Learning Uncertainty-Aware Temporally-Extended Actions

Joongkyu Lee[1*], **Seung Joon Park**[2*], **Yunhao Tang**[3], **Min-hwan Oh**[1]
[1]Seoul National University
[2]Samsung Research
[3]Google DeepMind
jklee0717@snu.ac.kr, soonjun.park@samsung.com, robintyh@deepmind.com, minoh@snu.ac.kr

Abstract

In reinforcement learning, temporal abstraction in the action space, exemplified by action repetition, is a technique to facilitate policy learning through extended actions. However, a primary limitation in previous studies of action repetition is its potential to degrade performance, particularly when sub-optimal actions are repeated. This issue often negates the advantages of action repetition. To address this, we propose a novel algorithm named **U**ncertainty-aware **T**emporal **E**xtension (UTE). UTE employs ensemble methods to accurately measure uncertainty during action extension. This feature allows policies to strategically choose between emphasizing exploration or adopting an uncertainty-averse approach, tailored to their specific needs. We demonstrate the effectiveness of UTE through experiments in Gridworld and Atari 2600 environments. Our findings show that UTE outperforms existing action repetition algorithms, effectively mitigating their inherent limitations and significantly enhancing policy learning efficiency.

Introduction

Temporal abstraction is a promising approach to solving complex tasks in reinforcement learning (RL) with complex structures and long horizons (Fikes, Hart, and Nilsson 1972; Dayan and Hinton 1992; Parr and Russell 1997; Sutton, Precup, and Singh 1999; Precup 2000; Bacon, Harb, and Precup 2017; Barreto et al. 2019; Machado, Barreto, and Precup 2021). Hierarchical reinforcement learning (HRL) enables the decomposition of this sequential decision-making problem into simpler lower-level actions or subtasks. Intuitively, an agent explores the environment more effectively when operating at a higher level of abstraction and solving smaller subtasks (Machado, Barreto, and Precup 2021). One of the most prominent approaches for HRL is the *option* framework (Sutton, Precup, and Singh 1999; Precup 2000), which describes the hierarchical structure in decision making in terms of temporally-extended courses of action. Temporally-extended actions have been shown to speed up learning, potentially providing more effective exploration compared to single-step explorative action and requiring a smaller number of high-level decisions when solving a problem (Stolle and Precup 2002; Biedenkapp et al. 2021). From a cognitive perspective, such observations are also coherent with how humans learn, generalize from experiences, and perform abstraction over tasks (Xia and Collins 2021).

There has been a line of works that propose repetition of action for an extended period as a specialized form of temporal abstraction (Lakshminarayanan, Sharma, and Ravindran 2017; Sharma, Srinivas, and Ravindran 2017; Dabney, Ostrovski, and Barreto 2020; Metelli et al. 2020; Biedenkapp et al. 2021; Park, Kim, and Kim 2021).[1] Hence, the action-repetition methods address the problem of learning when to perform a new action while repeating an action for multiple time-steps (Dabney, Ostrovski, and Barreto 2020; Biedenkapp et al. 2021). The extension length, the interaction steps to repeat the same action, is learned by an agent along with what action to execute (Sharma, Srinivas, and Ravindran 2017; Biedenkapp et al. 2021). As shown by the improved empirical performances (Dabney, Ostrovski, and Barreto 2020; Biedenkapp et al. 2021), these action repetition approaches can be well justified by the *commitment* to action for deriving a deeper exploration. These approaches can help suppress the dithering behavior of the agent that can result in short-sighted exploration in a local neighborhood.

However, simple action repetition alone cannot guarantee performance improvement. Repetition of a sub-optimal action for an extended period can lead to severe deterioration in the performance. For example, a game may terminate due to reckless action repetition when an agent is in a dangerous region. A more uncertainty-averse behavior would be helpful in this scenario. On the other hand, an agent may linger in the local neighborhood due to a lack of optimism, especially in sparse reward settings. In that case, a more exploration-favor behavior can be beneficial. In either case, a suitable control of uncertainty of value estimates over longer horizons can be a crucial element. In particular, the calibration of how much exploration the agent can take, or how uncertainty-averse the agent should be, can definitely depend on an environment. Thus, the degree of uncertainty to be considered should be

[*]These authors contributed equally.

[1]In fact, action repetition for a fixed number of steps was one of the strategies deployed in solving Atari 2600 games (Mnih et al. 2015; Machado et al. 2018). Despite its simplicity, the action repetition provided sufficient performance gains so that almost all modern methods of solving Atari games are still implementing such action repetitions.

adaptive depending on the environment. To this end, we propose to account for uncertainties when repeating actions. To our best knowledge, consideration of uncertainty in the future when instantiating action repetition has been not addressed previously. Such consideration is essential in action repetition in both uncertainty-averse and exploration-favor environments.

In this paper, we propose a novel method that learns to repeat actions while incorporating the estimated uncertainty of the repeated action values. We can either impose aggressive or uncertainty-averse exploration by controlling the degree of uncertainty in order to take suitable uncertainty-aware strategy for the environment. Through extensive experiments and ablation studies, we demonstrate the efficacy of our proposed method and how it enhances the performances of deep reinforcement learning agents in various environments. In comparison with the benchmarks, we show that our proposed method outperforms baselines, consistently outperforming the existing action repetition methods. Our contributions are:

- We present a novel framework that allows the agent to repeat actions in a uncertainty-aware manner using an ensemble method. Suitably controlling the amount of uncertainty induced by repeated actions, our proposed method learns to choose extension length and learns how *optimistic* or *pessimistic* it should be, hence enabling efficient exploration.
- Our method yields a salient insight that it is beneficial to consider environment-inherent uncertainty preference. Some environments are uncertainty-favor (Chain MDP), and some are uncertainty-averse (Gridworlds).
- In a set of testing environments, we show `UTE` consistently outperforms all of the existing action-repetition baselines, such as `DAR`, ϵz-`Greedy`, `DQN`, `B-DQN`, in terms of final evaluation scores, learning speed, and coverage of state-spaces.

Related Work

Temporal Abstraction and Action Repetition. Temporal abstractions can be viewed as an attempt to find a time scale that is adequate for describing the actions of an AI system (Precup 2000). The options framework (Sutton, Precup, and Singh 1999; Precup 2000; Bacon, Harb, and Precup 2017) formalizes the idea of temporally-extended actions. An MDP endowed with a set of options are called Semi-Markov Decision Process (SMDP) which we define in Preliminaries. The generalization of conventional action-value functions for the options framework is called *option*-value functions (Sutton, Precup, and Singh 1999). The mapping from states to probabilities of taking an option is called policy over options. In the options framework, the agent attempts to learn a policy over options that maximizes the option-value functions.

One simple form of an option is repeating a primitive action for certain number of steps (Schoknecht and Riedmiller 2002). Action repetition has been widely explored in the literature (Lakshminarayanan, Sharma, and Ravindran 2017; Sharma, Srinivas, and Ravindran 2017; Dabney, Ostrovski, and Barreto 2020; Metelli et al. 2020; Biedenkapp et al. 2021; Park, Kim, and Kim 2021). Action repetition has been empirically shown to induce deeper exploration (Dabney, Ostrovski, and Barreto 2020) and lead to efficient learning by reducing the granularity of control (Lakshminarayanan, Sharma, and Ravindran 2017; Sharma, Srinivas, and Ravindran 2017; Metelli et al. 2020; Biedenkapp et al. 2021). Action repetition can be implemented by deciding the extension length of an action which is either sampled from a distribution (Dabney, Ostrovski, and Barreto 2020) or returned by a policy (Lakshminarayanan, Sharma, and Ravindran 2017; Sharma, Srinivas, and Ravindran 2017). The closest related to our work is Biedenkapp et al. (2021). They proposed an algorithm called `TempoRL` that not only selects an action in a state but also for how long to commit to that action. `TempoRL` (Biedenkapp et al. 2021) proposes a hierarchical structure in which *behavior* policy determines the action a to be played given the current state s, and a *skip* policy determines how long to repeat this action. However, our main intuition is that simply repeating the chosen action is not enough. We may encounter undesirable states while repeating the action. This could lead to catastrophic failure when an agent enters a "risky" area (refer Gridworlds experiments). Our method has been shown to effectively manage this issue by quantifying the uncertainty of the option in form of repeating actions.

Uncertainty in Reinforcement Learning. Recently, many works have made significant advances in empirical studies by quantifying and incorporating uncertainty (Osband et al. 2016; Bellemare et al. 2016; Badia et al. 2020; Lee et al. 2022). There are two types of uncertainty: aleatoric and epistemic. Aleatoric uncertainty is the uncertainty caused by the uncontrollable stochastic nature of the environment and cannot be reduced. Epistemic uncertainty is caused by the current imperfect training of the neural network and can be reducible.

One mainstream of estimating the uncertainty in deep RL relies on bootstrapping. Osband et al. (2016) introduced Bootstrapped DQN as a method for effcient exploration. This approach is a variation of the classic DQN neural network architecture, which has a shared torso with $K \in \mathbb{Z}^+$ heads. Anschel, Baram, and Shimkin (2017); Peer et al. (2021) leveraged an ensemble of Q-functions to mitigate overestimation in DQN. In this paper, we propose an algorithm that quantifies uncertainty of Q-value estimates of the states reached under the repeated-action. This algorithm utilizes multiple randomly-initialized bootstrapped heads that stretch out from a shared network, providing multiple estimates of the *option*-value function. The variance between these estimates is then used as a measure of uncertainty. Notably, this approach allows us to capture both aleatoric and epistemic uncertainty. Then, we establish a UCB-style (Auer, Cesa-Bianchi, and Fischer 2002; Audibert, Munos, and Szepesvári 2009) option-selecting algorithm that simply adds the estimated uncertainty to the averaged ensemble Q-values and chooses an action that maximizes the quantity (Chen et al. 2017; Peer et al. 2021).

Preliminaries and Notations

In reinforcement learning, an agent interacts with an environment whose underlying dynamics is modeled by a Markov Decision Process (MDP) (Puterman 2014). The tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ defines an MDP $\mathcal{M}$, where $\mathcal{S}$ is a state

space, $\mathcal{A}$ is an action space, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a transition dynamics function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1]$ is the discount factor. We consider a Semi-Markov Decision Process (SMDP) model to incorporate the options framework (Sutton, Precup, and Singh 1999; Precup 2000). An SMDP is an original MDP with a set of options, i.e., $\mathcal{M}_o := \langle \mathcal{S}, \Omega, P_o, R_o \rangle$, where $\omega \in \Omega$ is an option in the option space, $P_o(s' \mid s, \omega) : \mathcal{S} \times \Omega \to \mathcal{S}$ is the probability of transitioning from state s to state s' after taking an option ω and $R_o : \mathcal{S} \times \Omega \to \mathbb{R}$ is the reward function for the option.

For any set $\mathcal{X}$, let $\mathcal{P}(\mathcal{X})$ denote the space of probability distributions over $\mathcal{X}$. Then a policy over option $\pi_\omega : \mathcal{S} \to \mathcal{P}(\Omega)$ assigns a probability to an option conditioned on a given state. Our goal is to learn a policy π_ω that maximizes the expectation of discounted return starting from a initial state s_0; then, define the value functions $V^{\pi_\omega}(s_0) = \mathbb{E}_{\pi_\omega}[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0]$, the action-value functions $Q^{\pi_\omega}(s_0, a) = \mathbb{E}_{\pi_\omega}[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0, a]$, or the option-value functions $\widetilde{Q}^{\pi_\omega}(s_0, \omega) = \mathbb{E}_{\pi_\omega}[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0, \omega]$.

In general, options depend on the entire *history* between time step t when they were initiated and the current time step $t + k$, $h_{t:t+k} := s_t a_t s_{t+1} ... a_{t+k-1} s_{t+k}$. Let $\mathcal{H}$ be the space of all possible histories h, then a *semi-Markov option* ω is a tuple $\omega := \langle \mathcal{I}_o, \pi_o, \beta_o \rangle$, where $\mathcal{I}_o \subset \mathcal{S}$ is an initiation set, $\pi_o : \mathcal{H} \to \mathcal{P}(\mathcal{A})$ is an *intra-option* policy, and $\beta_o : \mathcal{H} \to [0, 1]$ is a termination function. In this framework, we define an action repeating option to be $\omega_{aj} := \langle \mathcal{S}, \mathbf{1}_a, \beta(h) = \mathbf{1}_{|h|=j} \rangle$, in which $h \in \mathcal{H}$ and $\mathbf{1}_a$ indicates $|\mathcal{A}|$-dimensional vector where the element corresponding to a is 1 and 0 otherwise. This action repeating option takes action a for j times and then terminates.

When an agent plays a chosen action for extension length j, total of $\frac{j(j+1)}{2}$ skip-transitions are observed and stored in the replay buffer (Biedenkapp et al. 2021). Specifically, when repeating the action for j times from state s, we can also experience $(s \to s'_{(1)}), (s \to s'_{(2)}), \ldots, (s'_{(1)} \to s'_{(2)}), \ldots, (s'_{(j-1)} \to s'_{(j)})$, in total $\frac{j \cdot (j+1)}{2}$ transitions. We leverage these transitions to update option-values. Consequently, the observations for short extensions are updated more frequently, leading to smaller uncertainties for short extensions and larger uncertainties for long extensions.

Uncertainty-Aware Temporal Extension

In this section, we propose our algorithm UTE: **U**ncertainty-Aware **T**emporal **E**xtension, which repeats the action in consideration of uncertainty in Q-values. We first demonstrate temporally-extended Q-learning by decomposing the action repeating option. We then describe how we estimate the uncertainty of an option-value function $\widetilde{Q}^{\pi_\omega}$ by utilizing the ensemble method to select an extension length j in consideration of uncertainty. We additionally show that *n*-step targets can be used for learning the action-value function Q^{π_ω} without worrying about off-policy correction.

Temporally-extended Q-Learning

In this work, we mainly depend on techniques based on the Q-learning algorithm (Watkins and Dayan 1992), which seeks to approximate the Bellman optimality operator to learn the optimal policy:

Algorithm 1: UTE: Uncertainty-Aware Temporal Extension

Input: uncertainty parameter λ, the number of output heads of option-value functions B.
Initialize: Q^{π_ω}, $\{\widetilde{Q}^{\pi_\omega}_{(b)}\}_{b=1}^{B}$.
for episode = $1, \ldots, K$ **do**
  Obtain initial state s from environment
  repeat
    $a \leftarrow \epsilon$-greedy $\text{argmax}_{a'}\, Q^{\pi_\omega}(s, a)$
    Calculate $\hat{\mu}_{\pi_\omega}(s, \omega_{aj})$, $\hat{\sigma}^2_{\pi_\omega}(s, \omega_{aj})$ by Eq. (4).
    $j \leftarrow \text{argmax}_{j'} \{\hat{\mu}_{\pi_\omega}(s, \omega_{aj'}) + \lambda \hat{\sigma}_{\pi_\omega}(s, \omega_{aj'})\}$
    while $j \neq 0$ and s is not terminal **do**
      Take action a and observe s', r
      $s \leftarrow s'$, $j \leftarrow j - 1$
    end while
  until episode ends
end for

Definition 1. *We define the optimal action-value function $Q^{\pi^*_\omega}$ and the optimal option-value function $\widetilde{Q}^{\pi^*_\omega}$ respectively as*

$$Q^{\pi^*_\omega}(s, a) = \mathbb{E}_{s'_{(1)} \sim P}\Big[R(s, a) + \gamma \max_{a'} Q^{\pi^*_\omega}(s'_{(1)}, a')\Big], \tag{1}$$

$$\widetilde{Q}^{\pi^*_\omega}(s, \omega_{aj}) = \mathbb{E}_{s'_{(j)} \sim P_o}\Big[R_o(s, \omega_{aj}) + \gamma^j \max_{\omega'} \widetilde{Q}^{\pi^*_\omega}(s'_{(j)}, \omega')\Big], \tag{2}$$

where $s'_{(0)}$ and $s'_{(j)}$, respectively, indicate one-step and j-step later state from the state s . In practice, it is common to use a function approximator to estimate each Q-value, $Q^{\pi_\omega}(s, a; \theta) \approx Q^{\pi^*_\omega}(s, a)$ and $\widetilde{Q}^{\pi_\omega}(s, \omega_{aj}; \phi) \approx \widetilde{Q}^{\pi^*_\omega}(s, \omega_{aj})$. We use two different neural network function approximators parameterized by θ and ϕ respectively.

Option Decomposition. Learning the optimal policy over options, instead of the optimal action policy, has the same effect as enlarging the action space from $|\mathcal{A}|$ to $|\mathcal{A}| \times |\mathcal{J}|$, where $\mathcal{J} = \{1, 2, \ldots, \text{max repetition}\}$. Generally, inaccuracies in Q-function estimations can cause the learning process to converge to a sub-optimal policy, and this phenomenon is amplified in situations with large action spaces (Thrun and Schwartz 1993; Zahavy et al. 2018). Therefore, we consider decomposed policy over option (Biedenkapp et al. 2021), $\pi_\omega(\omega_{aj} \mid s) := \pi_a(a \mid s) \cdot \pi_e(j \mid s, a)$, in which an action policy $\pi_a(a \mid s) : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ assigns some probability to each action conditioned on a given state, and then an extension policy $\pi_e(j \mid s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{J})$ assigns some probability to each extension length conditioned on a given state and action. Note that there exists a hierarchy between decomposed policies π_a and π_e, thus, π_a always has to be queried before π_e at every time an option initiates. The agent first chooses an action a from action policy π_a based on the action-value function Q^{π_ω} (e.g. ϵ-greedy). Then, given this action a, it selects extension length j from π_e according to the option-value function $\widetilde{Q}^{\pi_\omega}$.

By decomposing the policy over option π_ω, we can decrease the search space from $|\mathcal{A}| \times |\mathcal{J}|$ to $|\mathcal{A}| + |\mathcal{J}|$. We empirically show that decomposing option can stabilize the Q-learning in Appendix. However, this learning process may converge to a sub-optimal policy because it is intractable to search all the possible combinations of actions and extension lengths (a, j). The agent may repeat the sub-optimal action excessively or sometimes be overly myopic. Our algorithm can mitigate this issue by controlling the level of uncertainty when executing the extension policy π_e.

Proposition 1. *In a Semi-Markov Decision Process (SMDP), let an option $\omega \in \Omega$ be the action repeating option defined by action a and extension length j, i.e. $\omega_{aj} := \langle \mathcal{S}, \mathbf{1}_a, \beta(h) = \mathbf{1}_{h=j} \rangle$. For all $\omega \in \Omega$, a policy over option, π_ω, can be decomposed by an action policy $\pi_a(a \mid s) : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ and an extension policy $\pi_e(j \mid s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(|J|)$, i.e. $\pi_\omega(\omega_{aj} \mid s) := \pi_a(a \mid s) \cdot \pi_e(j \mid s, a)$. Then, for the corresponding optimal policy π^*_ω, the following holds:*

$$V^{\pi^*_\omega}(s) = \max_{\omega_{aj}} Q^{\pi^*_\omega}(s, \omega_{aj}) = \max_a Q^{\pi^*_\omega}(s, a).$$

Proposition 1 implies that the target value for the option selection of repeated actions can be the same as the target for a single-step action selection within the option. In our implementation, we use $\max_{a'} Q^{\pi^*_\omega}(s'_{(j)}, a')$ instead of $\max_{\omega'} \widetilde{Q}^{\pi^*_\omega}(s'_{(j)}, \omega'_{aj})$ for the target value in Eq.(2). This can stabilize the learning process by sharing the same target.

Ensemble-based Uncertainty Quantification

In the previous action repetition methods (Lakshminarayanan, Sharma, and Ravindran 2017; Sharma, Srinivas, and Ravindran 2017; Dabney, Ostrovski, and Barreto 2020; Biedenkapp et al. 2021), they extend the chosen action without considering uncertainty which could easily run to failure. The only situation where these problems do not occur is when their extension policies are optimal, which means they need to expect the j step later state precisely. However, it is improbable in the sense that this situation rarely occurs in the learning process. In order to solve this problem, we propose a strategy of choosing a extension length j in an uncertainty-aware manner. `UTE` is a uncertainty-aware version of the `TempoRL` (Biedenkapp et al. 2021). Our main intuition is that it is crucial to consider the uncertainty of option-value functions $\widetilde{Q}^{\pi_\omega}$, when selecting extension length j by extension policy π_e.

We use the ensemble method, which has recently become prevalent in RL (Osband et al. 2016; Da Silva et al. 2020; Bai et al. 2021), to estimate uncertainty in our estimated option-value functions. We use a network consisting of a shared architecture with B independent. "head" branching off from the shared network. Each head corresponds to a option-value function, $\widetilde{Q}^{\pi_\omega}_{(b)}$, for $b \in \{1, 2, \ldots, B\}$. Each head is randomly-initialized and trained by different samples from an experience buffer. Unlike Bootstrapped DQN (`B-DQN`) (Osband et al. 2016) where each one of the value function heads is trained against its own target network, our `UTE` trains each value function head against the same target. If each head has its own target head respectively, since the objective function of neural networks is generally non-convex, each Q-value may converge to different modes. In this case, as training the policy, the estimated uncertainty of option Q-value, $\hat{\sigma}_{\pi_\omega}$, could not converge to zero. This means that it is unable to learn an optimal policy. Therefore, using the same target is one of the key points of our implementation.

Given state s and action a, $\widetilde{Q}^{\pi_\omega}_{(b)}$-values are aggregated by extension length j to estimate mean and variance as follows:

$$\hat{\mu}_{\pi_\omega}(s, \omega_{aj}) := \frac{1}{B} \sum_{b=1}^{B} \widetilde{Q}^{\pi_\omega}_{(b)}(s, \omega_{aj}) \tag{3}$$

$$\hat{\sigma}^2_{\pi_\omega}(s, \omega_{aj}) := \frac{1}{B} \sum_{b=1}^{B} (\widetilde{Q}^{\pi_\omega}_{(b)}(s, \omega_{aj}))^2 - (\hat{\mu}_{\pi_\omega}(s, \omega_{aj}))^2 \tag{4}$$

Then, we define *uncertainty-aware* extension policy π_e, which takes extension length j deterministically given state and action, by introducing the uncertainty parameter $\lambda \in \mathbb{R}$:

$$j = \operatorname*{argmax}_{j' \in \mathcal{J}} \{\hat{\mu}_{\pi_\omega}(s, \omega_{aj'}) + \lambda \hat{\sigma}_{\pi_\omega}(s, \omega_{aj'})\}.$$

where λ indicates the level of uncertainty to be considered. The positive λ induces more aggressive exploration, and the negative one causes uncertainty-averse exploration.

n-step Q-Learning

We make use of n-step Q-learning (Sutton 1988) to learn both Q^{π_ω} and $\widetilde{Q}^{\pi_\omega}$, whereas `TempoRL` (Biedenkapp et al. 2021) used it only for updating $\widetilde{Q}^{\pi_\omega}$. We found that n-step targets can also be used to update Q^{π_ω}-values without any off-policy correction (Harutyunyan et al. 2016), e.g., importance sampling. Given the sampled n-step transition $\tau_t = (s_t, a_t, R_o(s_t, o_{an}), s_{t+n})$ from replay buffer $\mathcal{R}$, as long as n is smaller than or equal to the current extension policy π_e's output j, the transition τ_t trivially follows our target policy π_ω. Thus, τ_t can be directly used to update the action-value function Q^{π_ω}. Instead of one step Q-learning in Eq.(1), `UTE` uses n-step Q-Learning to update Q^{π_ω}:

$$\mathcal{L}_{Q^{\pi_\omega}}(\theta) = \mathbb{E}_{\tau_t \sim \mathcal{R}} \Big[(Q^{\pi_\omega}(s_t, a_t; \theta) - \sum_{k=0}^{n-1} \gamma^k r_{t+k} - \gamma^n \max_{a'} Q^{\pi^*_\omega}(s_{t+n}, a'; \bar{\theta}))^2 \,\Big|\, n \leq j \sim \pi_e \Big]$$

where $\bar{\theta}$ are the delayed parameters of action-value function Q^{π_ω} and $j \sim \pi_e(j_t \mid s_t, a_t)$. In general, n-step returns can be used to propagate rewards faster (Watkins 1989; Peng and Williams 1994). It mitigates the overestimation problem in Q-learning as well (Meng, Gorbet, and Kulić 2021). We empirically illustrate that n-step learning leads to faster learning in Figure 12b. Note that we don't need to pre-define n because it is dynamically determined by current extension policy π_e.

Adaptive Uncertainty Parameter λ

Instead of fixing λ during the learning process, we propose the adaptive selection of λ utilizing a non-stationary multi-arm bandit algorithm, as described in (Badia et al. 2020). Consider Λ as the predefined set of uncertainty parameters. At the onset of each episode k, the bandit selects an arm, denoted by $\lambda_k \in \Lambda$, and subsequently receives feedback in the form of episode returns $R_k(\lambda_k)$. Given that the reward signal $R_k(\lambda_k)$ is non-stationary, we employ a sliding-window UCB combined with ϵ_{ucb}-greedy exploration to optimize the process. Further details regarding the algorithms can be found in Appendix.

Experiments

In this section, we present three principal experimental results: Chain MDP, Gridworlds, and Atari 2600 games, as described in Machado et al. (2018) (Machado et al. 2018). Initially, we confirm our hypothesis that a positive λ foster more aggressive exploration (ChainMDP), while a negative one results in uncertainty-averse exploration (Gridworlds). Subsequently, we demonstrate the significant impact of a well-tuned λ on performance in more complex environments and illustrate that the adaptive selection of λ consistently outperforms other baseline measures (Atari 2600 games). To ensure a fair comparison, we explored a considerable range of hyperparameters to identify the most optimal value for each algorithm (Refer Table 8, 9, 11, and 12 in Appendix)

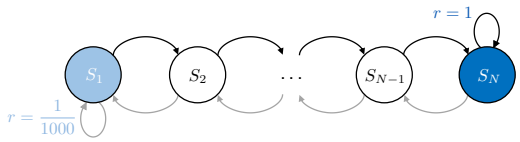


Figure 1: Chain MDP

Chain MDP

We experimented in the Chain MDP environment as described in Figure 1 (Osband et al. 2016). There are two possible actions {left, right}. If the agent reaches the left end (s_1) of the chain and performs a left action, a deceptive small reward (0.001) is given. And if the agent reaches the right end (s_n) of the chain and performs a right action, large reward (1.0) is given. Thus, the optimal policy is to take only right actions. Since the reward is very sparse, we need a "deep" exploration strategy to learn the optimal policy. In this toy environment, we will verify our intuition that positive uncertainty parameter λ induces deep exploration and as a result, show a better performance than other baselines, `DDQN` (Van Hasselt, Guez, and Silver 2016), ϵz-`Greedy` (Dabney, Ostrovski, and Barreto 2020) and `TempoRL` (Biedenkapp et al. 2021).

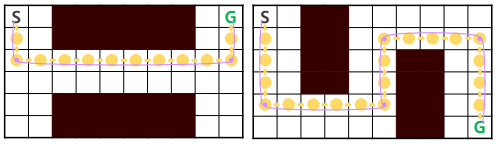
Setup. The agent interacts with the environment with a fixed horizon length, $N + 8$, where N is the chain length. Thus, the agent can obtain rewards from zero to 10 in each episode. We limited the maximum extension length as 10 for `TempoRL` and `UTE`.

Chain Length	10	30	50	70
ϵz-`Greedy`	0.654	0.427	0.434	0.131
`TempoRL`	0.904	0.740	0.246	0.052
`UTE` (ours)	**0.919**	**0.758**	**0.560**	**0.191**

Table 1: Normalized AUC on Chain MDP over 20 runs

Exploration-Favor. Table 1 summarizes the results on various levels of chain length in terms of normalized area under the reward curve (AUC), comparing `UTE` with the best uncertainty parameter (+2.0, the most optimistic λ) to ϵz-`Greedy` and `TempoRL`. A reward AUC value closer to 1.0 indicates that the agent was able to find the optimal policy faster. The total training episodes for calculating AUC was set to 1,000 across all chain lengths. The results in the table show that `UTE` outperforms the other two baselines notably throughout different chain lengths. This implies that `UTE` has a better exploration strategy which leads to higher reward even in the difficult settings (longer chain length). We also point out that `UTE` has a smaller variance than `TempoRL` after it has reached the optimal reward of 10. This is mainly because `UTE` can collect more diverse samples by exploratory extension policy π_e, which may lead to better generalization and more accurate approximation to the optimal option-value function.

More importantly, we can encode the exploration-favor strategy by adjusting the uncertainty parameter, λ. Note that we don't use ϵ-greedy for the extension policy π_e, whereas `TempoRL` do. In Appendix, Table 8 shows that more positive λ achieves higher AUC scores. When the agent selects a random action by the ϵ-greedy action policy, it can explore deeper by being more optimistic, which leads to faster convergence to the optimal solution. An aggressive exploration strategy is beneficial because the environment has no risky area where the game terminates while repeating the action.



(a) Bridge (b) ZigZag

Figure 2: 6×10 Gridworlds. Agents have to reach a goal state (G) from a starting state (S) detouring the lava. Dots and lines represent decision steps with and without temporally-extended actions, respectively.

Gridworlds

In this section, we analyze the empirical behavior of the various algorithms in the Gridworlds environment *Lava* (Figure 2). It is a 6×10 grid with discrete states and actions. An agent starts in the top-left corner and must reach the goal to receive a positive reward (+1) while avoiding stepping into the lava (-1 reward) on its way. In contrast to the chain MDP environment, since we have a risky area "lava", an

uncertainty-averse strategy must be preferred. We compare our method against vanilla DDQN (Van Hasselt, Guez, and Silver 2016) ϵz-Greedy (Dabney, Ostrovski, and Barreto 2020) and TempoRL (Biedenkapp et al. 2021).

Setup. We trained all agents for a total of 3.0×10^3 episodes using 3 different types of ϵ-greedy exploration schedule: linearly decaying from 1.0 to 0.0 over all episodes, logarithmically decaying, and fixed $\epsilon = 0.1$. We limited the maximum extension length to be 7. We use neural networks to learn Q-value functions instead of tabular Q-learning.

Env	ϵ decay	DDQN	TempoRL	ϵz-Greedy	UTE
Bridge	Linear	0.61	0.44	0.76	**0.86**
	Log	0.54	0.32	0.92	**0.92**
	Fixed	0.57	0.41	0.59	**0.83**
Zigzag	Linear	0.38	0.14	0.62	**0.84**
	Log	0.46	0.12	0.76	**0.89**
	Fixed	0.34	0.19	0.36	**0.76**

Table 2: Normalized AUC for reward across different ϵ exploration schedules over 20 random seeds.

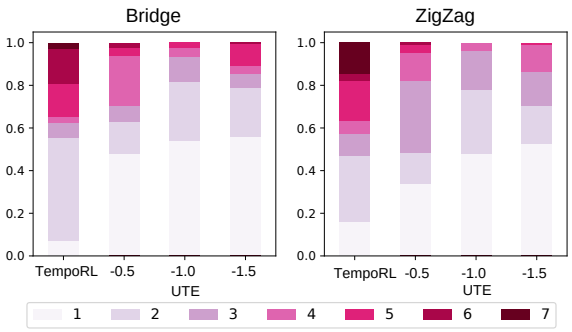


Figure 3: Distributions of extension length in Gridworlds.

Uncertainty-Averse. We compare our UTE to the other baselines in terms of normalized area under the reward curve for three different ϵ-greedy schedules (see Table 2). Across all ϵ exploration strategies, UTE outperforms other methods while showing better performance as uncertainty parameter λ becomes smaller (refer Table 9 in Appendix). This result supports our argument that a pessimistic strategy is preferred in environments with unsafe regions. Furthermore, though exploration rate for π_a is relatively large (e.g. fixed to $\epsilon = 0.1$), UTE consistently shows good performance than others.

Interestingly, the performance of TempoRL is a lot worse than the one described in the original paper (Biedenkapp et al. 2021). It is because we use function approximation to estimate Q-values, rather than tabular Q-learning. Generally, uncontrolled or undesirable overestimation bias can be caused when using function approximation (Moskovitz et al. 2021). Therefore, simply selecting extension length with the highest value leads to a catastrophic result, especially in function approximation setting. Table 2 verifies the fact that pessimistic extension policies perform well in Lava Gridworlds. Moreover, Table 9 in Appendix shows that more negative λ achieves higher AUC scores.

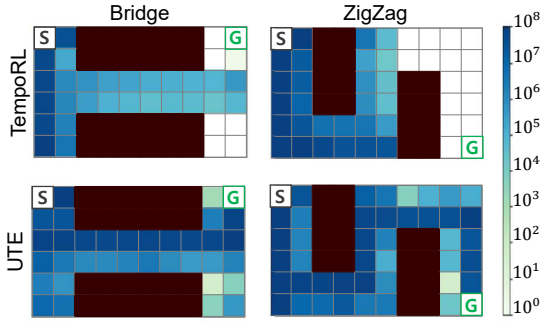


Figure 4: Coverage plots (right) on ZigZag environments. The blue represents states visited more often and white represents states rarely or never seen. See Appendix for the expanded version of the figures.

Coverage. In Figure 4, we present coverage plots comparing UTE and TempoRL on two types of Lava environments. For UTE, we have set λ to -1.5, a value that has demonstrated robust performance across tests. The results show that UTE provides significantly better coverage over the state space. We can induce our algorithm to repeat sub-optimal action less by using a pessimistic extension policy. Owing to this, our agent can survive for a longer time, leading to better coverage.

Distribution of Extension Length. Figure 3 depicts the extension length distributions of TempoRL and UTE on Birdge and ZigZag with logarithmically decaying ϵ exploration schedule. More red represents more repetitions. It shows that UTE prefers fewer repetitions compared to TempoRL when $\lambda < 0$. As previously articulated in the final paragraph in Preliminaries, observations for long extensions are seldom employed in the process of updating Q-values. Consequently, this propels our algorithm to favor fewer repetitions when $\lambda < 0$. In a pessimistic extension policy, the agent tends to refrain from repeating the chosen action many times because the value of a distant state could be much more uncertain than that of a neighbor one.

Atari 2600: Arcade Learning Environment

In this section, we evaluate the performance of UTE on the Atari benchmark, comparing the following six baseline algorithms: i) vanilla DDQN (Van Hasselt, Guez, and Silver 2016), ii) Fixed Repeat ($j = 4$), iii) ϵz-Greedy (Dabney, Ostrovski, and Barreto 2020), iv) DAR (Dynamic Action Repetition (Lakshminarayanan, Sharma, and Ravindran 2017)), v) TempoRL (Biedenkapp et al. 2021) vi) B-DQN (Bootstrapped DQN) (Osband et al. 2016). The Fixed Repeat is an algorithm that naively repeats the action a fixed amount of times.

Setup. Each algorithm is trained for a total of 2.5×10^6 training steps, which is only 10 million frames. All algorithms except B-DQN use a linearly decaying ϵ-greedy exploration schedule over the first 200,000 time-steps with a final ϵ fixed to 0.01. We evaluated all agents every 10,000 training steps

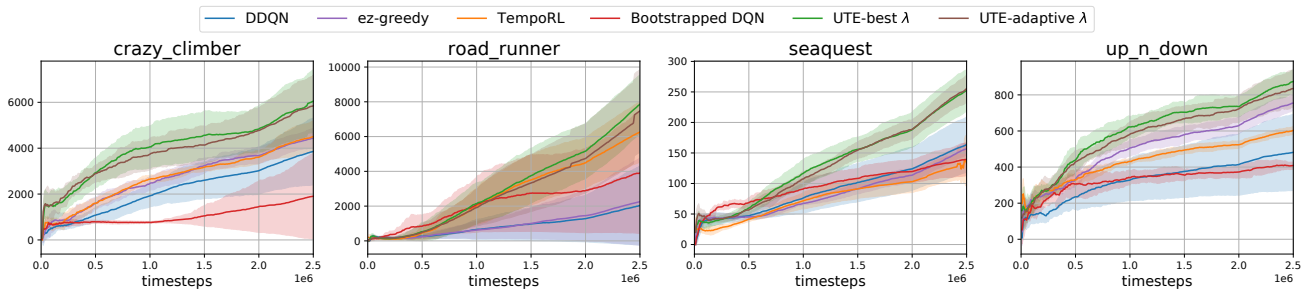


Figure 5: Learning curves of UTE with best λ, UTE with adaptive λ and other baseline algorithms on Atari environments. The shaded area represents the standard deviation over 7 random seeds.

and evaluated for 3 episodes with a very small ϵ exploration rate (0.001). We used *OpenAi Gym's* Atari environment with 4 frame-skips (Bellemare et al. 2013). For maximal extension length, we set it to 10.[2]

Uncertainty-Awareness. Figure 5 depicts learning curves for UTE and other baseline algorithms (see Figure 13 for full version). Overall, UTE achieves higher final rewards than other agents. These results demonstrate that if λ is properly tuned to the environment, our method shows significantly improved performance than existing action repetition methods (DAR, ϵz-Greedy and TempoRL) as well as a deep exploration algorithm (B-DQN). On top of that, we found that the Fixed Repeat algorithm fails at learning in most games. Hence, it is crucial to learn a extension policy for higher performance.

Adaptive Uncertainty Parameter λ. As illustrated in Figure 5, the learning speed of UTE with adaptively chosen λ is somewhat slower compared to the standard UTE. This slight decrease in speed primarily stems from the need for additional samples to optimize λ. Nevertheless, even with this adjustment, UTE with an adaptive λ continues to outperform other baseline methods by a considerable margin. These results are particularly encouraging as they obviate the need to predefine the value of λ, thereby reducing the burden of hyperparameter tuning. This aspect of our approach further underscores its practicality and effectiveness in complex learning scenarios.

Control Problem: Pendulum-v0

In this section, we show that UTE maintains its robustness to continuous control problems where there is a significant chance that repeated actions will surpass the balancing point. Consequently, selecting the appropriate extension length becomes even more crucial. We choose to evaluate on OpenAI gyms (Brockman et al. 2016) Pendulum-v0. Since the action space is continuous, we use DDPG (Lillicrap et al. 2015) as our action policy π_a, thus label it as UTE-DDPG, and apply the adaptive uncertainty parameter technique. The baseline agents are DDPG (Lillicrap et al. 2015), FiGAR (Sharma, Srinivas, and Ravindran 2017), and t-DDPG (TempoRL-DDPG) (Biedenkapp et al. 2021).

Setup. We trained all agents for a total of 3×10^4 training steps with evaluations conducted every 250 steps. For the initial 10^3 steps, a uniform random policy was applied to accumulate initial experiences.

[2]This approach aligns with the settings of Biedenkapp et al. (2021) to ensure a fair comparison.

Max J	DDPG	FiGAR	t-DDPG	UTE-DDPG
2		-172.7 (±48.6)	-163.2 (±28.6)	**-152.6** (±17.2)
4	**-156.9** (±23.2)	-352.8 (±181.5)	-160.1 (±50.7)	**-147.4** (±17.1)
6		-831.2 (±427.0)	-163.5 (±29.0)	**-159.2** (±20.8)
8		-1295.0 (±274.5)	-175.3 (±60.3)	**-165.0** (±26.4)

Table 3: Average rewards and standard deviations (numbers in parentheses) over the last 10,000 time steps in Pendulum-v0 over various maximal extension lengths (J).

Robustness to Continuous Control Environment. In Table 3, UTE-DDPG (with adaptively chosen λ) demonstrates superior performance, achieving either the top or second-best performance among the benchmarks. This suggests that our algorithm is robust to continuous control environments and consistently outperforms other established action-repeating algorithms, such as FiGAR and t-DDPG. This advantage can be attributed to our *uncertainty-aware* extension policy that prudently repeats actions. Additionally, UTE exhibits smaller standard deviations than all other baselines, except when the maximal extension length is large (i.e., $J = 8$), indicating enhanced learning stability.

Conclusion

We propose a novel method that learns to repeat actions while explicitly considering the uncertainty over the Q-value estimates of the states reached under the repeated-action option. By calibrating the level of uncertainty considered (denoted by λ), UTE consistently and significantly outperforms other algorithms, especially those focusing on action repetition, across various environments such as Chain MDP, Gridworlds, Atari 2600, and even in control problems. To our best knowledge, this is the first deep RL algorithm considering uncertainty in the future when instantiating temporally extended actions.

Acknowledgments

This work was supported by Creative-Pioneering Researchers Program through Seoul National University, and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1C1C100685912, 2022R1A4A103057912, and RS-2023-00222663).

References

Anschel, O.; Baram, N.; and Shimkin, N. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*, 176–185. PMLR.

Audibert, J.-Y.; Munos, R.; and Szepesvári, C. 2009. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19): 1876–1902.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47: 235–256.

Bacon, P.-L.; Harb, J.; and Precup, D. 2017. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Badia, A. P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, Z. D.; and Blundell, C. 2020. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, 507–517. PMLR.

Bai, C.; Wang, L.; Han, L.; Hao, J.; Garg, A.; Liu, P.; and Wang, Z. 2021. Principled exploration via optimistic bootstrapping and backward induction. In *International Conference on Machine Learning (ICML 2021)*, 577–587. PMLR.

Barreto, A.; Borsa, D.; Hou, S.; Comanici, G.; Aygün, E.; Hamel, P.; Toyama, D.; Mourad, S.; Silver, D.; Precup, D.; et al. 2019. The option keyboard: Combining skills in reinforcement learning. *Advances in Neural Information Processing Systems*, 32.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279.

Biedenkapp, A.; Rajan, R.; Hutter, F.; and Lindauer, M. 2021. TempoRL: Learning When to Act. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Chen, R. Y.; Schulman, J.; Abbeel, P.; and Sidor, S. 2017. UCB and infogain exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 9.

Da Silva, F. L.; Hernandez-Leal, P.; Kartal, B.; and Taylor, M. E. 2020. Uncertainty-aware action advising for deep reinforcement learning agents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 5792–5799.

Dabney, W.; Ostrovski, G.; and Barreto, A. 2020. Temporally-Extended ε-Greedy Exploration. In *9th International Conference on Learning Representations, ICLR 2021*.

Dayan, P.; and Hinton, G. E. 1992. Feudal reinforcement learning. *Advances in neural information processing systems*, 5.

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and executing generalized robot plans. *Artificial intelligence*, 3: 251–288.

Harutyunyan, A.; Bellemare, M. G.; Stepleton, T.; and Munos, R. 2016. Q (λ) with Off-Policy Corrections. In *International Conference on Algorithmic Learning Theory*, 305–320. Springer.

Lakshminarayanan, A.; Sharma, S.; and Ravindran, B. 2017. Dynamic action repetition for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Lee, S.; Seo, Y.; Lee, K.; Abbeel, P.; and Shin, J. 2022. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, 1702–1712. PMLR.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Machado, M. C.; Barreto, A.; and Precup, D. 2021. Temporal Abstraction in Reinforcement Learning with the Successor Representation. *arXiv preprint arXiv:2110.05740*.

Machado, M. C.; Bellemare, M. G.; Talvitie, E.; Veness, J.; Hausknecht, M. J.; and Bowling, M. 2018. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61: 523–562.

Meng, L.; Gorbet, R.; and Kulić, D. 2021. The effect of multi-step methods on overestimation in deep reinforcement learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 347–353. IEEE.

Metelli, A. M.; Mazzolini, F.; Bisi, L.; Sabbioni, L.; and Restelli, M. 2020. Control frequency adaptation via action persistence in batch reinforcement learning. In *International Conference on Machine Learning (ICML 2020)*, 6862–6873. PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Moskovitz, T.; Parker-Holder, J.; Pacchiano, A.; Arbel, M.; and Jordan, M. 2021. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped DQN. In *Advances In Neural Information Processing Systems 29*, 4026–4034.

Park, S.; Kim, J.; and Kim, G. 2021. Time Discretization-Invariant Safe Action Repetition for Policy Gradient Methods. *Advances in Neural Information Processing Systems*, 34.

Parr, R.; and Russell, S. 1997. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, 10.

Peer, O.; Tessler, C.; Merlis, N.; and Meir, R. 2021. Ensemble bootstrapping for Q-Learning. In *International Conference on Machine Learning*, 8454–8463. PMLR.

Peng, J.; and Williams, R. J. 1994. Incremental multi-step Q-learning. In *Machine Learning Proceedings 1994*, 226–232. Elsevier.

Precup, D. 2000. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Schoknecht, R.; and Riedmiller, M. 2002. Speeding-up reinforcement learning with multi-step actions. In *International Conference on Artificial Neural Networks*, 813–818. Springer.

Sharma, S.; Srinivas, A.; and Ravindran, B. 2017. Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017*.

Stolle, M.; and Precup, D. 2002. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, 212–223. Springer.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1): 9–44.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Thrun, S.; and Schwartz, A. 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, volume 6.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3): 279–292.

Watkins, C. J. C. H. 1989. Learning from delayed rewards.

Xia, L.; and Collins, A. G. 2021. Temporal and state abstractions for efficient learning, transfer, and composition in humans. *Psychological review*.

Zahavy, T.; Haroush, M.; Merlis, N.; Mankowitz, D. J.; and Mannor, S. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31.