# Temporally Layered Architecture for Adaptive, Distributed and Continuous Control

Extended Abstract

Devdhar Patel
University of Massachusetts Amherst
Amherst, USA
devdharpatel@cs.umass.edu

Joshua Russell
University of Massachusetts Amherst
Amherst, USA
jgrussell@cs.umass.edu

Francesca Walsh
University of Massachusetts Amherst
Amherst, USA
fnwalsh@umass.edu

Tauhidur Rahman
University of California San Diego
San Diego, USA
trahman@ucsd.edu

Terrence Sejnowski
Salk Institute for Biological Studies
San Diego, USA
terry@salk.edu

Hava Siegelmann
University of Massachusetts Amherst
Amherst, USA
hava@cs.umass.edu

## ABSTRACT

We present temporally layered architecture (TLA), a biologically inspired system for temporally adaptive distributed control. TLA layers a fast and a slow controller together to achieve temporal abstraction that allows each layer to focus on a different time-scale. Our design draws on the architecture of the human brain which executes actions at different timescales depending on the environment's demands. Such distributed control is widespread across biological systems because it increases survivability and accuracy in certain and uncertain environments. We demonstrate that TLA can provide many advantages over existing approaches, including persistent exploration, adaptive control, explainable temporal behavior, compute efficiency and distributed control. We present two different algorithms for training TLA: (a) Closed-loop control, where the fast controller is trained over a pre-trained slow controller, allowing better exploration for the fast controller and closed-loop control where the fast controller decides whether to "act-or-not" at each timestep; and (b) Partially open loop control, where the slow controller is trained over a pre-trained fast controller, allowing for open loop-control where the slow controller picks a temporally extended action or defers the next n-actions to the fast controller. We evaluated our method on a suite of continuous control tasks and demonstrate the advantages of TLA over several strong baselines.

## KEYWORDS

Reinforcement learning; Continuous Control; Distributed Control; Temporal Abstraction

**ACM Reference Format:**
Devdhar Patel, Joshua Russell, Francesca Walsh, Tauhidur Rahman, Terrence Sejnowski, and Hava Siegelmann. 2023. Temporally Layered Architecture for Adaptive, Distributed and Continuous Control: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.



## 1 INTRODUCTION

The success of deep reinforcement learning (DRL) comes from an RL agent acting at a constant frequency. Usually, this frequency is much faster than the average human response time [3]. However, a higher response frequency increases the energy consumption for processing inputs and performing actions. Fast actions also cause more jerky behavior, which might cause damage or discomfort to humans interacting with the RL agent. Additionally, a faster response time increases the task horizon, making it more difficult to train the agent. On the other hand, a slower response time forces the RL agent to take longer "macro" actions without supervision leading to poor performance.

An ideal solution to this issue would be to act fast only when necessary and to reduce the action frequency at other times. To this end, we propose Temporally Layered Architecture (TLA): a reinforcement learning architecture that layers two different networks with different response frequencies to achieve distributed temporally adaptive behavior. To avoid an exponential increase in the number of actions, each network has a constant response frequency - one fast and one slow. However, the RL agent can use their combination to adapt its response frequency. The temporally layered architecture allows TLA to easily abstract hierarchical temporal knowledge into layers that focus on different time-frames. We introduce two different modes of control for TLA: (a) TLA-C: Closed-loop control, where the fast controller is trained over a pre-trained slow controller and (b) TLA-O: Partially open loop control, where the slow controller is trained over a pre-trained fast controller. We demonstrate faster convergence with both TLA variants, as well as increased action repetition with the closed-loop approach and fewer decisions with the partially-open loop approach. Our work highlights that a temporally adaptive approach has similar benefits for AI as has been demonstrated in biology and is an important direction for future research in artificially intelligent control.

## 2 METHODS

The TLA consists of two deep RL networks acting at different timesteps in order to abstract different temporal information about the task. In our experiments, for simplicity, the slower timestep is picked to be a multiple of the faster timestep: $t_{slow} = n \cdot t_{fast}$. Thus, for each action the slow agent picks, the fast agent picks $n$ actions.

| Environment | AUC | | | | Avg. Return | | | |
|---|---|---|---|---|---|---|---|---|
| | TD3 | TempoRL | TLA-O | TLA-C | TD3 | TempoRL | TLA-O | TLA-C |
| Pendulum | 0.86 | 0.86 | **0.92** | **0.98** | -147.82 (30.84) | -148.22(30.24) | -149.97(31.6) | -149.64 (32.03) |
| MountainCar | 0.5 | **0.65** | **0.56** | **0.97** | 0(0) | **56.39(46.05)** | **37.4(45.81)** | **94.19 (0.48)** |
| Inv-Pendulum | 0.97 | 0.95 | 0.53 | **0.99** | 1000 (0) | 995.83(12.48) | 1000(0) | 1000 (0) |
| Inv-DPendulum | 0.96 | - | - | **0.98** | 9359.82(0.07) | - | - | 9358.94(0.82) |
| Hopper | 0.64 | - | - | **0.67** | 3405.53(130.04) | - | - | **3454.15(190.17)** |
| Walker2d | 0.57 | - | - | **0.74** | 4080.99(448.3) | - | - | 4008.8(633.34) |
| Ant | 0.63 | - | - | 0.63 | 4406.36(1047.34) | - | - | 3643.76(828.52) |

**Table 1: Average normalized AUC and average return results. The standard deviation is reported in parentheses. TLA-O and TLA-C represent the open and closed loop forms of TLA, respectively. We highlight all results that are better than the baseline (TD3). All results are averaged over 10 trials.**

| Environment | Action Repetitions | | | | Decisions | | | |
|---|---|---|---|---|---|---|---|---|
| | TD3 | TempoRL | TLA-O | TLA-C | TD3 | TempoRL | TLA-O | TLA-C |
| Pendulum | 7.50% | 23.02% | 14.32% | 69.84% | 200.00 | 159.96 | 183.95 | 200.00 |
| MountainCar | - | 70.81% | 13.10% | 67.77% | 999 | 138.98 | 616.67 | 77.3 |
| InvertedPendulum | 1.12% | 55.29% | 60.87% | 46.15% | 1000 | 445.2 | 392.8 | 1000 |
| InvertedDPendulum | 0.96% | - | - | 49.41% | 1000 | - | - | 1000 |
| Hopper | 3.43% | - | - | 21.74% | 996.61 | - | - | 989.41 |
| Walker2d | 2.00% | - | - | 29.87% | 990.57 | - | - | 942.36 |
| Ant | 0.58% | - | - | 22.73% | 975.82 | - | - | 943.62 |

**Table 2: Average action repetition percentage and decisions for each algorithm. Action repetition percentage represents the fraction of consecutive actions in an episode for which the actions are identical. Decisions represent the number of times in an episode a new action was computed. All results are averaged over 10 trials.**

### 2.1 Closed-loop temporally layered architecture (TLA-C)

In this setting, the slow network is pre-trained at a larger timestep. A larger timestep allows the network to better explore the environment and thus converge to an optimal policy faster. However, the optimal policy for the slow network might achieve a lower average return due to the longer timesteps. We then train the fast network on top of it so that the faster agent is acting in an environment where the actions are already being performed by the slower agent. To achieve this, we consider a residual action that is the combination of the actions picked by the fast and slow networks. During training, the fast network is penalized for taking actions with large magnitudes. After training, we threshold fast actions so that only actions above the threshold are performed in the environment. By thresholding the fast action when its influence on the final action is low, we reduce jerky behavior, promoting long smooth actions.

### 2.2 Open-loop temporally layered architecture (TLA-O)

In this setting, we allow partially open loop control by allowing the slow controller to gate the computation of the fast controller. This is achieved by training a slow agent over a pre-trained fast controller. The actions of the slow agent are augmented to include a binary gate output $g \in \{0, 1\}$ that gates the activation of the fast network. Gating the fast network reduces the compute cost in addition reducing the jerk.

## 3 EXPERIMENTS

To evaluate our algorithms we measure their performance on seven continuous control tasks, including five from the MuJoCo [6] suite, using the OpenAI Gym interface [2]. The neural networks are implemented and trained using the PyTorch framework [4]. We also compare TLA-O with TempoRL [1], a method that learns an additional action-repetition policy which decides on the number of timesteps to repeat a chosen action.

Tables 1 and 2 show the Area under curve (AUC) for learning speed, avg. return, action repetition percentage, and avg. decisions for each algorithm. Note that since open-loop control is difficult, we only run it on simple environments. For open loop control, the frequency of the slow network was four times slower than the fast network. For the rest of the environments, it was set to twice as slow. Fast network frequency is set to the default for the environment. As demonstrated, TLA-C learns significantly faster while increasing action repetitions (thus reducing jerk). On the other hand, TLA-O also reduces the number of decisions required for control, thus conserving compute energy. The full paper on this work is available online [5].

## ACKNOWLEDGMENTS

This material is based upon work partially supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112190041. The information contained in this work does not necessarily reflect the position or the policy of the Government.

## REFERENCES

[1] André Biedenkapp, Raghunandan Rajan, Frank Hutter, and Marius Thomas Lindauer. 2021. TempoRL: Learning When to Act. In *ICML*.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540

[3] OpenAI. 2018. OpenAI Five. https://blog.openai.com/openai-five/. Accessed:2022-08-12.

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[5] Devdhar Patel, Joshua Russell, Francesca Walsh, Tauhidur Rahman, Terrance Sejnowski, and Hava Siegelmann. 2022. Temporally Layered Architecture for Adaptive, Distributed and Continuous Control. *arXiv preprint arXiv:2301.00723* (2022).

[6] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033. https://doi.org/10.1109/IROS.2012.6386109