Optimizing Attention and Cognitive Control Costs Using Temporally Layered Architectures

Devdhar Patel
devdharpatel@gmail.com
Manning College of Information and Computer Science, University of Massachusetts,
Amherst, MA 01003, U.S.A.

Terrence Sejnowski
terry@snl.salk.edu
Computational Neurobiology Laboratory, Salk Institute for Biological Studies,
La Jolla, CA 92037, U.S.A., and Institute for Neural Computation and Department
of Neurobiology, University of California, San Diego, La Jolla, CA 92093, U.S.A.

Hava Siegelmann
hava@umass.edu
Manning College of Information and Computer Science, University of Massachusetts,
Amherst, MA 01003, U.S.A.

The current reinforcement learning framework focuses exclusively on performance, often at the expense of efficiency. In contrast, biological control achieves remarkable performance while also optimizing computational energy expenditure and decision frequency. We propose a decision-bounded Markov decision process (DB-MDP) that constrains the number of decisions and computational energy available to agents in reinforcement learning environments. Our experiments demonstrate that existing reinforcement learning algorithms struggle within this framework, leading to either failure or suboptimal performance. To address this, we introduce a biologically inspired, temporally layered architecture (TLA), enabling agents to manage computational costs through two layers with distinct timescales and energy requirements. TLA achieves optimal performance in decision-bounded environments and in continuous control environments, matching state-of-the-art performance while using a fraction of the computing cost. Compared to current reinforcement learning algorithms that solely prioritize performance, our approach significantly lowers computational energy expenditure while maintaining performance. These findings establish a benchmark and pave the way for future research on energy and time-aware control.

1 Introduction

Deep reinforcement learning (DRL) has demonstrated a remarkable capacity for learning control policies (Fujimoto et al., 2018; Haarnoja et al., 2018; Mnih et al., 2015). However, existing efforts solely focus on maximizing predefined environmental rewards at a constant decision-making frequency. This approach contrasts with biological control, where organisms adjust their behavior and energy expenditure based on environmental demands. This adaptive method allows for more efficient resource use and can enhance performance in complex, fluctuating environments where defining reward functions can be challenging, and optimizing only the reward function can lead to unexpected behavior.

Biological control is significantly more efficient, flexible, and effective compared to current artificial control methods despite severe time delays (More & Donelan, 2018), slow rates of information transmission, and slow response times (Jain et al., 2015). These limitations can be ameliorated by integrating the capabilities of diverse components (Nakahira et al., 2021), distributing control over many layers (Li et al., 2023), and by incorporating multiple adaptive response times into these layers, as explored in this study.

State-of-the-art reinforcement learning (RL) algorithms currently lack the ability to adapt their time step size (Fujimoto et al., 2018; Haarnoja et al., 2018; Schulman et al., 2017). Typically, a constant time step is selected to avoid the problem of continuous optimization of the time step in every state. As a result, RL agents prefer extremely fast frequencies. However, different environments have different temporal contexts, each requiring a different time step size for an optimal performance-energy trade-off. Moreover, even within the same environment, the optimal time step can change. For example, when moving from a relatively safe and predictable state to an unpredictable state, the time step should decrease. Ideally, as the state transitions become more predictable with training and repetition, less sensing should be required and larger time steps become feasible. A fixed time step cannot effectively handle dynamically changing environments and is the cause of many failures in current algorithms.

The RL framework models the environment as a Markov decision process (MDP). For continuous environments, an agent acting at a fixed frequency must operate at least as fast as the situation that requires the fastest response. Such fast response speed means that the agent divides an episode into more states, resulting in a longer task horizon. This can decrease action-value propagation and, in turn, slow convergence to optimal performance (McGovern et al., 1997). Similarly, the temporal density of the reward (reward/state transitions per unit time) decreases as the agent becomes faster since the total state transitions increase, leading to an increase in the difficulty of the RL problem. This is especially true for environments with sparse rewards, where a faster agent would have to explore more

zero-reward (uninteresting) state transitions before finally reaching the goal or failure state that provides a nonzero reward. The mountain-car problem is one such example of this scenario (Moore, 1990).

A faster response time also means that the agent processes more inputs per unit time and needs faster actuation, both of which require more energy. In energy-constrained agents like robots, the agent's response speed can have a significant impact on total energy consumption. It is worth noting that DRL algorithms rely on experience replay memory during learning (Mnih et al., 2015), and that a faster response time would result in the creation of more memories per unit time. Consequently, a small memory size would also bottleneck the performance of fast-acting RL.

We primarily focus here on the effect of step size on energy conservation and performance. To investigate this formally, we introduce the decision bounded-MDP (DB-MDP), an extension of the MDP that constrains the number of decisions the agent can make in each episode. This constraint motivates the agent to conserve its decision-making energy expenditure. Fixed time-step RL will yield suboptimal solutions or even fail completely on DB-MDP.

To design an agent that can handle such environments, we take inspiration from biological systems. The design of the brain enables it to use various contexts, such as safety, energy availability, and performance-energy trade-off coefficient, to modulate its response time, ensuring accurate responses in both familiar and unfamiliar environments. This design allows for energy conservation in predictable situations, where slower reactions are acceptable while allowing for faster reactions in unpredictable situations. Recent work has shown that the brain uses distributed control to allow multiple independent systems to process the environment and react accurately (Nakahira et al., 2021), building on the history of research on the speed/accuracy trade-off (Heitz, 2014). This distributed control enables multiple layers of biological neural networks, with different delays and speeds, to activate and control muscle groups for executing complex behaviors. As a result, the brain and central nervous system can trade off between speed and accuracy, depending on the situation's demands.

Inspired by the biological design, we propose temporally layered architecture (TLA; see Figure 1): a reinforcement learning architecture that layers two different policy networks with different frequencies, allowing the RL agent to adapt its response frequency online by using their combination. To switch between the two policies optimally, we introduce a switch policy that is also trained using reinforcement learning. Training two different policies together to act on the same environment (albeit at different times) is a challenging multiagent task with no communication between the policies. However, we introduce an algorithm that can not only train all three networks (fast, slow and switch), together but also demonstrates competitive sample efficiency compared to single policy algorithms. To aid with the parallel training of the two networks, we introduce two different intrinsic
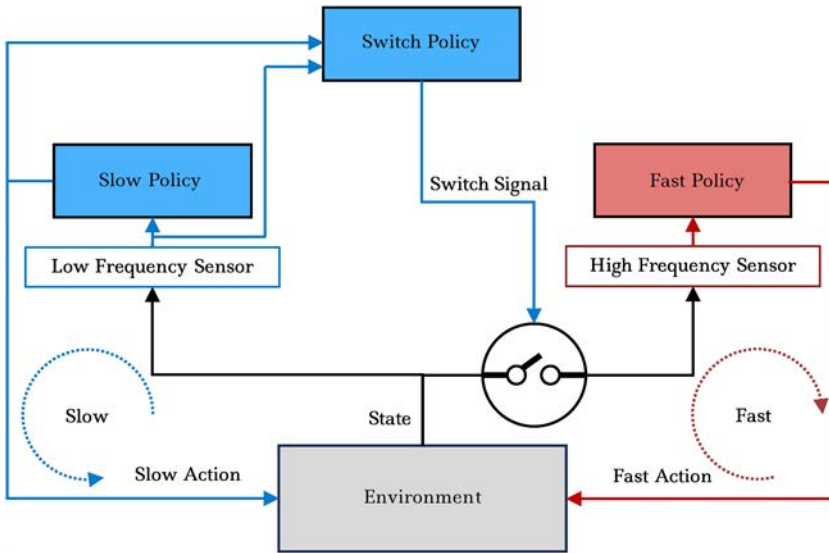
Figure 1: The temporally layered architecture (TLA) comprises two layers: the slow policy (blue) and the fast policy (red). The switch policy can activate or deactivate the fast policy, thus switching between the two layers. The reward given to each network is augmented differently with the energy and consistency penalty, which forces the overall policy to learn temporal abstractions from performance and energy-based contexts.

reward penalties: an energy penalty and a consistency penalty. The goal of the energy penalty is to encourage the use of the slow layer, aiding learning by adding a constraint that reduces the number of optimal policies to the most efficient ones. The consistency penalty is an intrinsic reward that encodes the inconsistencies between the actions picked by the slow and fast policies, enabling the policies to learn from each other by mimicking behavior.

1.1 Summary.

1. DB-MDPs extend the traditional MDP framework to include decision-making constraints. Traditional reinforcement learning algorithms are limited in their ability to explore and navigate DB-MDPs.
2. A novel biologically inspired temporally layered architecture (TLA) allows each layer to focus on a different temporal context and navigate the entire DB-MDP more effectively than classical RL agents.
3. The two layers of the TLA can be simultaneously trained with an efficient learning algorithm.

4. We empirically test decision-bounded environments for both tabular and parameterized policy on gridworld and continuous control environments and demonstrate that TLA succeeds where the state-of-the-art algorithms fail.

5. Empirical results on eight continuous control tasks demonstrate that TLA is able to achieve state-of-the-art performance with lower computational cost and fewer decisions.

2 Background

Our novel control architecture combining multiple controllers with different response is relevant to several subfields of AI.

2.1 Continuous Control. Continuous control refers to tasks that involve continuous actions. Compared to discrete control, exploration and learning for continuous control is more difficult and often requires a fast response frequency. Additionally, continuous control agents often need to control multiple joints and actuators.

We use the state-of-the-art twin-delayed deterministic policy gradient (TD3) algorithm for continuous control (Fujimoto et al., 2018). TD3 learns two Q-functions (critics) and uses the pessimistic value of the two for training a policy that is updated less frequently than the critics. Because TLA does not depend on the RL training algorithm, it can easily accommodate different training algorithms. For example, for gridworld environments, TLA is implemented using Q-learning on tabular policies.

2.2 Action Repetition and Frame Skipping. Reinforcement learning with a sequence of actions is challenging since the number of possible action sequences of length l is exponential in l. As a result, research in this area focuses on pruning the possible number of actions and states (Hansen et al., 1996; McCallum & Ballard, 1996; Tan, 1991). To avoid the exponential number of action sequences, some works have restricted the action sequences to repeating a single action. The number of actions is therefore linear in the number of time steps (Biedenkapp et al., 2021; Buckland & Lawrence, 1993; Kalyanakrishnan et al., 2021; Sharma et al., 2017; Srinivas et al., 2017). Frame skipping and action repetition have been used as a form of partial open-loop control, where the agent selects an action to be repeatedly executed without considering the intermediate states. TempoRL, introduced by Biedenkapp et al. (2021), learns an additional action-repetition policy that decides on the number of time steps to repeat a chosen action. This approach can lead to faster learning and reduce the number of action decision points during an episode. We use their approach as one of our benchmarks.

In their analysis of macro-actions, McGovern et al. (1997) identified two advantages: improved exploration and faster learning due to a reduced task horizon. Empirical evidence from Randløv (1998) shows that macroactions

also significantly reduce training time. Additionally, Braylan et al. (2015) showed that increasing the number of frames skipped can significantly improve the performance of the DQN algorithm (Mnih et al., 2015) on some Atari games.

However, these approaches require a predictable environment so that an action can be repeated safely without supervision. Furthermore, these approaches often require additional hyperparameter searching to find the best frame-skip parameter. In contrast, our approach can easily adapt its time-step size to changing conditions and the current predictability of the environment. In addition, it is less sensitive to the frame-skip parameter as it can switch between networks of different time steps.

In a similar vein, Yu et al. (2021) demonstrated a closed-loop temporal abstraction method in the continuous domain using an act-or-repeat decision after the action is picked, thus increasing action repetition. However, their approach requires two forward passes of the critic in addition to the actor and decision networks, as it uses the state-action value from the critic even after training. Thus, their approach does not reduce the number of decisions and is ill suited for the DB-MDP problems. Our approach (TLA) focuses on reducing the number of decisions and computing costs while increasing action repetition, making it well suited for bounded decision environments.

2.3 Residual and Layered RL. Recently, Jacq et al. (2022) proposed Lazy-MDPs where the RL agent is trained on top of a suboptimal base policy to act only when needed while deferring the rest of the actions to the base policy. They demonstrated that this approach makes the RL agent more interpretable as the states in which the agent chooses to act are deemed important. Similarly, for continuous environments, residual RL approaches learn a residual policy over a suboptimal base policy so that the final action is the addition of both actions (Johannink et al., 2019; Silver et al., 2018). Residual RL approaches have demonstrated better performance and faster training. Our approach is related to the residual approach, where a faster-frequency network is trained together with a slower-frequency base network to gain the benefits of macroactions and residual learning. However, unlike the residual approach, the final action for TLA is exclusively picked by a single network. While residual approaches rely on a pretrained base policy, the TLA demonstrates that both layers can be trained together. This is significant since TLA does not require a pretrained policy and yet can train both layers from scratch with competitive learning speeds to single-layered RL.

2.4 Options Framework. The options framework (Precup & Sutton, 2000) is a common framework for temporal abstraction in RL. Options are defined as 3-tuples $\langle \mathcal{I}, \pi, \beta \rangle$. Where $\mathcal{I}$ is the set of initiation states that defines in which states the option can start, π is the option policy that is

followed for the duration of the option, and β defines the probability of option termination in any given state. Options require prior knowledge about the environment to be defined. However, recent work has demonstrated that options that are automatically discovered by using the successor representation (Machado et al., 2021) or the connectedness graph (Chaganty et al., 2012) can help improve exploration and thus learning. In a similar vein of research, Dabney et al. (2021) demonstrated that temporally extended actions improve exploration. Our work takes advantage of this phenomenon by layering the fast network and slow network to gain the exploration benefit of extended actions along with the precision of the fast network. In TLA, both or either of the fast and the slow networks can be formulated as options policy. The slow network encodes stateless options for open loop control while the switch network can learn the initiation states $\mathcal{I}$ and the option termination probability β.

2.5 Multiagent Reinforcement Learning and Nonstationarity. Multiagent reinforcement learning (MARL) is an open problem with many challenges (Zhang et al., 2021). One of the main difficulties when training multiple agents is dealing with nonstationary environments (Padakandla, 2020). In an environment where multiple agents interact during training, the transition function for each agent is not constant because the outcome depends on the joint action of all the agents. As a result, traditional reinforcement learning approaches based on the assumption that the environment can be modeled as a stationary MDP- fail to solve MARL tasks.

TLA is a unique cooperative MARL task in which all agents learn to control the same body together with limited information shared by the agents. In cooperative settings, many strategies have been proposed to train agents together (Oroojlooyjadid & Hajinezhad, 2019). However, uniquely in our approach, we find that introducing energy constraints using differential intrinsic rewards induces cooperation and stable learning.

3 Decision-Bounded Markov Decision Process

The standard reinforcement learning setting (Sutton & Barto, 2018) involves an agent that can take actions to cause state transitions in the environment and in the process gain rewards. The environment is represented as a Markov decision process (MDP) (Puterman, 1990). The goal of the agent is to maximize the amount of reward it gets during a single episode. Reinforcement learning problems are defined as typically defined as a 6-tuple $(\mathcal{S}, \mathcal{A}, p, R, d_0, \gamma)$, where:

- $\mathcal{S}$: Set of all possible states in the environment
- $\mathcal{A}$: Set of all possible actions the agent can take
- p: Transition function that defines how the environment changes; hidden from the agent. $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$

- R: Reward function; Typically hidden from the agent. $R : S \times A \to \mathbb{R}$
- d_0: Initial state distribution. $d_0 : S \to [0, 1]$
- γ: Discount factor

The agent is characterized by a policy $\pi : S \times A \to [0, 1]$. The objective of the agent is to learn the optimal policy, π^8, that maximizes the expected sum of discounted rewards,

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t | \pi\right] := \mathbb{E}\left[G_t | \pi\right], \tag{3.1}$$

where R_t is the reward at time t and G_t is the return from time t.

Here, the agent and the environment are defined as two distinct entities that can interact solely through actions. However, this setting does not capture the internal state of the agent, which can also be modeled as an MDP. For example, during the modeling of a battery-powered robot, if two different robots start in the same state of a deterministic environment and take the same actions but their algorithms consume different amounts of energy at each step, the state of the environment will be the same, yet it will not reflect the differing remaining charge in their batteries. Hansen et al. (1996) proposed a control setting where sensing incurs a cost, thus allowing the agent to perform a sequence of actions to reduce the sensing cost. However, similar to the MDP, it does not capture the computation cost, as algorithms that maintain an accurate model of the environment can easily reduce their sensing cost by increasing the computation cost.

Thus, to capture this complexity, we introduce decision-bounded MDP (DB-MDP) that puts a bound on the number of decisions that can be taken by the agent in each episode. Decisions are defined as a sequence of n actions where $n \in \mathbb{N}$. We note that in time-limited tasks, simple agents that make one decision at each step ($n = 1$) can be modeled as time-dependent MDPs (Pardo et al., 2018). Time-dependent MDPs can be thought of as a stack of T time-independent MDPs followed by a terminal state. Therefore, at each time step, the actions result in transitioning to the next state in the MDP in the next stack. However, the reachable states in the next stack are the same as the states reachable from the current state in the original MDP.

When agents can plan a sequence of actions or repeat the same action ($n \geq 1$), it is possible to transition to states that are not adjacent in the original MDP, thus preserving the number of remaining decisions (see Figure 2). Thus, we can see that DB-MDPs are an extension of time-limited MDPs that dissociate the decisions from time.

Decisions are not formally defined in the RL framework, yet they play an integral role in control. Typically, decision making incurs a cognitive cost, and commonly repeated actions become more efficient over time in the brain as fewer decisions are required (Wiestler & Diedrichsen, 2013).

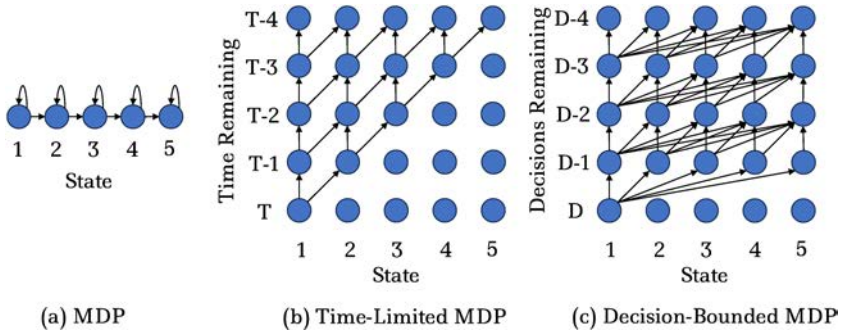(a) MDP   (b) Time-Limited MDP   (c) Decision-Bounded MDP

Figure 2: (a) A simple MDP with $S = 5$ states. Each state has two actions, one that leads to the next state and one that results in the same state change. (b) Time-limited MDP. In the time-limited MDP setting, there is an additional limit on the amount of time available (T). The MDP thus is expanded to $S \times T$ states. (c) Decision-bounded MDP. In decision-bounded MDP, the number of decisions is limited. However, a single decision can result in multiple planned actions. Similar to time-limited MDP, there are $S \times D$ states where D is the number of available decisions. However, a larger part of the MDP is reachable if the agent is able to take multiple actions per decision, resulting in cognitive cost reduction.

Thus, tasks like walking, while complex, do not require many decisions or much cognitive load. Formally, in this work, we define a decision as each time a single state is taken as an input to produce one or more actions. We note that this does not fully capture the cognitive energy expended behind each decision, which may be variable. However, most of current state-of-the-art algorithms incur the same amount of cognitive energy regardless of the complexity of the decision. Thus, decisions are directly proportional to the cognitive energy expended in such cases. Additionally, decisions also measure the amount of time between each forward pass. More decisions also mean that the agent has less processing time between each decision. Finally, decisions also capture the predictive power of each algorithm. Fewer decisions also mean that the agent can predict a longer action sequence. Thus, complementary to performance, decision is a versatile metric for the overall "goodness" of the algorithm.

4 Temporally Layered Architecture

This section discusses various methods for achieving temporal adaptivity in reinforcement learning. We introduce our novel architecture and its accompanying learning algorithm, which learns the two distinct temporal abstractions simultaneously and switches between them to optimize both performance and efficiency.

4.1 Temporal Adaptivity. In control tasks, different states require different levels of temporal attention. Some states are unpredictable, resulting in higher entropy for the transition function $p(s_{t+1}, a_t, s_t)$. In these transitions, increased supervision is required to monitor and correct any undesired transitions so that the expected reward does not decrease after the action is taken. On the other hand, some states are predictable and have lower entropy for the transition function. In these states, since the outcome is expected and predictable, the agent can take more time before sampling input from the environment. The brain takes advantage of this phenomenon by reducing attention in familiar states while increasing it in unfamiliar or unpredictable states (Del Giudice et al., 2014; van Helden & Naber, 2023). The primary benefit is to reduce the energy required for computation when it does not affect performance, thus increasing efficiency.

In the context of RL, temporal adaptivity refers to changing the time step t based on the state. However, this is a nontrivial task because for any given policy π, its expected sum of rewards $J(\pi)$ depends on the time step t. This is because the reward gained from the environment for performing an action a might change based on how long (t) it is performed in the environment.

4.2 Temporally Adaptive Reinforcement Learning. A naive approach to adding temporal adaptivity is to treat each action-time step pair as a different action. The action space is augmented to include the time step; thus the policy is $\pi : \mathcal{S} \times (\mathcal{A} \times \mathcal{T}) \to [0, 1]$. However, this is undesirable as it makes the policy intractable due to the exponential increase in the possible number of actions.

To overcome this issue, many approaches have been proposed that focus on increasing action repetition, as noted in section 2.2. Among them, the most successful in reducing the number of decisions is TempoRL (Biedenkapp et al., 2021). TempoRL uses two networks: one to select an action and another to determine how long that action should be performed in the environment. However, they do not impose additional constraints or penalties to incentivize longer actions. Additionally, the actions are optimized for a single time step, so in situations where the optimal extended action differs from the optimal single-step action, TempoRL will not be able to learn the extended action.

4.3 Temporally Layered Architecture. We draw inspiration from the brain and biological reflexes, which use multiple layers of computation with different latencies to enable temporal adaptivity. TLA has two layers, slow and fast, that learn two policies, each with a different step size, π^s and π^f, where s and f denote the slow and fast layers, respectively. The fast layer is similar to traditional RL agents and can observe and act at every time step, whereas the slow layer can only observe and act every τ time steps, where $\tau \geq 2$ and $\tau \in \mathbb{Z}$. Therefore, for any $t \bmod \tau$ that is equal to 0, the next action

is sampled from the slow policy (see equation 4.1), and the previous action is repeated otherwise.

To switch between these two policies, we introduce a switch policy that decides whether to activate the fast network based on the state and the slow action. Therefore, at each time step:

$$a_t^s = \begin{cases} a_{t-1}^s & \text{if } t \bmod \tau \neq 0 \\ \sim \pi^s(\cdot \mid s_t) & \text{otherwise,} \end{cases} \tag{4.1}$$

$$g_t = \mu^g(\cdot \mid s_t, a_t^l), \tag{4.2}$$

$$a_t^f \sim \pi^f(\cdot \mid s_t), \tag{4.3}$$

$$a_t = a_t^s \cdot (1 - g_t) + a_t^f \cdot g_t, \tag{4.4}$$

where s_t is the state at time t, a^s is the slow action, a^f is the fast action, and $g \in \{0, 1\}$ is the switch action. π^s, π^f, and μ^g are the slow, fast, and switch policies, respectively. Thus, the fast network is only activated when $g = 1$.

Thus, during an episode, the agent might use both layers depending on the context. The value of any given state is dependent on both the fast and slow policies, making it is difficult to simultaneously optimize both layers. Each layer is unaware of the other layer's policy and thus needs to navigate in a nonstationary environment. To aid training, experiences are added to the replay memory of both the slow and the fast networks whenever either network is activated. This is straightforward for the fast network, as it has τ experiences with the same action created whenever a slow action is chosen. However, the slow network can only observe its own actions. When the fast network is activated, we augment the slow reward with a consistency penalty that captures the difference between the slow and the fast actions, facilitating the sharing of information between them. Additionally, the rewards of the slow and switch networks are augmented by the energy penalty to incentivize slow actions. Hence, the final rewards for each network are as follows:

$$R_t^f = R_t - g_t \cdot ((a_t^s - a_t^f)/a_{\max}) \cdot j, \tag{4.5}$$

$$R_t^s = R_t - p \cdot g_t - g_t \cdot ((a_t^s - a_t^f)/a_{\max}) \cdot j, \tag{4.6}$$

where j is the consistency penalty parameter and p is the energy penalty parameter that incentivizes slow actions. Thus, even though the action a^s is not performed, it can affect the reward, aiding the training of the slow network. R_t^q is the reward provided to the fast network at each step, while the slow and the switch network receive the sum of rewards for τ steps: $\sum_{k=t-\tau}^{t} R_k^l$. This formulation allows the use of a single hyperparameter $j = p$ for simpler environments with consistency and energy penalties. For multidimensional environments that already have an action magnitude penalty, j is set

to zero, and only p needs to be searched. Additionally, for multidimensional environments, since j is set to 0 when the fast network is activated, the slow action has no influence on the reward received or the next state. This increases the relative nonstationarity of the slow network. To avoid this, we set the slow action to zero every time the fast action is picked for the environments with multiple action dimensions. Thus, all nonstationary memories in the replay memory of the slow network have zero action while all nonzero action memories will be stationary.

The slow and fast networks are trained using the TD3 algorithm, while the switch policy is trained using deep Q-learning (Mnih et al., 2015). For completeness, the pseudo-code and an alternate architecture diagram with reward penalties are presented in the appendix.

5 Experiments

We evaluate the performance of TLA on various environments including decision-bounded, gridworld, and continuous environments. Additionally, we open-source our code[1] and compare TLA to three benchmark algorithms:

1. **Base algorithm:** TLA can use any model-free algorithm. In this work, we use Q-learning for gridworld environments and TD3 for continuous control. Therefore, we also compare TLA to the original algorithm that uses a constant time step. In environments without a decision bound, this algorithm provides an upper bound on reward since it represents the state of the art.
2. **Extended-action base algorithm:** Since the slow layer of TLA uses a larger time step, we also evaluate the base algorithms with larger constant time steps. This algorithm provides a lower bound on the decisions for TLA as it always picks the slow time step.
3. **TempoRL:** To our knowledge, we are the first to introduce an algorithm that optimizes the dual objectives of performance and decisions in continuous control environments. However, we provide the TempoRL algorithm as an additional comparison, as it uses action repetition as a means of further optimizing performance and, in the process, reduces the number of decisions.

5.1 Decision-Bounded Environments.

5.1.1 Discrete Decision-Bounded Environment (Gridworld). Since many aspects of reinforcement learning are linked to the time step, temporal adaptivity allows TLA to focus on multiple objectives and gain an advantage

---

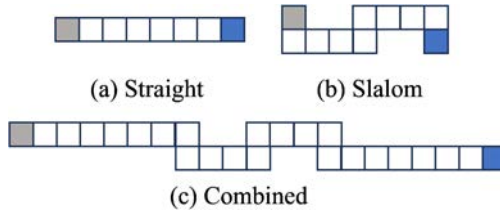[1]https://github.com/dee0512/Temporally-Layered-Architecture.

Figure 3: Gridworld environments. The gray box represents the starting state, and the blue box is the goal state.

in performance, learning speed, and the number of decisions in decision-bounded environments. To demonstrate this, we introduce three gridworld environments, each suited to a different optimal step size: straight, slalom, and combined (see Figure 3). The straight environment consists of a straight corridor of length 30. This environment can be easily solved by repeating a single action; thus it is easy to reduce the number of decisions in this environment. The slalom environment consists of long horizontal corridors connected with short vertical turns where action repetition is suboptimal. Finally the combined environment, combines the straight and slalom, so that the episode starts and ends with straights and has slalom in between. The environments are deterministic with four available actions to move in one of the four directions. At each environment transition, the agent receives a reward of -1. Upon reaching the goal, the episode ends. However, if the agent runs out of decisions before reaching the goal, the agent receives a large penalty of -50 reward for straight and slalom and -100 for combined. The choice of a very large negative reward is chosen solely to distinguish between agents that run out of decisions and agents that do not. The number of decisions is limited to 15 for straight and slalom and 60 for combined.

We use Q-learning and tabular policies for evaluation and compare three algorithms. Q-learning is a standard policy that takes one decision per time step. Extended-action Q-learning repeats each action four times, thus reducing the number of decisions per unit of time by a factor of four. Finally, TLA can switch between taking one decision per time step and repeating an action four times. All values in the tabular policies are initialized to 0. Since in the tabular policy, the consistency penalty cannot be implemented, j is set to 0 while p is set to 1 (see equations 4.5 and 4.6).

We tested each policy on each environment over 20 independent trials. Figure 4 shows the average reward and average number of decisions versus training episodes. When optimizing the dual goals of reward (performance) and decisions, priority is given to the reward to maintain competitive performance compared to decision-agnostic algorithms.

In the straight environment, TLA quickly explores and reaches the goal and then optimizes the number of decisions. Curiously, it converges in
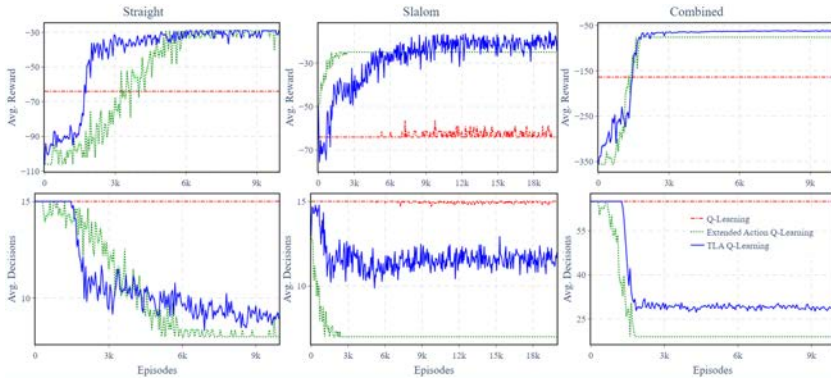
Figure 4: Decision bounded gridworld environments. TLA (blue) achieves the optimal performance with the fewest required decisions. All results are averaged over 20 trials. Top: Average reward versus episodes during training. Bottom: Decisions versus episodes during training.

performance faster than the extended action Q-learning even though the environment is ideal for action repetition. On the other hand, it is slow to converge toward the optimal decisions as it prioritizes performance over efficiency. Finally, Q-learning fails to solve the environment as there are fewer decisions allowed than the distance to the goal (see Figure 4).

In the slalom environment, the optimal path length is equal to the decision bound; thus, it is possible for Q-learning to solve the environment. However, the decision bound makes exploration for Q-learning difficult. Extended-action Q-learning cannot reach optimal reward as it cannot navigate the sharp turns effectively. TLA is the only algorithm that reaches optimal reward. We see that TLA needs to take more decisions than extended action Q-learning. This is because it prioritizes reward over decisions. However, this behavior can be modulated by changing the energy penalty p.

Similarly, in the combined environment, optimal performance requires fast actions for the sharp turns and extended action elsewhere. In conclusion, TLA successfully switches between the two to achieve the perfect trade-off between performance and decisions so that it finds the optimal performance with the fewest possible decisions when acting at two timescales.

5.1.2 Continuous Control. Next, we evaluate the performance and decisions of four agents—TLA, TempoRL, TD3, and TD3 extended action (TD3-EA)—on two decision-bound continuous action environments: LunarLanderContinuous-v2 and MountainCarContinuous-v0. We modified the Gym environments (Brockman et al., 2016) to create decision-bounded environments. The number of decisions was constrained to 70 and
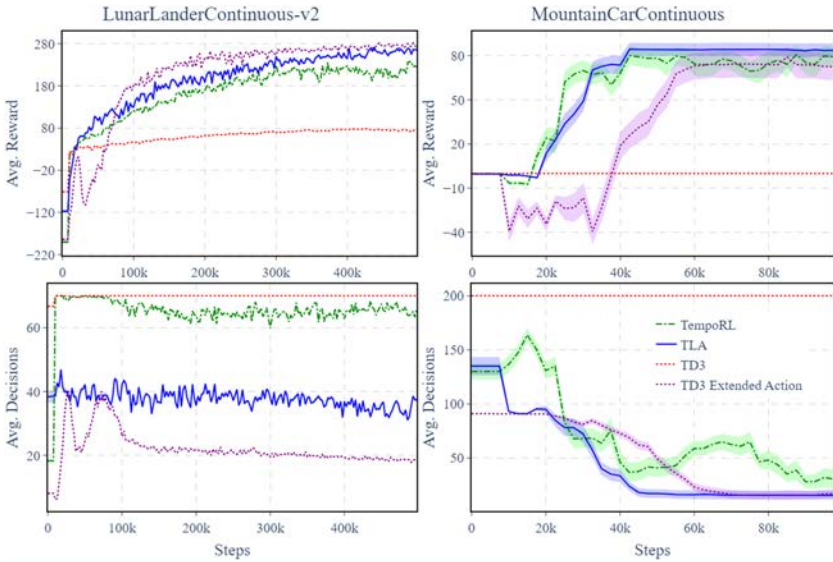
Figure 5: Decision bounded continuous control environments. Top: Average reward versus training episodes. Bottom: Decisions versus training episodes. All results are averaged over 10 trials. The shaded region represents standard error. Left: In the Lunar Lander environment, which is robust to action repetition, TD3 extended action (TD3-EA) shows superior performance. In this environment, it takes longer for TLA to converge toward optimal average reward, and thus decisions are not properly optimized during the training period as TLA prioritizes reward over decisions. Yet TLA outperforms other algorithms and is able to successfully solve the environment under the decision constraint. Right: Due to the longer step size, TLA, TD3-EA, and TempoRL are able to successfully solve the Mountain Car task. However, TLA achieves better performance than TempoRL and TD3-EA while achieving the lower bound on decisions represented by TD3-EA.

200 for the Lunar Lander and Mountain Car, respectively. Unlike the gridworld environments, there is no negative reward when the agent runs out of decisions.

The maximum skip lengths for TempoRL, action repetition for TD3-EA, and τ for TLA are set to be the same for a fair comparison. The τ is set to 12 for Lunar Lander and 11 for Mountain Car. Figure 5 shows the learning curves for rewards and decisions. All results are averaged over 10 trials. We find that the Lunar Lander environment is well suited for extended actions. TLA, TempoRL, and TD3-EA achieve comparable performance on Lunar Lander. While TempoRL is not able to reduce the number of decisions effectively, TLA achieves the lowest decisions without forced extended action.

We found that the Lunar Lander environment is uniquely robust to large time steps and therefore TD-EA achieves superior result. However, as we demonstrate in the following section, extended action algorithms are not suitable for most environments.

TD3 cannot solve the Mountain Car environment due to inefficient exploration. We see that TLA is simultaneously able to optimize average reward and decisions, thus outperforming all algorithms. While TempoRL is not able to optimize the number of decisions effectively, TLA achieves the fastest convergence to the highest performance with the fewest decisions.

5.2 Decision-Unbounded Continuous Control Environments. We evaluate TLA on a suite of eight continuous control environments using the OpenAI gym library (Brockman et al., 2016): two classic control problems and six MuJoCo environments (Todorov et al., 2012). We selected tasks on which TD3 has demonstrated state-of-the-art performance to highlight the inefficiency of the policies learned by current state-of-the-art algorithms. We set the time step of the fast network to be equal to the default step size of the environment so that TLA has the same step size as TD3 and TempoRL. For the TempoRL algorithm, the max skip length J was set to be equal to τ, so the longest action repetition possible is the same for both TLA and TempoRL. Additionally, for reference, we also present the TD3-EA results. However, continuous control tasks are especially difficult to learn using extended actions, and we find that TLA uses both the fast and the slow layer for almost every task. Therefore, for a fair comparison, we set the time step of TD3-EA to be approximately equal to the average decision-step size of TLA after training.

The algorithms' hyperparameters and neural network sizes were kept the same as in previous work (Fujimoto et al., 2018). The maximum training steps were set to 30,000 for the Pendulum-v1 environment and 100,000 for MountainCarContinuous-v0. The rest of the environments were trained until 1 million steps. The initial exploration steps were set to 1000 for Pendulum-v1, InvertedPendulum-v2, and InvertedDoublePendulum-v2; 10,000 for MountainCarContinuous-v0; and 20,000 for Hopper-v2, Walker2d-v2, Ant-v2, and HalfCheetah-v2. A complete list of hyperparameters is included in the appendix.

For each environment, a hyperparameter search for τ and p was conducted over 5 random seeds. The final results presented are averaged over 10 random seeds. The hyperparameter search for τ was limited to a maximum of 11, and p was evaluated over the range [0.1, 6]. Note that for different environments, the average reward per time step varies, and therefore, the optimal value of p also varies with it. The environments with multidimensional actions (Hopper-v2, Walker2d-v2, Ant-v2, and HalfCheetah-v2) have a control cost included in their rewards, which is similar to the consistency penalty. Thus, for those environments, $j = 0$. For the rest of the environments, $j = p$ for simplicity (see equations 4.4 and 4.5).

We note that learning extended actions for multidimensional actions is more difficult than for single-dimensional actions as it enforces the repetition syncing across all the dimensions. Therefore, we split the results into single-action dimension and multiple-action dimensions.

5.2.1 Single-Action Dimension. We evaluate on four environments: Pendulum, Mountain Car Continuous, Inverted Pendulum, and Inverted Double Pendulum. Figure 6 shows the average reward and decisions for the four algorithms. We see that TLA outperforms all algorithms in all four environments (prioritiziing high reward, then lower decisions). On the Mountain Car environment, TLA uses more decisions than TempoRL and TD3-EA to achieve greater rewards. In the rest of the environments, TLA achieves optimal performance while achieving the lowest number of decisions. TD3-EA, however, fails to learn the Pendulum tasks and thus results in very few decisions before the episode ends. We find that on the Inverted Pendulum task, after training, TLA almost never activates the fast network, resulting in an optimal policy with a time step that is 10 times larger. Yet this policy cannot be learned by standard RL algorithms that only use a single layer, as evidenced by TD3-EA that has the same time step.

5.2.2 Multiple-Action Dimensions. To demonstrate the scalability of TLA, we also present results on four difficult continuous control problems that are ill suited for action repetition: Hopper, Walker2d, Ant, and Half Cheetah. In multidimensional environments, action repetition forces all actuators to repeat the same action for the same amount of time in a synchronous manner. This results in a suboptimal policy, as repetition syncing is almost never the optimal behavior.

Figure 7 presents the reward and decision learning curves for the four algorithms. Surprisingly, TLA outperforms TD3 on the Hopper environment despite the challenge of repetition syncing. Additionally, in all environments tested, TLA achieves comparable performance with fewer decisions. TempoRL however, cannot scale to the difficult environments of Ant and Half Cheetah.

5.3 Energy, Action-Repetition, and Jerk. The number of decisions gives an overall picture of many important underlying metrics like cognitive cost, actuation cost, and reaction time. However, while generally it is beneficial to reduce the number of decisions, it results in different scaling of individual metrics for different algorithms. For example, it might be possible to employ an accurate model of the environment and plan a sequence of actions using online planning to reduce the number of decisions, but this is a prohibitively costly strategy in terms of cognitive cost.

Thus, we measure four additional different metrics that are affected by reduced decisions and macroactions:
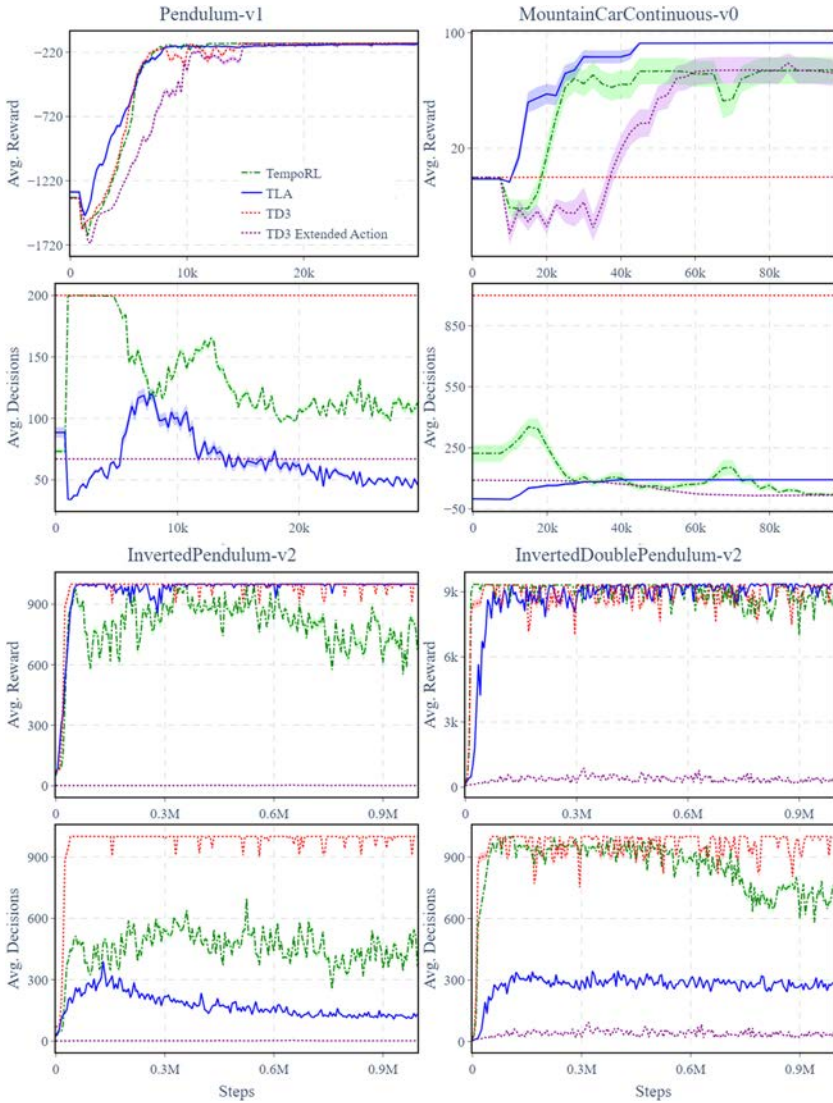
Figure 6: Average reward and average decisions during training. TLA (blue) achieves state-of-the-art performance using a fraction of the decisions.

1. Computation cost: The computation cost is dependent on the actual neural network architecture and the algorithm employed. Since the algorithms we evaluate have different architectures with different numbers of parameters, the cognitive cost can vary significantly. TLA
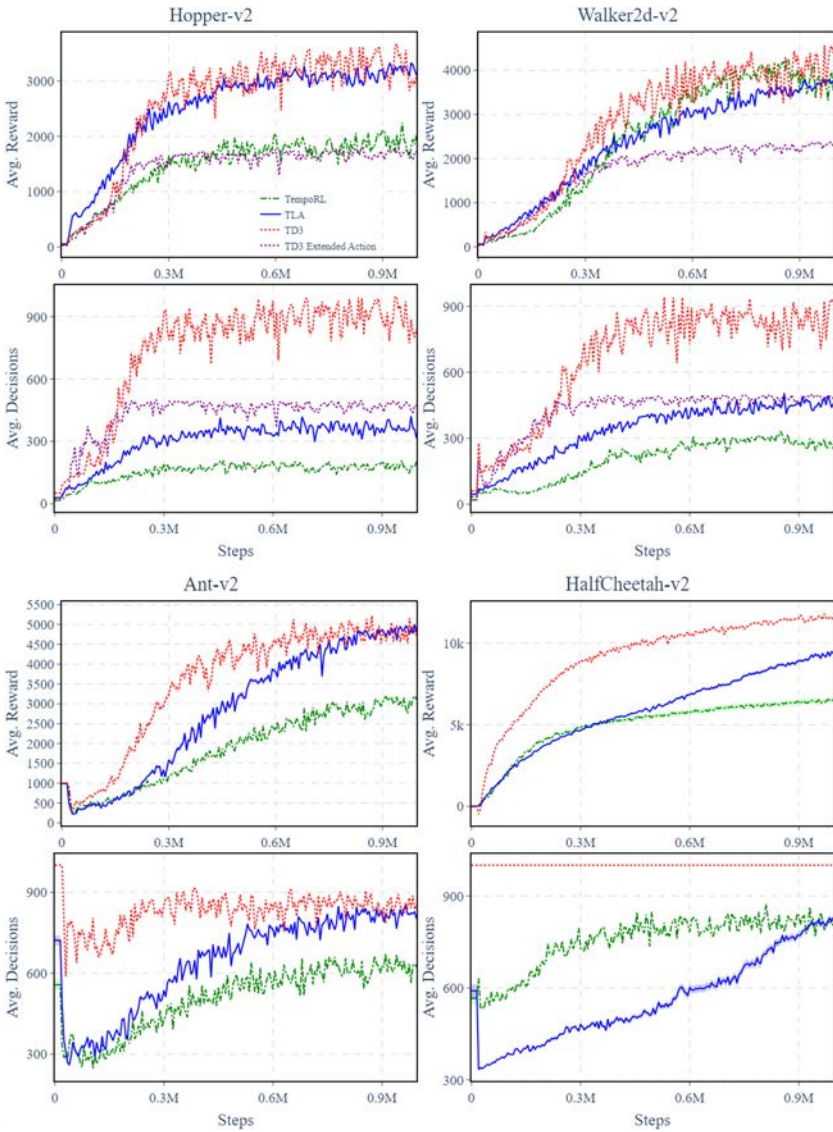
Figure 7: Average reward and average decisions during training on environments with multiple action dimensions. Multiple action dimensions are not well suited for macroactions that are action repetitions yet TLA (blue) achieves comparable performance using a fraction of the decisions.
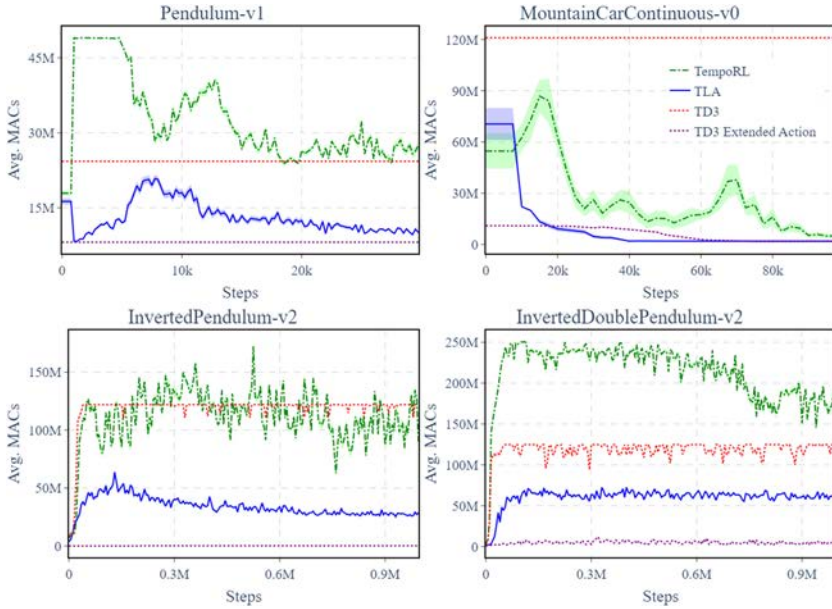
Figure 8: Average multiply-accumulate operations (MACs) versus training. TLA (blue) uses only a fraction of the computing cost even though it has roughly 3 times the parameters as TD3 and 1.5 times the parameters.

has roughly three times more parameters since it uses three policies (fast, slow, and switch), and yet it demonstrates that it uses a fraction of the computation cost compared to other algorithms. While TempoRL reduces the number of decisions, it often has a higher computational cost. Figure 8 demonstrates the average Multiply-Accumulate operations (MACs) per episode versus training steps for all algorithms.

2. Action repetition: In real-world tasks, there is often latency and communication cost involved between action selection and actuation. In such applications, increasing action repetition can reduce the amount of communication required since the same action can be repeated until a new action or directive is received. Therefore, we measure action repetition percentage as the average percentage of time steps in an episode where the previous action was repeated. For multidimensional actions, we calculate action repetition individually over each dimension before averaging, as each action dimension represents a different actuator that requires a separate channel of communication. Unsurprisingly, TLA and TempoRL have significantly higher action repetition across all environments. We provide detailed results in the appendix.

Table 1: Number of Environments (out of Eight) Where the Evaluated Algorithm Demonstrated the Best Performance on the Given Metric.

Metric	TD3	TempoRL	TLA (Ours)
Average reward	**7**	3	**7**
Normalized AUC	**5**	0	3
Action repetition	0	1	**7**
Jerk	1	1	**6**
Decisions	0	1	**7**
Compute cost (MACs)	2	1	**5**

Notes: For the Average Reward, all values within 1 standard deviation of the best performance are accepted. TLA demonstrates a well-rounded policies when compared to other algorithms.

3. Jerk: The single most important metric in continuous control is jerk. The motions of the human body minimize jerk in their behavior to reduce joint stress and energy cost (Voros, 1999). Thus, it is desirable to reduce jerk in the control task as it reduces energy expended during actuation and lowers the risk of damage or wear to the actuators (Tack et al., 2007). We measure jerk as the difference in action magnitude per time step, as each action represents the torque or force applied. We find that TLA reduces jerk in all but one environment. We provide detailed results in the appendix.

4. Area under curve (AUC): The area under the average reward training curve is used in previous work to measure the convergence speed of RL algorithms (Biedenkapp et al., 2021). One benefit of using action repetition and macroactions is that the agent can explore the environment more effectively and thus converge to optimal performance quickly. However, we find that this is not always the case in continuous action spaces. TempoRL demonstrated better AUC in discrete environments, yet when tested in continuous environments, it fails to converge faster than TD3. Yet, TLA, despite training three different policies in parallel, is able to outperform TD3 in three of the eight tested environments.

We demonstrate that environments can have multiple solutions (optimal policies), and they are difficult to distinguish solely based on performance (average Reward). Table 1 summarizes the evaluation of all the algorithms on all the metrics tested. We note that TD3-EA is not incuded in this table since it is not a different algorithm, rather just TD3 with the time step changed, and it reaches acceptable performance on only two out of the eight environments. For each RL algorithm, since the first task is to find the optimal policy, we first measure the average reward. To reduce the effect of randomness, we accept all values that are within 1 standard deviation of

the best reward. Additionally, for the inverted double pendulum environment, we accept all solutions since it is designed to have a very high reward while the optimized policies display a very low standard deviation (detailed results are in the appendix).

Finally, for every environment where the algorithm reaches a competitive solution as described above, we further test it on six different metrics: AUC, action repetition, jerk, decisions, and MACs. We find that TLA reaches a competitive performance in seven out of the eight environments (all except Walker2d). Furthermore, it also demonstrates that the policies learned by TLA are well rounded and demonstrate superior performance when measured on other metrics. Importantly, it demonstrates higher action repetition and fewer decisions on all seven environments. Additionally, it demonstrates lower jerk in all but one environment and lower compute cost in all environment except the Ant and Half Cheetah, where it suffers from repetition syncing.

We provide more detailed results on each metric in the appendix.

6 Conclusion

This letter focuses on a new biologically inspired method to make RL more practical in realistic environments. Learning enhances layered control architectures (Li et al., 2023) and optimizes the speed-accuracy trade-offs found in biological control systems (Nakahira et al., 2021). We first introduce the decision-bounded MDPs and demonstrate that state-of-the-art RL approaches are unable to optimize their strategies, and frequently they completely fail. Our temporally layered architecture (TLA) is able to change the number of decisions and computing based on realistic needs and thus can effectively solve problems in decision-bounded environments. We then tested the TLA in decision-unbounded environments and showed its superiority there as well: TLA is able to find more efficient solutions due to its temporal adaptivity. Additionally, we show how optimizing the dual objectives of performance and energy results in a reduced jerk, producing more natural and safer control.

Our new architecture, the TLA, achieves temporal awareness by allowing the agent to choose between two different policies that make up its two layers. The slow layer allows TLA to plan a sequence of actions without intermediate sampling from the environment. Thus, the slow layer operates at a higher latency using a partially open-loop control where the next action in the sequence of actions does not depend on the next state sampled of the environment. On the other hand, the fast action acts similarly to the traditional reinforcement agent and is reflexive (reactive) in nature. It is a closed-loop system where each action depends on the state resulting from the previous action. A third network helps in achieving the switch between these two, and together, the layered system mimics the biological

control achieved by the spinal cord and the brain (Weiler et al., 2019). Similar parallel pathways are also present inside the brain that allows it to change between planning and reflex depending on the situation (Nakahira et al., 2021). A look at the architecture of the brain demonstrates that it is designed in this layered manner. The brain has connections to the superior colliculus from almost every other area (Harting et al., 1992). The superior colliculus is a site of sensorimotor integration responsible for motor control. This might allow the brain to spend variable amounts of computing resources and change its latency while picking actions. We note that this idea is also related to the idea of early exits in deep neural networks (Patel & Siegelmann, 2022; Scardapane et al., 2020); however, our approach proves that it is better suited for control where the difference in computing and latency has a significant impact. Furthermore, recent work has also shown evidence for parallel multi-timescale learning, similar to TLA, in the brain that might explain the discrepancies in the dopamine activations of the brain and the TD error after learning (Masset et al., 2023).

We can perceive TLA as a novel control framework where three agents control the same body. This is different from traditional multiagent approaches where each agent generally controls a different body or a different set of actuators. Often in multiagent reinforcement learning, all the agents act at the same frequency and are often synced with the environmental time step. TLA offers an alternative approach where the agents act at different frequencies and thus are suited for different optimization goals for the same task. TLA uses time as a distinguishing factor to assign different goals and create a natural hierarchy between the agents. We plan to explore this multiagent paradigm, especially in war games where agents act in different hierarchies and timescales.

The main limitation of TLA is that it can plan only a sequence of actions consisting of a single action. Therefore, the benefits of TLA are limited to multidimensional actions. In future work, we plan to implement a slow layer that can plan a sequence of actions instead of repeating the same action. In TLA, the fast layer acts as a reflex that only activates when needed, while the slow layer acts as a planning network in the brain. However, even within the brain, temporal attention is adaptable (Morillon et al., 2016). Thus, we will also explore making the step size of the slow layer adaptable to allow for changes in the planning horizon, while the fast network will remain for unexpected situations. This will be tested for improvements in the Walker2d environment.

Finally, beyond what was shown here, TLA may be most suitable for use in energy-constrained environments, environments that require a distributed approach, and environments with high communication costs or delays, such as drones and robotic systems. TLA sets a benchmark for decision and energy-constrained environments and paves the way for future research in time and energy-aware artificial intelligence.

Appendix

A.1 Implementation Details. All experiments were performed on a GPU cluster with the following GPUs: Nvidia 1080ti, Nvidia TitanX, and Nvidia 2080ti.

A.2 Hyperparameters. The hyperparameters used for all our experiments for the TD3 algorithms are given in Tables 2 and 3.

Table 2: List of Common Hyperparameters.

Hyperparameter	Value	Description
Exploration noise	0.1	Standard deviation of the gaussian noise added to actions
Batch size	256	Batch size for learning
Discount factor	0.99	Discount factor
τ	0.005	Update rate for the target network in TD3
Policy noise	0.2	Noise added to target policy during critic update
Noise clip	0.5	Range to clip the target policy noise
Policy frequency	2	Delay factor for policy update
Replay buffer size	1e6	Size of the replay buffer

Table 3: List of Environment-Specific Hyperparameters.

Environment	j	p	τ	Start t	Max T	Evaluation Frequency
Pendulum-v1	1	1	6	1000	30,000	250
MountainCarContinuous-v0	1	1	11	10,000	100,000	2500
InvertedPendulum-v2	0.5	0.5	10	1000	1,000,000	5000
InvertedDoublePendulum-v2	4.5	4.5	5	1000	1,000,000	5000
Hopper-v2	0	1	9	20,000	1,000,000	5000
Walker2d-v2	0	0.5	7	20,000	1,000,000	5000
Ant-v2	0	0.2	3	20,000	1,000,000	5000
HalfCheetah-v2	0	4.5	3	20,000	1,000,000	5000

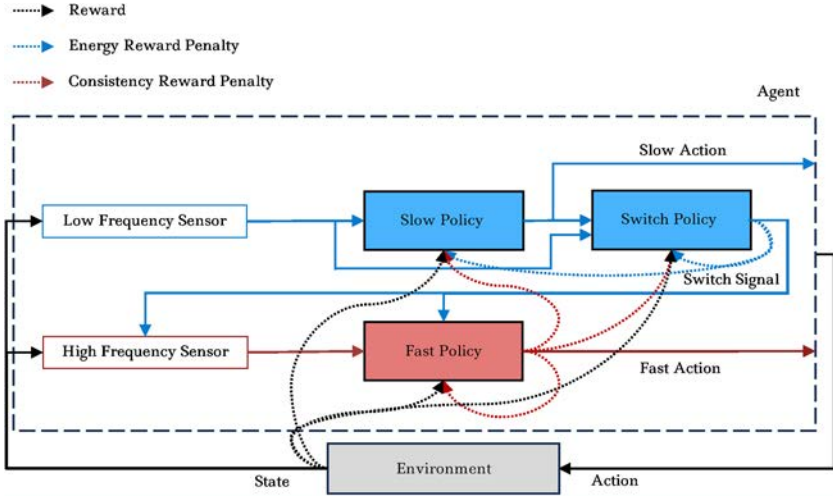**A.3 TLA Architecture.** Figure 9 provides a detailed TLA architecture.

Figure 9: The TLA architecture has two layers: slow (blue) and fast (red). It has three different RL agents that are trained in parallel: slow, fast, and switch. Each network receives a different reward penalty, resulting in the optimization of energy in addition to the performance.

A.4 TLA Algorithm.

Initialize Slow network with critics $Q^l_{\theta_{l1}}, Q^l_{\theta_{l2}}$ and actor $\pi^l_{\phi_l}$

Initialize Fast network with critics $Q^q_{\theta_{q1}}, Q^q_{\theta_{q2}}$ and actor $\pi^q_{\phi_q}$

Initialize Gate network with Q-function Q^g_θ

Initialize target networks $\theta'_{l1} \leftarrow \theta_{l1}, \theta'_{l2} \leftarrow \theta_{l2}, \phi'_l \leftarrow \phi_l, \theta'_{q1} \leftarrow \theta_{q1}, \theta'_{q2} \leftarrow \theta_{q2}, \phi'_q \leftarrow \phi_q$

Initialize replay buffers $\mathcal{B}_l, \mathcal{B}_q, \mathcal{B}_g$

for $t = 1$ **to** T **do**

  if $t \bmod \tau$ **then**

    select slow action with exploration noise $a_l \sim \pi^l(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$

    select gate action with epsilon greedy policy $a_g \sim \pi^g(s, a_l)$

    if $a_g = 0$ **then**

      $r_l = 0$

      $r_g = 0$

      $s_l = 0$

    else

      $r_l = -p * \tau$

      $r_g = -j * \tau$

        end if
      end if
      if $a_g = 1$ **then**
          select fast action with exploration noise $a_q \sim \pi^q(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$
          $a = a_q$
      else
          $a = a_l$
      end if
      Perform the action a and observe the reward r and new state s'
      if $a_g = 1$ **then**
          $r = r - j(a/a_{\max})$
      end if
      Store transition tuple (s, a, r, s') in $\mathcal{B}_q$
      $r_p + = r$
      $r_g + = r$
      if $t + 1 \bmod \tau$ **then**
          Store transition tuple (s_l, a_l, r_l, s') in $\mathcal{B}_l$
          Store transition tuple $((s_l, a_l), a_g, r_g, s')$ in $\mathcal{B}_g$
      end if
      Sample mini-batches and update parameters for slow and fast
      according to TD3
      Sample mini-batch and update parameters for Gate according to
      Q-learning
    end for

A.5 Detailed Results. Tables 4, 5, and 6 provide the detailed results of our continuous-control experiments.

Table 4: Average Normalized Area-under-Curve (AUC) and Average Return Results.

| Environment | τ | Normalized AUC | | | Average Return | | |
		TD3	TempoRL	TLA	TD3	TempoRL	TLA
Pendulum	6	0.85	0.85	0.87	−147.38 (29.68)	−149.38 (44.64)	−154.92 (31.97)
MountainCar	11	0.19	0.64	0.82	0 (0)	84.56 (28.27)	93.88 (0.75)
Inv-Pendulum	10	0.97	0.77	0.96	1000 (0)	984.21 (47.37)	1000 (0)
Inv-DPendulum	5	0.96	0.94	0.92	9359.82 (0.07)	9352.61 (2.20)	9356.67 (1.23)
Hopper	9	0.66	0.43	0.75	3439.12 (120.98)	2607.86 (342.23)	3458.22 (117.92)
Walker2d	7	0.56	0.52	0.53	4223.47 (543.6)	4581.69 (561.95)	3878.41 (493.97)
Ant	3	0.6	0.33	0.52	5131.90 (687.00)	3507.85 (579.95)	5163.54 (573.19)
HalfCheetah	3	0.79	0.5	0.58	10,352.58 (947.69)	6627.74 (2500.78)	9571.99 (1816.02)

Notes: The standard deviation is reported in parentheses. All results are averaged over 10 trials.

Table 5: Average Action Repetition Percentage and Jerk per Time Step.

Environment	τ	Action Repetition			Average Jerk/Time Step		
		TD3	TempoRL	TLA	TD3	TempoRL	TLA
Pendulum	6	7.44%	34.94%	70.32%	1.02	0.94	0.62
MountainCar	11	9.08%	75.99%	91.4%	0.1	1.12	1.11
Inv-Pendulum	10	1.12%	45.97%	88.82%	1.11	1.62	0.12
Inv-DPendulum	5	0.95%	14.9%	75.22%	0.1	0.61	0.14
Hopper	9	2.51%	64.99%	57.22%	0.46	0.4	0.25
Walker2d	7	2.14%	69.47%	47.45%	0.27	0.2	0.21
Ant	3	0.82%	22.01%	12.68%	0.43	0.39	0.38
HalfCheetah	3	5.64%	14.07%	18.05%	0.8	0.65	0.67

Notes: All results are averaged over 10 trials. Action repetition is measured as the percentage of actions that are the same as the previous action taken.

Table 6: Average Decisions and Million Multiply-Accumulate Operations (MMACs) per Episode and the Multiobjective Score for All Environments.

Environment	Average Decisions			Average MMACs		
	TD3	TempoRL	TLA	TD3	TempoRL	TLA
Pendulum	200	139.39	62.31	24.30	34.14	12.42
MountainCar	999	116.47	10.6	120.98	28.60	2.54
Inv-Pendulum	1000	532.57	111.79	121.90	131.01	26.05
Inv-DPendulum	1000	850.95	247.76	124.70	213.59	57.46
Hopper	998.99	269.85	423.91	125.17	68.43	72.02
Walker2d	988.17	297.4	513.12	127.08	77.29	92.07
Ant	960.57	741.22	860.21	160.22	248.53	243.22
HalfCheetah	1000	889.57	831.42	128.60	230.13	182.35

Note: Decisions and MMACs are averaged over 10 trials.

References

Biedenkapp, A., Rajan, R., Hutter, F., & Lindauer, M. (2021). TempoRL: Learning when to act. In *Proceedings of the International Conference on Machine Learning* (pp. 914–924).

Braylan, A., Hollenbeck, M., Meyerson, E., & Miikkulainen, R. (2015). Frame skip is a powerful parameter for learning to play Atari. In *Proceedings of the AAAI Workshop: Learning for General Competency in Video Games*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym*. arXiv:1606.01540.

Buckland, K. M., & Lawrence, P. D. (1993). Transition point dynamic programming. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems*, 6. Morgan Kaufmann.

Chaganty, A. T., Gaur, P., & Ravindran, B. (2012). Learning in a small world. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*.

Dabney, W., Ostrovski, G., & Barreto, A. (2021). *Temporally-extended ϵ-greedy exploration*. arXiv:2006.01782.

Del Giudice, R., Lechinger, J., Wislowska, M., Heib, D. P., Hoedlmoser, K., & Schabus, M. (2014). Oscillatory brain responses to own names uttered by unfamiliar and familiar voices. *Brain Research*, *1591*, 63–73. 10.1016/j.brainres.2014.09.074

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *Proceedings of the International Conference on Machine Learning* (pp. 1587–1596).

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning* (pp. 1861–1870).

Hansen, E. A., Barto, A. G., & Zilberstein, S. (1996). Reinforcement learning for mixed open-loop and closed-loop control. In M. C. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems*, 9. MIT Press.

Harting, J. K., Updyke, B. V., & van Lieshout, D. P. (1992). Corticotectal projections in the cat: Anterograde transport studies of twenty-five cortical areas. *Journal of Comparative Neurology*, *324*(3), 379–414. 10.1002/cne.903240308

Heitz, R. P. (2014). The speed-accuracy tradeoff: History, physiology, methodology, and behavior. *Frontiers in Neuroscience*, *8*, 150. 10.3389/fnins.2014.00150

Jacq, A., Ferret, J., Pietquin, O., & Geist, M. (2022). *Lazy-MDPs: Towards interpretable reinforcement learning by learning when to act*. arXiv:2203.08542.

Jain, A., Bansal, R., Kumar, A., & Singh, K. (2015). A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *International Journal of Applied and Basic Medical Research*, *5*(2), 124–127. 10.4103/2229-516X.157168

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., . . . Levine, S. (2019). Residual reinforcement learning for robot control. In *Proceedings of the 2019 International Conference on Robotics and Automation* (pp. 6023–6029).

Kalyanakrishnan, S., Aravindan, S., Bagdawat, V., Bhatt, V., Goka, H., Gupta, A., . . . Piratla, V. (2021). *An analysis of frame-skipping in reinforcement learning*. arXiv:2102.03718.

Li, J. S., Sarma, A. A., Sejnowski, T. J., & Doyle, J. C. (2023). Internal feedback in the cortical perception–action loop enables fast and accurate behavior. *Proceedings of the National Academy of Sciences, USA*, *120*(39), e2300445120.

Machado, M. C., Barreto, A., & Precup, D. (2021). *Temporal abstraction in reinforcement learning with the successor representation*. arXiv:2110.05740.

Masset, P., Tano, P., Kim, H. R., Malik, A. N., Pouget, A., & Uchida, N. (2023). *Multi-timescale reinforcement learning in the brain*. bioRxiv:2023–11.

McCallum, A., & Ballard, D. H. (1996). *Reinforcement learning with selective perception and hidden state*. https://api.semanticscholar.org/CorpusID:60716402

McGovern, A., Sutton, R. S., & Fagg, A. H. (1997). Roles of macro-actions in accelerating reinforcement learning. In *Proceedings of the 1997 Grace Hopper Celebration of Women in Computing*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529–533. 10.1038/nature14236

Moore, A. W. (1990). *Efficient memory-based learning for robot control*. Technical report. University of Cambridge.

More, H. L., & Donelan, J. M. (2018). Scaling of sensorimotor delays in terrestrial mammals. *Proceedings of the Royal Society B: Biological Sciences*, *285*. 10.1098/rspb.2018.0613

Morillon, B., Schroeder, C. E., Wyart, V., & Arnal, L. H. (2016). Temporal prediction in lieu of periodic stimulation. *Journal of Neuroscience*, *36*, 2342–2347. 10.1523/JNEUROSCI.0836-15.2016

Nakahira, Y., Liu, Q., Sejnowski, T. J., & Doyle, J. C. (2021). Diversity-enabled sweet spots in layered architectures and speed–accuracy trade-offs in sensorimotor control. *Proceedings of the National Academy of Sciences USA, 118*. 10.1073/pnas.1916367118

Oroojlooyjadid, A., & Hajinezhad, D. (2019). *A review of cooperative multi-agent deep reinforcement learning*. arXiv1908.03963.

Padakandla, S. (2020). A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys*, *54*, 1–25. 10.1145/3459991

Pardo, F., Tavakoli, A., Levdik, V., & Kormushev, P. (2018). Time limits in reinforcement learning. In *Proceedings of the International Conference on Machine Learning* (pp. 4045–4054).

Patel, D., & Siegelmann, H. (2022). *Quicknets: Saving training and preventing overconfidence in early-exit neural architectures*. arXiv:2212.12866.

Precup, D., & Sutton, R. S. (2000). Temporal abstraction in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Puterman, M. L. (1990). Markov decision processes. In D. Heyman & M. Sobel (Eds.), *Handbook in operations research and management science* (vol. 2, pp. 331–434). North-Holland. 10.1016/S0927-0507(05)80172-0

Randløv, J. (1998). Learning macro-actions in reinforcement learning. In M. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems*, *11*. MIT Press.

Scardapane, S., Scarpiniti, M., Baccarelli, E., & Uncini, A. (2020). Why should we add early exits to neural networks? *Cognitive Computation*, *12*(5), 954–966. 10.1007/s12559-020-09734-4

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms*. arXiv:1707.06347.

Sharma, S., Srinivas, A., & Ravindran, B. (2017). *Learning to repeat: Fine grained action repetition for deep reinforcement learning*. arXiv:1702.06054.

Silver, T., Allen, K. R., Tenenbaum, J. B., & Kaelbling, L. P. (2018). *Residual policy learning*. arXiv:1812.06298.

Srinivas, A., Sharma, S., & Ravindran, B. (2017). Dynamic action repetition for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Tack, G.-R., Choi, J., Yi, J., & Kim, C. (2007). Relationship between jerk cost function and energy consumption during walking. In *Proceedings of the World Congress on Medical Physics and Biomedical Engineering 2006* (pp. 2917–2918).

Tan, M. (1991). Cost-sensitive reinforcement learning for adaptive classification and control. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5026–5033).

van Helden, J. F., & Naber, M. (2023). Effects of natural scene inversion on visual-evoked brain potentials and pupillary responses: A matter of effortful processing of unfamiliar configurations. *Neuroscience*, *509*, 201–209. 10.1016/j.neuroscience.2022.11.025

Voros, T. (1999). Minimum jerk theory revisited. In *Proceedings of the First Joint BMES/EMBS Conference* (vol. 1, p. 532).

Weiler, J., Gribble, P. L., & Pruszynski, J. A. (2019). Spinal stretch reflexes support efficient hand control. *Nature Neuroscience*, *22*(4), 529–533. 10.1038/s41593-019-0336-0

Wiestler, T., & Diedrichsen, J. (2013). Skill learning strengthens cortical representations of motor sequences. *eLife*, *2*, e00801. 10.7554/eLife.00801

Yu, H., Xu, W., & Zhang, H. (2021). TAAC: Temporally abstract actor-critic for continuous control. In M. A. Beygelzimer, Y. Dauphin, P. S. Liang, & J. Wortman Vaughan (Eds.), *Advances in neural information processing systems*, *34*. Curran.

Zhang, K., Yang, Z., & Başar, T. (2021). Multi-agent reinforcement learning: A selective overview of the ories and algorithms. In K. G. Vamvoudakis, Y. Wan, & F. L. Lewis (Eds.), *Handbook of reinforcement learning and control*, 321–384.