9	4362 ± 405.5	$\mathbf{4742 \pm 484.4}$	8.71
Ant	1762 ± 540.0	$\mathbf{2378 \pm 843.4}$	35.0	3492 ± 1745.7	$\mathbf{5074 \pm 519.3}$	45.3	4846 ± 1306.4	$\mathbf{5482 \pm 511.9}$	13.1
HalfCheetah	2636 ± 1344.3	$\mathbf{3104 \pm 926.0}$	17.8	10316 ± 193.8	$\mathbf{10473 \pm 563.4}$	1.52	11154 ± 457.1	$\mathbf{11587 \pm 418.3}$	3.88
Humanoid	619 ± 92.1	$\mathbf{646 \pm 127.1}$	4.36	5973 ± 257.7	$\mathbf{6275 \pm 483.2}$	5.06	$\mathbf{5261 \pm 186.4}$	5218 ± 228.4	−0.82
RPI Mean	-	-	11.3	-	-	23.0	-	-	33.1

MuJoCo testbed is a widely used physics-based simulation environment. MuJoCo provides a suite of continuous control tasks where agents must learn to perform various actions, such as locomotion, manipulation, and balancing, within simulated robotic environments. Since comparisons among state-of-the-art RL baselines, such as PPO, SAC, TD3, and DSAC on MuJoCo, have been extensively covered in previous studies, our focus is on investigating how CAE can enhance these algorithms.

Results are summarized in Table 3, using seeds $\{1, 2, 3, 4, 5\}$, demonstrating that CAE consistently improves the performance of PPO, TD3, and SAC across most MuJoCo tasks. Notably, TD3 and SAC, when enhanced with CAE, show significantly better performance on the *Swimmer* task. While this task is not typically regarded as particularly challenging, standalone implementations of TD3 and SAC have achieved limited performance.

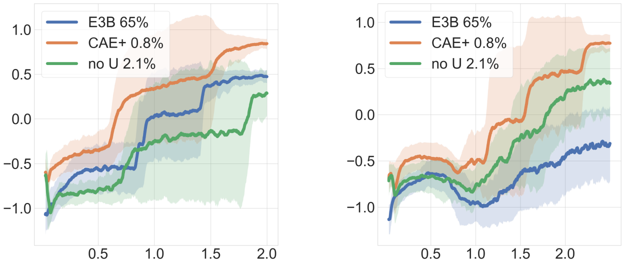
In Table 4, we summarize the performance of DSAC with and without CAE. It is important to note that these experiments are conducted on MuJoCo-v3 rather than MuJoCo-v4 solely because the open-source DSAC codebase is based on MuJoCo-v3. As shown, incorporating CAE consistently improves the performance of DSAC on the MuJoCo benchmark.

Table 4: Experimental results after $1e6$ steps on MuJoCo-v3, except for the *Humanoid* task, which is evaluated after $2e6$ steps.

Alg. Env	DSAC	DSAC + CAE	*RPI %*
Swimmer	131 ± 14.8	$\mathbf{150 \pm 7.96}$	14.5
Hopper	2417 ± 541.6	$\mathbf{2845 \pm 594.5}$	17.7
Walker2d	5550 ± 624.0	$\mathbf{6069 \pm 422.1}$	9.35
Ant	5912 ± 809.7	$\mathbf{6305 \pm 322.7}$	6.65
HalfCheetah	16036 ± 439.1	$\mathbf{16338 \pm 249.1}$	1.88
Humanoid	10059 ± 996.1	$\mathbf{10333 \pm 1104.4}$	2.72
RPI Mean	-	-	8.80

5.2 MiniHack tasks with Sparse Rewards

MiniHack (Samvelyan et al., 2021) is built on the NetHack Learning Environment (Küttler et al., 2020), a challenging video game where an agent navigates procedurally generated dungeons to retrieve a magical amulet. MiniHack tasks present a diverse set of challenges, such as locating and utilizing magical objects, traversing hazardous environments like lava, and monsters. These tasks are characterized by sparse rewards, and the state provides a wealth of information, including images, texts, and more, though only a subset is relevant to each specific task.



(a) Horn-Restricted

(b) N6-Locked

Figure 2: Ablation study to $\boldsymbol{U}$ on MiniHack. Horizontal axis denotes the steps in multiples of $1e7$.

As shown in Table 2, CAE+ introduces only a 0.8% increase in parameters compared to the base RL algorithm, IMPALA. In contrast, the other exploration baselines, such as RIDE and E3B, require $60\% - 80\%$ additional parameters, underscoring the lightweight design of CAE+. Experimental results for E3B and CAE+, using seeds $\{1, 2, 3\}$, are summarized in Table 5, where their performance is evaluated across nine representative tasks, *i.e.*, five *navigation* tasks and four *skill* tasks. The results demonstrate that CAE+ consistently outperforms E3B. Notably, on challenging tasks such as *N6-Locked* and *LavaCross*, CAE+ achieves substantial performance improvements of 348% and 282%, respectively, highlighting its strong exploration capabilities.

Ablation study to CAE on MiniHack. Results of CAE on MiniHack tasks are provided in Table 5. As shown, CAE successfully solves a subset of tasks and even surpasses CAE+ in certain cases. However, it struggles to achieve positive performance in others, such as *N6-Locked*, *etc.*, which pose significant exploration challenges. This limitation stems from the difficulty of training effective value networks in complex environments, adversely affecting exploration reliant on them.

Ablation study to the transformation $\boldsymbol{U}$. Additionally, we present experimental results for CAE+ without the transformation matrix $\boldsymbol{U}$ in the auxiliary network f. As shown in Figure 2, CAE+ without $\boldsymbol{U}$ occasionally outperforms E3B, though there are instances where it does not. Importantly, it consistently underperforms compared to the full CAE+ with $\boldsymbol{U}$. Moreover, CAE+ introduces more additional parameters without matrix $\boldsymbol{U}$, specifically 2.1%.

Wall-clock running time analysis. RL experiments involve both network training and environment interaction on CPUs; therefore, wall-clock time is typically not regarded as a standard evaluation metric. Nevertheless, to provide a clear picture of the practical efficiency of CAE, we report wall-clock running times on MiniHack tasks. CAE requires approximately 17 hours to complete training on each task, whereas E3B takes around 22 hours, demonstrating the superior efficiency of CAE.

Table 5: Experimental results after $2e7 - 3e7$ interaction steps on nine MiniHack tasks, whose detailed descriptions are in Appendix E.2. *RPI* quantifies the improvement of CAE+ compared to E3B. Since E3B is the state-of-the-art on MiniHack (Henaff et al., 2022), and CAE+ consistently outperforms it, we conclude that **CAE+ $\succ$ E3B, ICM, RND, RIDE, NovelD**. Refer to Appendix E for experimental figures.

Env Alg.	N4	N4-Locked	N6	N6-Locked	N10-OD	Horn	Random	Wand	LavaCross
E3B	0.86 ± 0.010	0.72 ± 0.090	0.75 ± 0.019	-0.31 ± 0.403	0.71 ± 0.042	0.47 ± 0.071	0.57 ± 0.071	0.49 ± 0.201	0.22 ± 0.412
CAE	0.93 ± 0.014	0.84 ± 0.041	-0.46 ± 0.193	-0.37 ± 0.310	-0.84 ± 0.216	$\mathbf{0.92 \pm 0.022}$	$\mathbf{0.93 \pm 0.022}$	$\mathbf{0.93 \pm 0.030}$	0.16 ± 0.394
CAE+	$\mathbf{0.97 \pm 0.006}$	$\mathbf{0.87 \pm 0.017}$	$\mathbf{0.94 \pm 0.014}$	$\mathbf{0.77 \pm 0.093}$	$\mathbf{0.86 \pm 0.023}$	0.84 ± 0.055	0.80 ± 0.040	0.65 ± 0.131	$\mathbf{0.84 \pm 0.024}$
RPI %	12.79	20.83	25.3	348.39	21.13	78.72	40.35	32.65	281.82

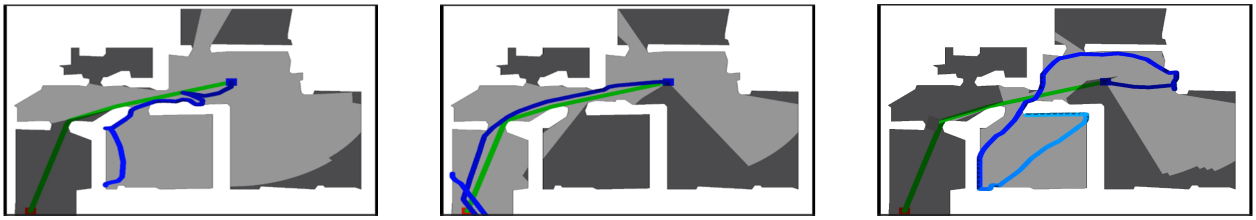
5.3 Reward-free Habitat task

Habitat (Savva et al., 2019; Andrew et al., 2021; Xavi et al., 2023) is a platform for embodied AI research that supports agent navigation and interaction within simulations of real-world indoor environments. The experiments in this subsection are designed to evaluate exploration capabilities in visually rich, realistic settings. We employ the HM3D dataset (Santhosh et al., 2021), which comprises high-quality reconstructions of $1,000$ diverse indoor spaces. Agents are trained using intrinsic rewards for the *PointNav* task. Evaluation is conducted in unseen test environments by measuring the proportion of the revealed environment.

Table 6: Experimental results after $1e8$ interaction steps on Habitat.

E3B	CAE	CAE+	*RPI %*
0.51 ± 0.097	0.49 ± 0.102	$\mathbf{0.69 \pm 0.074}$	35.29

Results for E3B, CAE, and CAE+, averaged over three seeds $\{1, 2, 3\}$, are summarized in Table 6, highlighting the superior performance of CAE+. Since E3B is the state-of-the-art on Habitat, and CAE+ further outperforms E3B, it follows that **CAE+ $\succ$ E3B, ICM, RND, RIDE, NovelD**. Figure 3 provides an illustration of a specific case, where the trained policy of CAE+ explores a larger portion of the environment compared to that of E3B.



(a) CAE+ reveals 0.67 of the map

(b) E3B reveals 0.53 of the map

(c) CAE reveals 0.51 of the map

Figure 3: Trajectories of the learned policies on a Habitat environment unseen during training.

6 Conclusion

In this paper, we propose CAE, a lightweight exploration method that integrates with existing RL algorithms without adding parameters. CAE exploits the value network's embedding layers to guide exploration, requiring no changes to the rest of the algorithm. A simple scaling strategy ensures its stability. For sparse-reward tasks, we extend it to CAE+ by adding a small auxiliary network, accelerating learning with minimal overhead. We provide theoretical guarantees with sub-linear regret bounds and demonstrate strong sample efficiency. Extensive experiments show that CAE and CAE+ consistently outperform state-of-the-art methods across dense and sparse reward settings.

Broader Impact Statement

CAE and CAE+ bridge the gap between provably efficient and practically successful exploration methods. While theoretically grounded approaches often suffer from scalability and applicability issues, empirically driven methods typically lack theoretical guarantees and require extensive parameter training. Inspired by the *deep representation and shallow exploration* paradigm, this work proposes a novel framework that decomposes value networks in deep RL and repurposes them for exploration, thereby achieving theoretical guarantees with minimal or no additional parameter training.

This framework contributes to the development of sample-efficient and interpretable exploration methods in deep RL, potentially accelerating progress in applications such as robotics, recommendation systems, autonomous decision-making systems and large language models. Moreover, the lightweight nature of the method makes it suitable for deployment in resource-constrained settings. Future research could focus on integrating a broader spectrum of MAB techniques, evaluating the robustness across a wide range of tasks, and improving the computational efficiency of uncertainty estimation.

References

Alekh Agarwal, Sham Kakade, Mikael Henaff, and Wen Sun. Pc-pg: Policy cover directed exploration for provable policy gradient learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020.

Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 127–135. PMLR, 2013.

Russo Alessio and Vannella Filippo. Multi-reward best policy identification. In *Proceedings of the 38th Conference on Neural Information Processing System*, 2024.

Zanette Andrea, Brandfonbrener David, Brunskill Emma, Pirotta Matteo, and Lazaric Alessandro. Frequentist regret bounds for randomized least-squares value iteration. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

Szot Andrew, Clegg Alex, Undersander Eric, Wijmans Erik, Zhao Yili, Turner John, Maestre Noah, Mukadam Mustafa, Chaplot Devendra, Maksymets Oleksandr, Gokaslan Aaron, Vondrus Vladimir, Dharur Sameer, Meier Franziska, Galuba Wojciech, Chang Angel, Kira Zsolt, Koltun Vladlen, Malik Jitendra, Savva Manolis, and Batra Dhruv. Habitat 2.0: Training home assistants to rearrange their habitat. In *Proceedings of the 35th Conference on Neural Information Processing System*, 2021.

Y. Ng Andrew, Harada Daishi, and J. Russell Stuart. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*. PMLR, 1999.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

Jordan T. Ash, Cyril Zhang, Surbhi Goel, Akshay Krishnamurthy, and Sham Kakade. Anti-concentrated confidence bonuses for scalable exploration. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 30th Conference on Neural Information Processing System*, 2016.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and A. Alexei Efros. Large-scale study of curiosity-driven learning. In *Proceedings of the 7th International Conference on Learning Representations*, 2019a.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations*, 2019b.

Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

Tiapkin Daniil, Belomestny Denis, Moulines Éric, Naumov Alexey, Samsonov Sergey, Tang Yunhao, Valko Michal, and Pierre Ménard. From dirichlet to rubin: Optimistic exploration in rl without bonuses. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022.

Tiapkin Daniil, Belomestny Denis, Calandriello Daniele, Moulines Éric, Munos Remi, Naumov Alexey, Perrault Pierre, Valko Michal, and Ménard Pierre. Model-free posterior sampling via learning rate randomization. In *Proceedings of the 37th Conference on Neural Information Processing System*, 2023.

Jingliang Duan, Yang Guan, Shengbo Li, Yangang Ren, Qi Sun, and Bo Cheng. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 6584 – 6598, 2021.

Jingliang Duan, Wenxuan Wang, Liming Xiao, Jiaxin Gao, and Shengbo Li. Dsac-t: Distributional soft actor-critic with three refinements. *arXiv:2310.05858v4*, 2023.

Mohamed Elsayed, Gautham Vasan, and A. Rupam Mahmood. Deep reinforcement learning without experience replay, target networks, or batch updates. *38th Workshop on Fine-Tuning in Machine Learning, NeurIPS*, 2024.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. Adversarially guided actor-critic. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018.

Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via elliptical episodic bonuses. In *Proceedings of the 36th Conference on Neural Information Processing System*, 2022.

Mikael Henaff, Minqi Jiang, and Roberta Raileanu. A study of global and episodic bonuses for exploration in contextual mdps. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, pp. 1–18, 2022.

Haque Ishfaq, Qiwen Cui, Viet Nguyen, Alex Ayoub, Yang Zhuoran, Zhaoran Wang, Doina Precup, and F. Lin Yang. Randomized exploration for reinforcement learning with general value function approximation. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Haque Ishfaq, Qingfeng Lan, Pan Xu, A. Rupam Mahmood, Doina Precup, Anima Anandkumar, and Kamyar Azizzadenesheli. Provable and practical: Efficient exploration in reinforcement learning via langevin monte carlo. In *Proceedings of the 12nd International Conference on Learning Representations*, 2024a.

Haque Ishfaq, Yixin Tan, Yu Yang, Qingfeng Lan, Jianfeng Lu, A. Rupam Mahmood, Doina Precup, and Pan Xu. More efficient randomized exploration for reinforcement learning via approximate sampling. In *Proceedings of the 1st Reinforcement Learning Conference*, 2024b.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd Conference on Neural Information Processing System*, 2018.

Daniel Jarrett, Corentin Tallec, Florent Altché, Thomas Mesnard, Rémi Munos, and Michal Valko. Curiosity in hindsight: Intrinsic exploration in stochastic environments. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 2023.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient? In *Proceedings of the 32nd Conference on Neural Information Processing System*, 2018.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I. Jordan. Provably efficient reinforcement learning with linear function approximation. In *In Conference on Learning Theory*. PMLR, 2020.

Azizzadenesheli Kamyar and Anandkumar Animashree. Efficient exploration through bayesian deep q-networks. *Information Theory and Applications Workshop*, 2018.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 2002.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, 2020.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 661–670, 2010.

Yexin Li, Zhancun Mu, and Siyuan Qi. A contextual combinatorial bandit approach to negotiation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv: 1312.5602v1*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra1, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *nature*, pp. 529–533, 2015.

Ian Osband, Van Benjamin Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on Machine Learning*. PMLR, 2016.

Ian Osband, Van Benjamin Roy, J. Daniel Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, pp. 1–61, 2019.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, 2021.

K. Ramakrishnan Santhosh, Gokaslan Aaron, Wijmans Erik, Maksymets Oleksandr, Clegg Alex, Turner John, Undersander Eric, Galuba Wojciech, Westbury Andrew, X. Chang Angel, Savva Manolis, Zhao Yili, and Batra Dhruv. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv: 2109.08238v1.*, 2021.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv: 1707.06347v2*, 2017.

Jack Sherman and J. Winifred Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 1950.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, and Timothy et al. Lillicrap. Mastering the game of go without human knowledge. *Nature*, 2017.

Michal Valko, Nathaniel Korda, Remi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv: 1309.6869.*, 2013.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, and Petko et al. Georgiev. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.

B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 1962.

Zheng Wen, Branislav Kveton, and Azin Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1113–1122. PMLR, 2015.

Puig Xavi, Undersander Eric, Szot Andrew, Dallaire Cote Mikael, Partsey Ruslan, Yang Jimmy, Desai Ruta, William Clegg Alexander, Hlavac Michal, Min Tiffany, Gervet Theo, Vondrus Vladimir, Berges Vincent-Pierre, Turner John, Maksymets Oleksandr, Kira Zsolt, Kalakrishnan Mrinal, Malik Jitendra, Singh Chaplot Devendra, Jain Unnat, Batra Dhruv, Rai Akshara, and Mottaghi Roozbeh. Habitat 3.0: A co-habitat for humans, avatars, and robots. *arXiv preprint arXiv: 2310.13724v1*, 2023.

Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I. Jordan. On function approximation in reinforcement learning: Optimism in the face of large state spaces. In *Proceedings of the 34th Conference on Neural Information Processing System*, 2020.

Wang Yiming, Yang Ming, Dong Renzhi, Sun Binbin, Liu Furui, and Hou U Leong. Efficient potential-based exploration in reinforcement learning using inverse dynamic bisimulation metric. In *Proceedings of the 37th Conference on Neural Information Processing System*, 2023.

Wang Yiming, Zhao Kaiyan, Liu Furui, and Hou U Leong. Rethinking exploration in reinforcement learning with effective metric-based exploration bonus. In *Proceedings of the 38th Conference on Neural Information Processing System*, 2024.

Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Automatic intrinsic reward shaping for exploration in deep reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 40531–40554. PMLR, 2023.

Tom Zahavy and Shie Mannor. Neural linear bandits: Overcoming catastrophic forgetting through likelihood matching. *arXiv preprint arXiv: 1901.08612v2*, 2019.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. In *Proceedings of the 35th Conference on Neural Information Processing System*, 2021a.

Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. In *Proceedings of the 9th International Conference on Learning Representations*, 2021b.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 11492–11502. PMLR, 2020.

A Implementation

Our experiments are based on the following open-source codebases:

- E3B (Henaff et al., 2022): https://github.com/facebookresearch/e3b
- CleanRL (Huang et al., 2022): https://github.com/vwxyzjn/cleanrl
- DSAC (Duan et al., 2021; 2023): https://github.com/Jingliang-Duan/DSAC-v2
- Habitat-lab (Andrew et al., 2021; Xavi et al., 2023): https://github.com/facebookresearch/habitat-lab

In Listing 1, we present the core code of CAE, while the rest of the deep RL algorithm remains unchanged. As shown, CAE is simple to implement, integrates seamlessly with any existing RL algorithm, and requires no additional parameter learning beyond what is already in the RL algorithm.

Listing 1: CAE core code

```
A_inverse = torch.inverse(A)
phi = q_net.get_emb(torch.Tensor(obs), torch.Tensor(action)).squeeze().detach()
bouns = np.sqrt(torch.matmul(phi.T, torch.matmul(A_inverse, phi)).item())
reward += scaled_bonus
A += torch.outer(phi, phi)
```

In Listing 2, we present the additional code of CAE+ alongside that of CAE. As shown, CAE+ minimizes an additional loss, specifically the Inverse Dynamics Network (IDN) loss, in addition to the losses from the original RL algorithm.

Listing 2: Additional core code of CAE+

```
phi = q_net.get_emb(torch.Tensor(batch[’obs’]), torch.Tensor(default_actions))
predict_action = inverse_dynamic_net(phi[: -1], phi[1: ])
idn_loss = compute_inverse_dynamics_loss(predict_action, batch[’action’][: -1])

def compute_inverse_dynamics_loss(action, true_action):
    loss=F.nll_loss(F.log_softmax(torch.flatten(action, 0, 1), dim=-1), target=torch.flatten(true_action,
         0, 1), reduction=’none’)
    return torch.sum(torch.mean(loss.view_as(true_action), dim=1))
```

B Long version of CAE

For a more thorough theoretical analysis, we present the long and theoretical version of CAE in Algorithm 3, where, for conciseness, we denote the embedding of the state-action pair at time step h in episode m as $\boldsymbol{\phi}_h^m = \phi(s_h^m, a_h^m)$. As per the standard notation in the literature on provable algorithms (Jin et al., 2020; Yang et al., 2020), function parameters are not shared across different time steps $h \in [H]$, which is also the case in Algorithm 3. As we can see, the algorithm iteratively updates parameters $\boldsymbol{\theta}_h$ and $\boldsymbol{W}_h$, learning the two decomposed components of the action-value function in Equation 2 by Bellman equation. Specifically, the parameter $\boldsymbol{\theta}_h$ is updated in Line 9 using its closed-form solution (Li et al., 2010), while the extraction network $\phi_h(\cdot,\cdot)$ remains fixed. Afterwards, the extraction network $\phi_h(s,a|\boldsymbol{W}_h)$ is updated in Line 10, with the parameter $\boldsymbol{\theta}_h$ held constant. In this line, η is the learning rate, L_h^m is the Bellman loss function, and s_h^t, a_h^t, r_h^t for $\forall t \in [m]$ and $\forall h \in [H]$ represent historical experiences.

Algorithm 3 DQN (Mnih et al., 2015) enhanced with CAE

```
1: Input: Ridge parameter $\lambda > 0$, the exploration parameter $\alpha \geq 0$, episode length $H$, episode number $M$
2: Initialize: Gram matrix $\boldsymbol{A}_h^1 = \lambda \boldsymbol{I}$, $\boldsymbol{b}_h^1 = \boldsymbol{0}$, parameters $\boldsymbol{\theta}_h^1 \sim \frac{1}{d}N(\boldsymbol{0}, \boldsymbol{I})$, networks $\phi_h^1(\cdot,\cdot|\boldsymbol{W}_h^1)$ (Xu et al., 2022), $Q_h^1 = (\boldsymbol{\theta}_h^1)^\intercal \phi_h^1(\cdot,\cdot|\boldsymbol{W}_h^1)$, and the target value-networks $\bar{Q}_h^1 = Q_h^1$, where $h \in [H]$
3: for episode $m = 1$ to $M$ do
4:     Sample the initial state of the episode $s_1^m$
5:     for step $h = 1, 2, ..., H$ do
6:         Conduct action $a_h^m = \arg\max_a Q_h^m(s_h^m, a)$ and get the next state $s_{h+1}^m$ and reward $r_h^m$
7:         Compute the target value $q_h^m = r_h^m + \gamma \cdot \max_a \bar{Q}_{h+1}^m(s_{h+1}^m, a)$
8:         Update $\boldsymbol{A}_h^{m+1} = \boldsymbol{A}_h^m + \boldsymbol{\phi}_h^m(\boldsymbol{\phi}_h^m)^\intercal$ and $\boldsymbol{b}_h^{m+1} = \boldsymbol{b}_h^m + q_h^m \boldsymbol{\phi}_h^m$
9:         Update parameter $\boldsymbol{\theta}_h^{m+1} = (\boldsymbol{A}_h^{m+1})^{-1} \boldsymbol{b}_h^{m+1}$
10:        Update the extraction network to $\phi_h^{m+1}(\cdot,\cdot)$ with parameters $\boldsymbol{W}_h^{m+1} = \boldsymbol{W}_h^m + \eta \nabla_{\boldsymbol{W}_h^m} L_h^m$ where
               $L_h^m = \sum_{t=1}^{m} \left| (\boldsymbol{\theta}_h^{m+1})^\intercal \phi_h^m(s_h^t, a_h^t | \boldsymbol{W}_h^m) - r_h^t - \gamma \cdot \max_a \bar{Q}_{h+1}^m(s_{h+1}^t, a) \right|^2$
11:        Obtain UCB-based uncertainty
               $\beta_h^{m+1}(\cdot,\cdot) = \sqrt{(\phi_h^{m+1}(\cdot,\cdot))^\intercal (\boldsymbol{A}_h^{m+1})^{-1} \phi_h^{m+1}(\cdot,\cdot)}$
12:        Obtain Thompson Sampling-based uncertainty
               $\boldsymbol{\Delta\theta}_h^{m+1} \sim N(0, (\boldsymbol{A}_h^{m+1})^{-1}) \Longrightarrow \beta_h^{m+1}(\cdot,\cdot) = (\boldsymbol{\Delta\theta}_h^{m+1})^\intercal \phi_h^{m+1}(\cdot,\cdot)$
13:        Approximate the action-value function
               $Q_h^{m+1}(\cdot,\cdot) = (\boldsymbol{\theta}_h^{m+1})^\intercal \phi_h^{m+1}(\cdot,\cdot) + \alpha \beta_h^{m+1}(\cdot,\cdot)$
14:    end for
15:    Update the target network $\bar{Q}_h^{m+1}(\cdot,\cdot) = Q_h^{m+1}(\cdot,\cdot), h \in [H]$
16: end for
```

Notably, $Q_h^m(s_h^m, a_h^m)$ denotes the estimated value, whereas q_h^m represents the corresponding target value at each time step h in episode m. By concatenating the values over all time steps $h \in [H]$ and episodes $m \in [M]$, we construct the estimated value vector $\tilde{\boldsymbol{q}}$ and the target value vector $\boldsymbol{q}$ of the action-value network, as referenced in Theorem 1.

C Proof of Theorem 1

In this section, we analyze the cumulative regret bound of Algorithm 3. As $\boldsymbol{U}$ is a straightforward linear transformation of the embeddings, it approximately preserves the theoretical guarantees of UCB- and Thompson Sampling-based exploration strategies. This ensures the rigor of CAE+.

Before delving into the detailed theory, we first review the notation used in this appendix. Let π^* denote the true optimal policy and π_m represent the policy executed in episode $m \in [M]$. The action-value and state-value functions corresponding to the policies π^* and π_m are represented by Q^*, V^*, and Q^{π_m}, V^{π_m}, respectively. The relationship between the state-value and action-value functions under a specific policy, such as π_m, is given as follows.

$$V_h^{\pi_m}(s) = \max_a Q_h^{\pi_m}(s,a) \tag{16}$$

$$Q_h^{\pi_m}(s,a) = r_h(s,a) + \mathbb{E}_{s_{h+1}\sim\mathbb{P}_h(\cdot|s,a)} V_{h+1}^{\pi_m}(s_{h+1}) \tag{17}$$

In Algorithm 3, the estimated action-value function at step h in episode m is denoted by $Q_h^m(s,a)$, with the corresponding state-value function represented as $V_h^m(s)$. For clarity of presentation, we introduce the following additional notations.

$$(\mathbb{P}_h V_{h+1}^m)(s_h^m, a_h^m) = \mathbb{E}_{s_{h+1}^m\sim\mathbb{P}_h(\cdot|s_h^m,a_h^m)} V_{h+1}^m(s_{h+1}^m). \tag{18}$$

$$\delta_h^m(s_h^m, a_h^m) = r_h^m + (\mathbb{P}_h V_{h+1}^m)(s_h^m, a_h^m) - Q_h^m(s_h^m, a_h^m). \tag{19}$$

$$\zeta_h^m = V_h^m(s_h^m) - V_h^{\pi_m}(s_h^m) + Q_h^m(s_h^m, a_h^m) - Q_h^{\pi_m}(s_h^m, a_h^m). \tag{20}$$

$$\varepsilon_h^m = (\mathbb{P}_h V_{h+1}^m)(s_h^m, a_h^m) - (\mathbb{P}_h V_{h+1}^{\pi_m})(s_h^m, a_h^m) + V_{h+1}^m(s_{h+1}^m) - V_{h+1}^{\pi_m}(s_{h+1}^m). \tag{21}$$

Specifically, $\delta_h^m(s_h^m, a_h^m)$ represents the temporal-difference error for the state-action pair (s_h^m, a_h^m). The notations ζ_h^m and ε_h^m capture two sources of randomness, *i.e.*, the selection of action $a_h^m \sim \pi_m(\cdot|s_h^m)$ and the generation of the next state $s_{h+1}^m \sim \mathbb{P}_h(\cdot|s_h^m, a_h^m)$ from the environment.

Proof. **Theorem 1**.

Based on Lemma 1, the cumulative regret in Equation 13 can be decomposed into three terms as follows, where $\langle\cdot,\cdot\rangle$ means the inner product of two vectors.

$$\begin{aligned}
\mathrm{R}_M &= \sum_{m=1}^{M} Q_1^*(s_1^m, u_1^m) - Q_1^{\pi_m}(s_1^m, a_1^m) \\
&= \sum_{m=1}^{M}\sum_{h=1}^{H} \left[\mathbb{E}_{\pi^*}\left[\delta_h^m(s_h, a_h)|s_1 = s_1^m\right] - \delta_h^m(s_h^m, a_h^m)\right] + \sum_{m=1}^{M}\sum_{h=1}^{H}(\zeta_h^m + \varepsilon_h^m) \\
&\quad + \sum_{m=1}^{M}\sum_{h=1}^{H} \mathbb{E}_{\pi^*}\left[\langle Q_h^m(s_h,\cdot), \pi_h^*(\cdot|s_h) - \pi_m(\cdot|s_h)\rangle\,|s_1 = s_1^m\right]
\end{aligned}$$

According to the definition of π_m, there is Equation 22.

$$\langle Q_h^m(s_h,\cdot), \pi_h^*(\cdot|s_h) - \pi_m(\cdot|s_h)\rangle \le 0 \tag{22}$$

Consequently, with probability at least $1-\sigma$ where $\sigma \in (0,1)$, the cumulative regret in Equation 13 can be bounded as follows.

$$\begin{aligned}
\mathrm{R}_M \leq & \sum_{m=1}^{M}\sum_{h=1}^{H}\left[\mathbb{E}_{\pi^*}\left[\delta_h^m(s_h,a_h)|s_1=s_1^m\right]-\delta_h^m(s_h^m,a_h^m)\right]+\sum_{m=1}^{M}\sum_{h=1}^{H}(\zeta_h^m+\varepsilon_h^m) \\
\leq & \sum_{m=1}^{M}\sum_{h=1}^{H}\left[\mathbb{E}_{\pi^*}\left[\delta_h^m(s_h,a_h)|s_1=s_1^m\right]-\delta_h^m(s_h^m,a_h^m)\right]+\sqrt{16MH^3\log\frac{2}{\sigma_1}} \\
\leq & H\sqrt{2MH\log\frac{2}{\sigma_2}}+C_2\alpha H\sqrt{Md\cdot\log(1+\frac{M}{\lambda d}})\\
& +\frac{C_3\cdot HL^3d^{\frac{5}{2}}M\sqrt{\log(\iota+\frac{1}{\sigma_2}+\frac{M|\mathcal{A}|}{\sigma_2})}\,\|\boldsymbol{q}-\tilde{\boldsymbol{q}}\|_{\boldsymbol{H}^{-1}}}{\iota^{\frac{1}{6}}}+\sqrt{16MH^3\log\frac{2}{\sigma_1}} \\
\leq & C_4H\sqrt{MH\log\frac{2}{\sigma}}+C_2\alpha H\sqrt{Md\cdot\log(1+\frac{M}{\lambda d}})+\frac{C_3HL^3d^{\frac{5}{2}}M\sqrt{\log(\iota+\frac{1}{\sigma}+\frac{M|\mathcal{A}|}{\sigma})}\,\|\boldsymbol{q}-\tilde{\boldsymbol{q}}\|_{\boldsymbol{H}^{-1}}}{\iota^{\frac{1}{6}}}
\end{aligned} \tag{23}$$

Specifically, the second inequality is based on Lemma 2, while the third one is based on Lemma 3. By choosing $\sigma = \max\{\sigma_1, \sigma_2\}$ and $C_4 \geq \sqrt{2}+4$, we complete the proof.

□

D Lemmas

Lemma 1. *Adapted from Lemma 5.1 of Yang et al. (2020), the regret in Equation 13 can be decomposed as Equation 24, where $\langle\cdot,\cdot\rangle$ means the inner product of two vectors.*

$$\begin{aligned}
\mathrm{R}_M &= \sum_{m=1}^{M} Q_1^*(s_1^m,u_1^m)-Q_1^{\pi_m}(s_1^m,a_1^m) \\
&= \sum_{m=1}^{M} V_1^*(s_1^m)-V_1^{\pi_m}(s_1^m) \\
&= \sum_{m=1}^{M}\sum_{h=1}^{H}\left[\mathbb{E}_{\pi^*}\left[\delta_h^m(s_h,a_h)|s_1=s_1^m\right]-\delta_h^m(s_h^m,a_h^m)\right]+\sum_{m=1}^{M}\sum_{h=1}^{H}(\zeta_h^m+\varepsilon_h^m) \\
&\quad+\sum_{m=1}^{M}\sum_{h=1}^{H}\mathbb{E}_{\pi^*}\left[\langle Q_h^m(s_h,\cdot),\pi_h^*(\cdot|s_h)-\pi_m(\cdot|s_h)\rangle\,|s_1=s_1^m\right]
\end{aligned} \tag{24}$$

Lemma 2. *Adapted from Lemma 5.3 of Yang et al. (2020), with probability at least $1-\sigma_1$, the second term in Equation 23 can be bounded as follows:*

$$\sum_{m=1}^{M}\sum_{h=1}^{H}(\zeta_h^m+\varepsilon_h^m) \leq \sqrt{16MH^3\log\frac{2}{\sigma_1}} \tag{25}$$

Lemma 3. *For any $\sigma_2 \in (0,1)$, assume the width of the action-value network satisfies:*

$$\iota = \mathrm{poly}(L, d, \frac{1}{\sigma_2}, \log\frac{M|\mathcal{A}|}{\sigma_2}) \tag{26}$$

where L is the number of layers in the action-value network, and $\mathrm{poly}(\cdot)$ *means a polynomial function depending on the incorporated variables, and let:*

$$\alpha = \sqrt{2(d \cdot \log(1 + \frac{M \cdot \log|\mathcal{A}|}{\lambda}) - \log \sigma_2)} + \sqrt{\lambda} \tag{27}$$

$$\eta \leq C_1(\iota \cdot d^2 M^{\frac{11}{2}} L^6 \cdot \log \frac{M|\mathcal{A}|}{\sigma_2})^{-1} \tag{28}$$

then with probability at least $1 - \sigma_2$*, the first term in Equation 23 is bounded as:*

$$\begin{aligned}
&\sum_{m=1}^{M} \sum_{h=1}^{H} [\mathbb{E}_{\pi^*} [\delta_h^m(s_h, a_h)|s_1 = s_1^m] - \delta_h^m(s_h^m, a_h^m)] \qquad (29)\\
\leq & H\sqrt{2MH \log \frac{2}{\sigma_2}} + C_2 \alpha H \sqrt{Md \cdot \log(1 + \frac{M}{\lambda d})} + \frac{C_3 \cdot HL^3 d^{\frac{5}{2}} M \sqrt{\log(\iota + \frac{1}{\sigma_2} + \frac{M|\mathcal{A}|}{\sigma_2})} \, \|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{\boldsymbol{H}^{-1}}}{\iota^{\frac{1}{6}}}
\end{aligned}$$

Proof. According to Yang et al. (2020), there is:

$$\sum_{m=1}^{M} \sum_{h=1}^{H} [\mathbb{E}_{\pi^*} [\delta_h^m(s_h, a_h)|s_1 = s_1^m] - \delta_h^m(s_h^m, a_h^m)] \leq \sum_{m=1}^{M} \sum_{h=1}^{H} -\delta_h^m(s_h^m, a_h^m) \tag{30}$$

Considering $\delta_h^m(s_h^m, a_h^m)$, it can be decomposed as:

$$\begin{aligned}
\delta_h^m(s_h^m, a_h^m) =& r_h^m + (\mathbb{P}_h V_{h+1}^m)(s_h^m, a_h^m) - Q_h^m(s_h^m, a_h^m) \qquad (31)\\
=& r_h^m + (\mathbb{P}_h V_{h+1}^m)(s_h^m, a_h^m) - Q_h^*(s_h^m, a_h^m) + Q_h^*(s_h^m, a_h^m) - Q_h^m(s_h^m, a_h^m)\\
=& \mathbb{P}_h(V_{h+1}^m - V_{h+1}^*)(s_h^m, a_h^m) + (Q_h^* - Q_h^m)(s_h^m, a_h^m)\\
=& \underbrace{\mathbb{P}_h(V_{h+1}^m - V_{h+1}^*)(s_h^m, a_h^m) - (V_{h+1}^m - V_{h+1}^*)(s_{h+1}^m)}_{\omega_h^m}\\
& + \underbrace{(V_{h+1}^m - V_{h+1}^*)(s_{h+1}^m)}_{\rho_{h+1}^m} + \underbrace{(Q_h^* - Q_h^m)(s_h^m, a_h^m)}_{\varphi_h^m}
\end{aligned}$$

By Azuma-Hoeffding inequality, we can bound $\sum_{m=1}^{M} \sum_{h=1}^{H} \omega_h^m$ as Equation 32 with probability at least $1 - \sigma_3$ where $\sigma_3 \in (0, 1)$.

$$-H\sqrt{2MH \log \frac{2}{\sigma_3}} \leq \sum_{m=1}^{M} \sum_{h=1}^{H} \omega_h^m \leq H\sqrt{2MH \log \frac{2}{\sigma_3}} \tag{32}$$

As ρ_{h+1}^m can be decomposed as Equation 33 where $u_{h+1}^m \sim \pi_{h+1}^*(\cdot|s_{h+1}^m)$, there is Equation 34.

$$\rho_{h+1}^m = (V_{h+1}^m - V_{h+1}^*)(s_{h+1}^m) = Q_{h+1}^m(s_{h+1}^m, a_{h+1}^m) - Q_{h+1}^*(s_{h+1}^m, u_{h+1}^m) \tag{33}$$

$$\Rightarrow \sum_{m=1}^{M}\sum_{h=1}^{H}(\rho_{h+1}^m + \varphi_h^m) \tag{34}$$

$$= \sum_{m=1}^{M}\sum_{h=1}^{H-1}\left[Q_{h+1}^m(s_{h+1}^m, a_{h+1}^m) - Q_{h+1}^*(s_{h+1}^m, u_{h+1}^m)\right] + \sum_{m=1}^{M}\sum_{h=1}^{H}(Q_h^* - Q_h^m)(s_h^m, a_h^m)$$

$$= \underbrace{\sum_{m=1}^{M}\sum_{h=2}^{H} Q_h^*(s_h^m, a_h^m) - Q_h^*(s_h^m, u_h^m)}_{\mathrm{R}_{\mathrm{MAB}}} + \sum_{m=1}^{M}(Q_1^* - Q_1^m)(s_1^m, a_1^m) \tag{35}$$

Specifically, the second equation is because of $Q_{H+1}^*(s_{H+1}^m, a_{H+1}^m) = 0$ and $Q_{H+1}^m(s_{H+1}^m, a_{H+1}^m) = 0$. The second term in Equation 35 originates from the estimation error of the action-value function, which is constrained by the convergence properties of DQN. Consequently, to complete the proof of Lemma 3, it suffices to establish a bound for the R $_{\mathrm{MAB}}$ term, while the second term is omitted for conciseness in the remaining discussion. Bounds of R $_{\mathrm{MAB}}$ under UCB- and Thompson Sampling-based exploration strategies are proved in Lemma 4 and Lemma 5, respectively.

Choosing $\sigma_2 = \max\{\sigma_3, \sigma_4\}$ and $C_2 = \max\{C_{ucb}, C_{ts}\}$ completes this proof.

□

D.1 Regret Bound of UCB-based Exploration

Lemma 4. *Adapted from Theorem 4.4 of Xu et al. (2022), suppose the standard initializations and assumptions hold. Additionally, assume without loss of generality that $\|\boldsymbol{\theta}^*\|_2 \le 1$ and $\|\phi(s_h, a_h)\|_2 \le 1$. If with the UCB-based exploration strategy, then for any $\sigma_4 \in (0, 1)$, let the width of the action-value network satisfies:*

$$\iota = \mathrm{poly}(L, d, \frac{1}{\sigma_4}, \log\frac{M|\mathcal{A}|}{\sigma_4}) \tag{36}$$

where L is the number of layers in the action-value network, and $\mathrm{poly}(\cdot)$ means a polynomial function depending on the incorporated variables, and let:

$$\alpha = \sqrt{2(d\cdot\log(1+\frac{M\cdot\log|\mathcal{A}|}{\lambda}) - \log\sigma_4)} + \sqrt{\lambda} \tag{37}$$

$$\eta \le C_1(\iota\cdot d^2 M^{\frac{11}{2}} L^6 \cdot \log\frac{M|\mathcal{A}|}{\sigma_4})^{-1} \tag{38}$$

then with probability at least $1-\sigma_4$, the term $\mathrm{R}_{\mathrm{UCB}}$ in Equation 35 can be bounded as follows:

$$\mathrm{R}_{\mathrm{UCB}} \le C_{ucb}\cdot\alpha H\sqrt{Md\cdot\log(1+\frac{M}{\lambda d})} + \frac{C_3 H L^3 d^{\frac{5}{2}} M\sqrt{\log(\iota+\frac{1}{\sigma_4}+\frac{M|\mathcal{A}|}{\sigma_4})}\,\|\boldsymbol{q}-\tilde{\boldsymbol{q}}\|_{\boldsymbol{H}^{-1}}}{\iota^{\frac{1}{6}}} \tag{39}$$

where C_1, C_{ucb}, C_3 are constants independent of the problem; $\boldsymbol{q} = (q_1^1; q_2^1; ...; q_H^1; ...; q_H^M)$ and $\tilde{\boldsymbol{q}} = (Q_1^1(s_1^1, a_1^1); Q_2^1(s_2^1, a_2^1); ...; Q_H^1(s_H^1, a_H^1); ...; Q_H^M(s_H^M, a_H^M))$ are the target and the estimated value vectors, respectively, as detailed in Appendix B.

Notably, the proof of the above lemma relies on the concentration of self-normalized stochastic processes. However, since Q_h^m is not independent of historical data, this result cannot be directly applied. Instead, a similar approach to that in Yang et al. (2020) is adopted by leveraging Uniform Convergence across all possible inputs within the value function class $\mathcal{Q}$. This ensures that the maximum deviation between the true and learned values over all time steps remains small, *i.e.*, $\sup_{Q_h^m \in \mathcal{Q}} |Q_h^m - Q_h^*| \leq \epsilon_m$. Crucially, the error ϵ_m decreases as the number of episodes increases. Applying $(a+b)^2 \leq 2a^2 + 2b^2$, we decompose the error bound $\mathrm{R}_{\mathrm{UCB}}$ into two terms to manage the dependency:

- A true optimal value function component, to which the concentration of self-normalized stochastic processes applies.
- A cumulative error term dependent on ϵ_m which can be systematically bounded.

D.2 Regret Bound of Thompson Sampling-based Exploration

Lemma 5. *Under the same settings as those of Lemma 4, if with the Thompson Sampling-based exploration strategy, the term* $\mathrm{R}_{\text{ Thompson Sampling}}$ *in Equation 35 can be bounded as Equation 40, where* C_{ts} *is another problem-independent constant.*

$$\mathrm{R}_{\text{Thompson Sampling}} \leq C_{ts} \cdot \alpha H \sqrt{Md \cdot \log(1 + \frac{M}{\lambda d})} + \frac{C_3 H L^3 d^{\frac{5}{2}} M \sqrt{\log(\iota + \frac{1}{\sigma_4} + \frac{M|\mathcal{A}|}{\sigma_4})} \, \|\boldsymbol{q} - \tilde{\boldsymbol{q}}\|_{\boldsymbol{H}^{-1}}}{\iota^{\frac{1}{6}}} \tag{40}$$

Proof. According to Lemma A.1 of Xu et al. (2022), $Q_h^*(s,u) - Q_h^*(s,a)$ can be decomposed as Equation 41, where $g(s,a;\boldsymbol{W}) = \nabla_{\boldsymbol{W}} \phi(s,a;\boldsymbol{W})$.

$$\begin{aligned} & Q_h^*(s,u) - Q_h^*(s,a) \\ =& (\boldsymbol{\theta}_h^*)^\intercal \left[\phi(s,u;\boldsymbol{W}_h^m) - \phi(s,a;\boldsymbol{W}_h^m)\right] + (\boldsymbol{\theta}_h^1)^\intercal \left[g(s,u;\boldsymbol{W}_h^1) - g(s,a;\boldsymbol{W}_h^1)\right] (\boldsymbol{W}_h^* - \boldsymbol{W}_h^m) \\ =& (\boldsymbol{\theta}_h^1)^\intercal \left[g(s,u;\boldsymbol{W}_h^1) - g(s,a;\boldsymbol{W}_h^1)\right] (\boldsymbol{W}_h^* - \boldsymbol{W}_h^m) \\ & + \underbrace{(\boldsymbol{\theta}_h^m)^\intercal \left[\phi(s,u;\boldsymbol{W}_h^m) - \phi(s,a;\boldsymbol{W}_h^m)\right]}_{\vartheta_h^m} - (\boldsymbol{\theta}_h^m - \boldsymbol{\theta}_h^*)^\intercal \left[\phi(s,u;\boldsymbol{W}_h^m) - \phi(s,a;\boldsymbol{W}_h^m)\right] \end{aligned} \tag{41}$$

Based on the action selection process using Thompson Sampling-based exploration strategy in Algorithm 3, we derive Equation 42.

$$(\boldsymbol{\theta}_h^m + \alpha_h^m \Delta \boldsymbol{\theta}_h^m)^\intercal \phi(s,u;\boldsymbol{W}_h^m) \leq (\boldsymbol{\theta}_h^m + \alpha_h^m \Delta \boldsymbol{\theta}_h^m)^\intercal \phi(s,a;\boldsymbol{W}_h^m) \tag{42}$$

Consequently, ϑ_h^m can be bounded as Equation 43.

$$\begin{aligned} \vartheta_h^m \leq & \|\Delta \boldsymbol{\theta}_h^m\|_{\boldsymbol{A}_h^m} \|\phi(s,a;\boldsymbol{W}_h^m) - \phi(s,u;\boldsymbol{W}_h^m)\|_{(\boldsymbol{A}_h^m)^{-1}} \\ \leq & (\sqrt{d} + \sqrt{2 \log \frac{1}{\sigma_4}}) \|\phi(s,a;\boldsymbol{W}_h^m) - \phi(s,u;\boldsymbol{W}_h^m)\|_{(\boldsymbol{A}_h^m)^{-1}} \end{aligned} \tag{43}$$

Specifically, the last inequality above is because $\boldsymbol{\Delta \theta}_h^m \sim N(0, (\boldsymbol{A}_h^m)^{-1})$. Substituting the bound of ϑ_h^m back into Equation 41 further yields:

$$
\begin{aligned}
Q_h^*(s,u) - Q_h^*(s,a) \leq & (\boldsymbol{\theta}_h^1)^\top \left[g(s,u;\boldsymbol{W}_h^1) - g(s,a;\boldsymbol{W}_h^1)\right](\boldsymbol{W}_h^* - \boldsymbol{W}_h^m) \\
& + (\sqrt{d} + \sqrt{2\log\frac{1}{\sigma_4}})\,\|\phi(s,a;\boldsymbol{W}_h^m) - \phi(s,u;\boldsymbol{W}_h^m)\|_{(\boldsymbol{A}_h^m)^{-1}} \\
& - (\boldsymbol{\theta}_h^m - \boldsymbol{\theta}_h^*)^\top\left[\phi(s,u;\boldsymbol{W}_h^m) - \phi(s,a;\boldsymbol{W}_h^m)\right]
\end{aligned} \tag{44}
$$

Comparing Equation 44 with A.7 of Xu et al. (2022), the difference between the regrets of Thompson Sampling-based and UCB-based exploration strategies is bounded as Equation 45, with probability at least $1-\sigma_4$ where $\sigma_4 \in (0,1)$.

$$
\begin{aligned}
& |\mathrm{R}_{\text{Thompson Sampling}} - \mathrm{R}_{\text{UCB}}| \\
\leq & \sum_{m=1}^{M}\sum_{h=1}^{H}(\sqrt{d} + \sqrt{2\log\frac{1}{\sigma_4}})\,\|\phi(s,a;\boldsymbol{W}_h^m) - \phi(s,u;\boldsymbol{W}_h^m)\|_{(\boldsymbol{A}_h^m)^{-1}} \\
& + \sum_{m=1}^{M}\sum_{h=1}^{H}\alpha_h^m\,\|\phi(s,a;\boldsymbol{W}_h^m)\|_{(\boldsymbol{A}_h^m)^{-1}} + \sum_{m=1}^{M}\sum_{h=1}^{H}\alpha_h^m\,\|\phi(s,u;\boldsymbol{W}_h^m)\|_{(\boldsymbol{A}_h^m)^{-1}} \\
\leq & H\sqrt{Md\log(1+\frac{M}{\lambda d}})(\sqrt{d\log(1+\frac{M\log MA}{\lambda}) + \log\frac{1}{\sigma_4}} + \sqrt{\lambda}) \\
\leq & C_5\alpha H\sqrt{Md\cdot\log(1+\frac{M}{\lambda d}})
\end{aligned} \tag{45}
$$

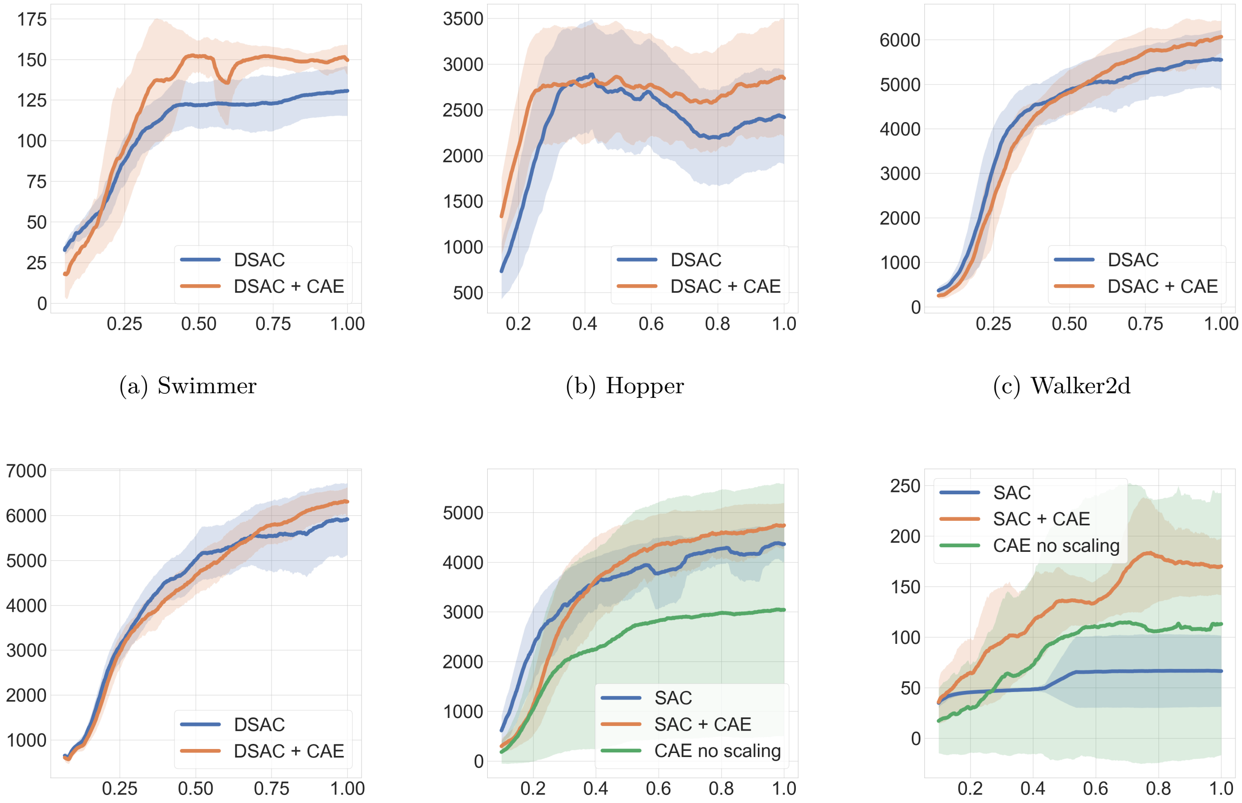
Setting $C_{ts} = C_{ucb} + C_5$ completes the proof. □

D.3 Discussion about the Standard Assumptions

Regarding Assumption 1, it can be readily satisfied by transforming the state–action pairs $(s;a)$ into $(s;a;s;a)$ and applying an appropriate scaling.

Regarding Assumption 2, it holds when the gradient of the neural network with respect to certain weights does not fluctuate excessively. In other words, within a neighborhood of a given state–action pair, the gradient with respect to these weights remains well-controlled. This is a standard assumption in the non-convex optimization literature, commonly used to ensure the convergence of alternating optimization procedures in which parameters are updated iteratively.

Regarding Assumption 3, it is a mild condition. Specifically, prior studies have demonstrated that for two-layer ReLU networks, this assumption can be directly derived from Asumption 1. A diverse input distribution and a wide neural network can largely ensure that $\boldsymbol{H}$ remains positive definite.



(a) Swimmer

(b) Hopper

(c) Walker2d

(d) Ant

(e) Walker2d

(f) Swimmer

Figure 4: Experimental results of DSAC on MuJoCo-v3, *i.e.*, from Figure 4a to Figure 4d, and those of ablation study to the *scaling strategy* on MuJoCo-v4, *i.e.*, Figure 4e and Figure 4f.

E Experiment

In this section, we present additional experimental results and provide the hyperparameters used to reproduce the experiments reported in this paper.

E.1 Experiment on Mujoco

Ablation study to the *scaling strategy*. Additionally, we conduct experiments to assess the necessity of the scaling strategy. Experiment results, illustrated in Figure 4e and Figure 4f, are based on randomly selected tasks for SAC. As shown, SAC enhanced with CAE fails to deliver satisfactory performance on *Walker2d* and *Swimmer* tasks when the scaling strategy is not applied. It achieves high performance under certain seeds, while it performs poorly under others, leading to poor overall results and large variance. This highlights the critical role of the scaling strategy in ensuring the practical stability and effectiveness of CAE.

Hyperparameters of various RL algorithms for the experiments on MuJoCo are completely the same as those in the public codebase CleanRL. CAE introduces only two more hyperparameters, *i.e.*, the exploration coefficient α and the ridge which is set as $\lambda = 1$. Exploration coefficients are summarized in Table 7 for various tasks and algorithms, and the experimental results on various MuJoCo tasks involving different RL algorithms are in Figure 4 and Figure 5. Notably, we only present results for cases where the *RPI* exceeds 5.0%, as smaller *RPI* is not clearly distinguishable in the figures.

Table 7: Exploration coefficient for various MuJoCo tasks and algorithms

Algorithms Env	SAC	PPO	TD3	DSAC
Swimmer	0.2	0.1	0.1	0.1
Ant	0.7	0.2	0.3	1.0
Walker2d	1.0	0.13	0.8	3.0
Hopper	0.4	0.14	0.3	2.0
HalfCheetah	0.4	0.5	3.7	3.0
Humanoid	4.0	0.1	6.0	4.5

E.2 Experiment on MiniHack

In Subsection 5.2, we evaluate our methods CAE and CAE+ on nine representative MiniHack tasks, whose detailed descriptions are provided in Table 8.

Table 8: MiniHack tasks evaluated in Subsection 5.2.

Short names in Subsection 5.2	Full names	Task types
N4	MultiRoom-N4	Navigation-based
N4-Locked	MultiRoom-N4-Locked	
N6	MultiRoom-N6	
N6-Locked	MultiRoom-N6-Locked	
N10-OD	MultiRoom-N10-OpenDoor	
Horn	Freeze-Horn-Restricted	Skill-based
Random	Freeze-Random-Restricted	
Wand	Freeze-Wand-Restricted	
LavaCross	LavaCross-Restricted	

The hyperparameters for IMPALA, E3B, CAE, and CAE+ used in our experiments are summarized in Table 9 and Table 10, aligning with those from the E3B experiments (Henaff et al., 2022). The experimental results on MiniHack are presented in Figure 6.

Table 9: IMPALA Hyperparameters for MiniHack (Henaff et al., 2022)

Learning rate	0.0001
RMSProp smoothing constant	0.99
RMSProp momentum	0
RMSProp	10^{-5}
Unroll length	80
Number of buffers	80
Number of learner threads	4
Number of actor threads	8
Max gradient norm	40
Entropy cost	0.0005
Baseline cost	0.5
Discounting factor	0.99

Table 10: E3B, CAE, and CAE+ Hyperparameters for MiniHack

E3B, CAE, and CAE+	Running intrinsic reward normalization	True
	Ridge regularizer	0.1
	Entropy Cost	0.005
	Exploration coefficient	1
CAE+	Dimension of $\boldsymbol{U}$	256

E.3 Experiment on Habitat

The hyperparameters for PPO, E3B, CAE, and CAE+ used in our experiments are summarized in Table 11 and Table 12.

Table 11: PPO Hyperparameters for Habitat are adopted from *habitat-lab* (Savva et al., 2019; Andrew et al., 2021; Xavi et al., 2023)

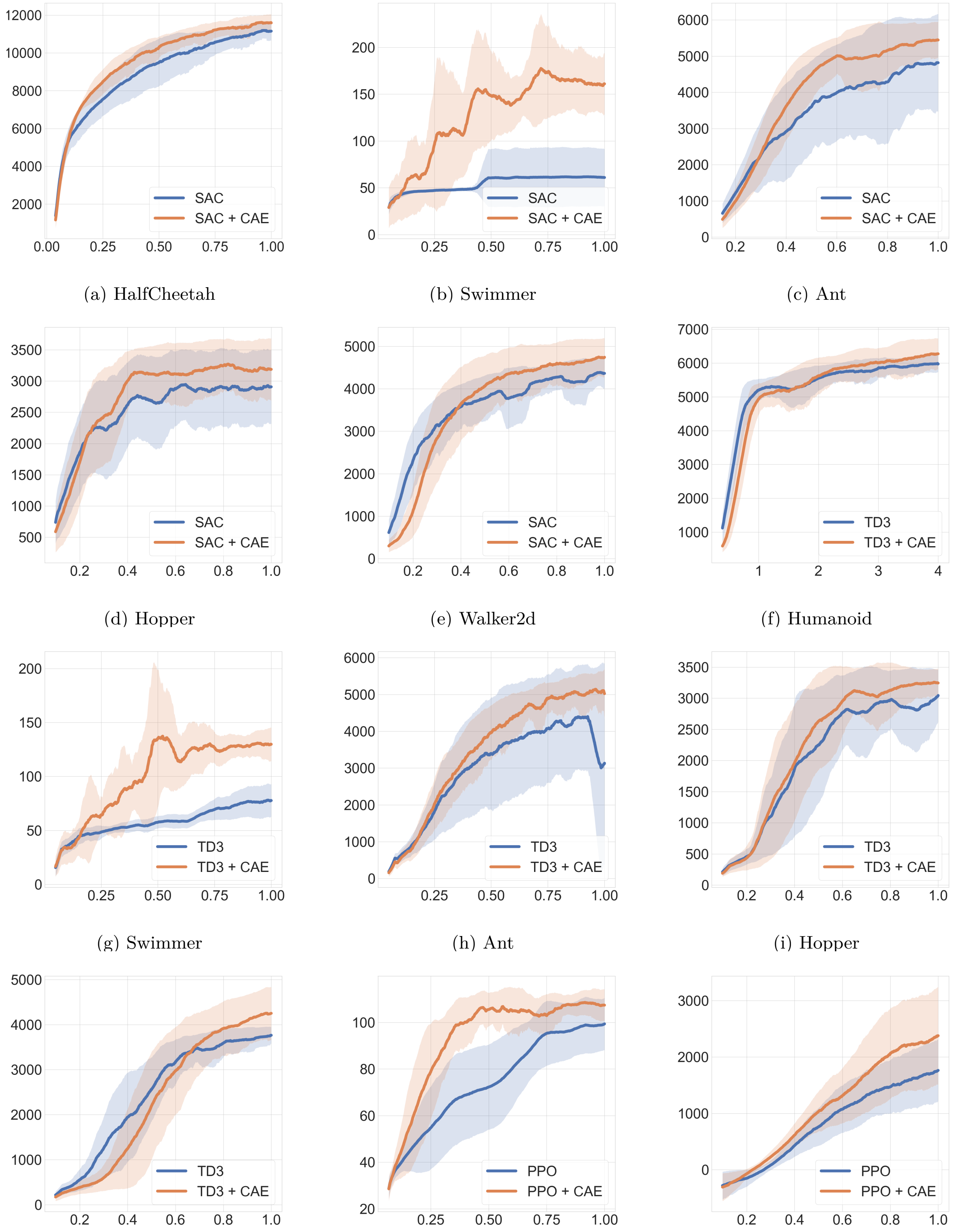
Clipping	0.2
PPO epoch	4
Value loss coefficient	0.5
Entropy coefficient	0.01
Learning rate	$2.5e-4$
$\epsilon-$greedy	1e-5
Max gradient norm	0.2
Rollout steps	128
Use GAE	True
Discounting factor	0.99
Number of actor threads	16

Table 12: E3B, CAE, and CAE+ Hyperparameters for Habitat

E3B, CAE, and CAE+	Running intrinsic reward normalization	False
	Ridge regularizer	0.1
	Inverse Dynamics Model updates per PPO epoch	3
	Exploration coefficient	0.1
CAE+	Dimension of $\boldsymbol{U}$	256

F Limitations

One limitation of the proposed method is that, despite the small number of additional trainable parameters, computing the uncertainty incurs a non-negligible computational overhead due to the need for matrix inversion. This can be mitigated by using approximation techniques, such as the **Rank-1** method described in subsection 3.3. Additionally, other linear MAB methods could be integrated into the proposed framework to avoid the need to calculate the inverse Gram matrix.



(a) HalfCheetah

(b) Swimmer

(c) Ant

(d) Hopper

(e) Walker2d

(f) Humanoid

(g) Swimmer

(h) Ant

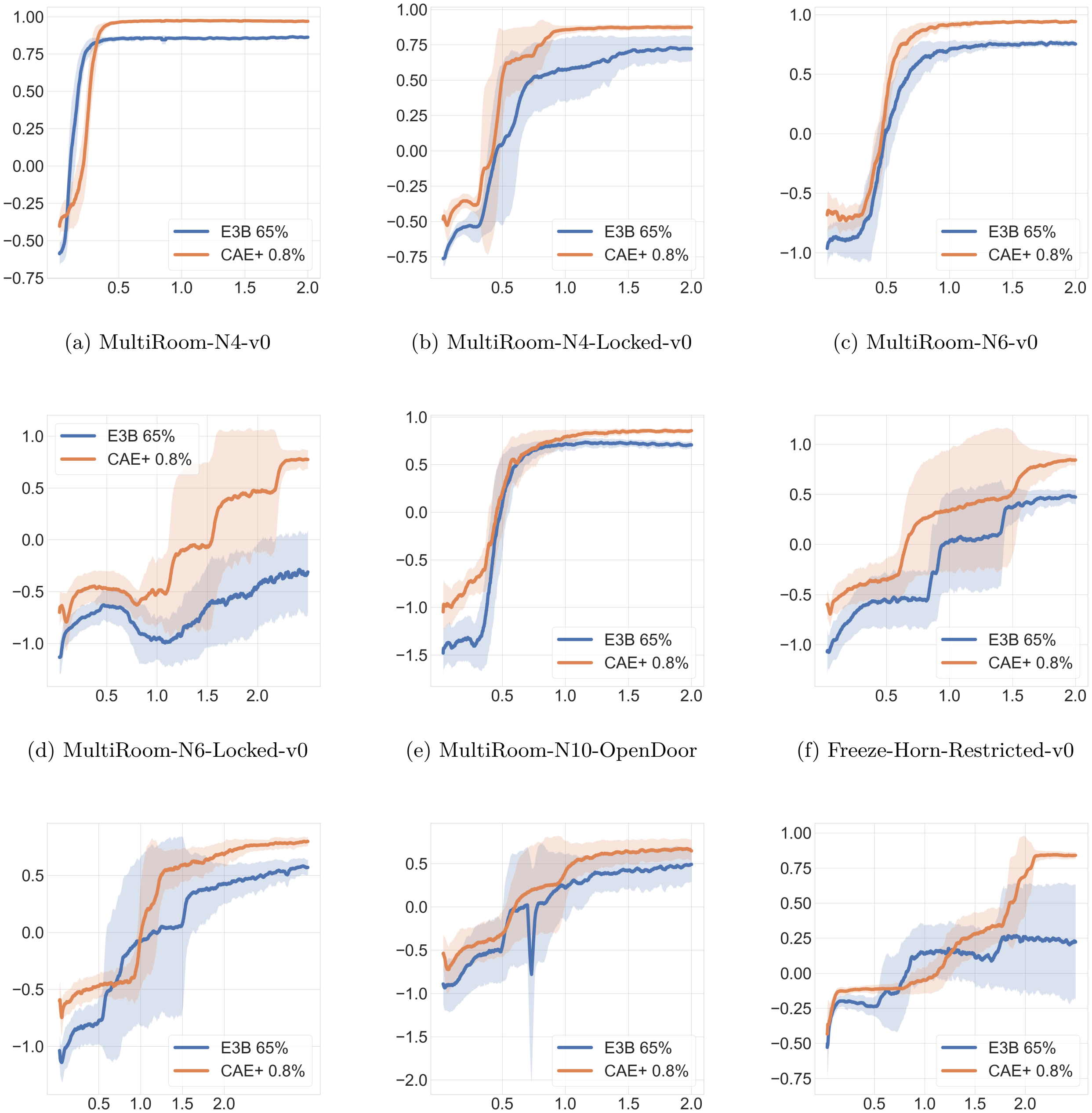
(i) Hopper

(j) Walker2d

(k) Swimmer

(l) Ant

Figure 5: Experimental results on MuJoCo-v4.



(a) MultiRoom-N4-v0

(b) MultiRoom-N4-Locked-v0

(c) MultiRoom-N6-v0

(d) MultiRoom-N6-Locked-v0

(e) MultiRoom-N10-OpenDoor

(f) Freeze-Horn-Restricted-v0

(g) Freeze-Random-Restricted-v0

(h) Freeze-Wand-Restricted-v0

(i) LavaCross-Restricted

Figure 6: Experimental results on MiniHack.