Efficient Multi-Horizon Learning for Off-Policy Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Value estimates at multiple timescales can help create advanced discounting functions and allow agents to form more effective predictive models of their environment. In this work, we investigate learning over multiple horizons concurrently for off-policy deep reinforcement learning using an efficient architecture that combines a deeper network with the crucial components of Rainbow, a popular value-based off-policy algorithm. We use an advantage-based action selection method and our proposed agent learns over multiple horizons simultaneously while using either an exponential or hyperbolic discounting function to estimate the advantage that guides the acting policy. We test our approach on the Procgen benchmark, a collection of procedurally-generated environments, to demonstrate the effectiveness of this approach, and specifically, to evaluate the agent's performance in previously unseen scenarios.

1 Introduction

A reinforcement learning agent learns to maximize the rewards it receives over its expected lifetime. But what if it's estimate of the expected lifetime is biased? While RL algorithms usually plan for a single, fixed, *distant* horizon by setting the discount factor γ to a value closer to 1, the agent may not live long enough as it had previously expected. Thus, it might have fared better by basing its policy on estimates over multiple time horizons; short, intermediate and long. For example, humans make plans for the immediate future (work day), short-term (professional goals) and long-term (retirement savings), but a person's policy (based on valuation of distant rewards) may change drastically because of a terminal diagnosis (curtailment of expected lifetime). Since the goal of an RL agent is to optimize its return (cumulative reward) through a combination of prediction and control, the prediction method used to estimate the value function (estimated return) plays a central role in the agent's performance.

The discount factor γ determines the timescale of the return. When γ is closer to 0, the agent becomes short-sighted and only maximizes near-term reward, whereas when γ reaches closer to 1, the agent values rewards far into the future. It has been shown that a lower discount value early in learning can help policies to converge, although too low a discount can lead to sub-optimal policies [Bertsekas and Tsitsiklis, 1995]. Thus learning a value representation that consider multiple discount factors is important. In deep RL, previous works have attempted to learn a value function at multi-timescale either by concurrently learning value estimate for a fixed set of γs or conditioning the value estimate over γ. All of these methods require multiple fully connected layers at the end of the network to enable γ-specific value prediction and introduce computational overhead compared to the single-valued approach. Off-policy reinforcement learning is generally considered data-efficient in the sense that it can reuse transitions collected from old policies. However, the added computational complexity for multi-horizon learning entails a necessity for an efficient neural network architecture that can enhance the learning process.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

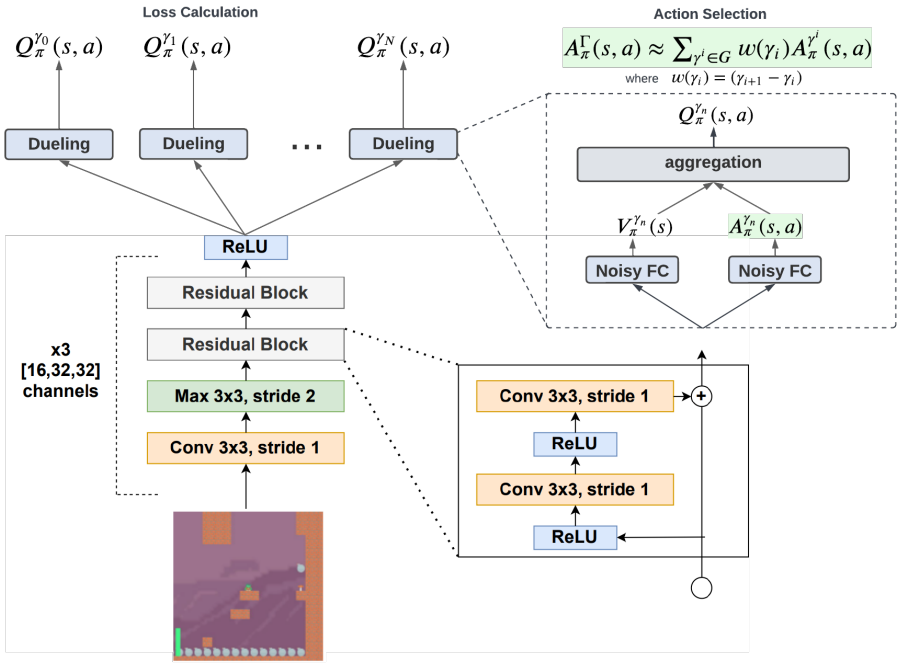


Figure 1: Multi-Horizon advantage is calculated using advantages from the dueling head for each gamma value, and approximated using a Riemann sum over the discrete internal $G = \gamma_0, \gamma_1, ..., \gamma_N$. Output layers predict Q-values for different discount factors using an individual output block for each gamma, and these layers are used for the loss calculation.

In this work, we explore multi-horizon learning from the perspective of enabling performance improvement of off-policy algorithm through deep feature learning and better exploration in diverse settings. We present results on procedurally generated environments, which are designed to test an agent's ability to learn a robust and generalizable policy. Our approach behind the multi-horizon learning is primarily based on Fedus et al. [2019], however, we extend the idea by using an extension of the dueling DQN architecture [Wang et al., 2016] coupled with parametric noise injection in weights and an advantage-based action selection method. We evaluate both exponential and hyperbolic discounting functions in the multi-horizon setting and compare and contrast these with learning over a fixed, single-horizon setting, across the Procgen benchmark [Cobbe et al., 2020].

2 Related Work

The need for abstract planning (such as modeling over multiple timescales) is a fundamental problem in AI. Sutton [1995] extend temporal-difference methods to make predictions that are not specific to a single time scale. Studies from psychology and neuroscience show that humans and animals estimate rewards at numerous timescales [Tanaka et al., 2016] and do not discount future rewards exponentially using a single discount factor γ [Mazur, 1997], instead discounting them hyperbolically [Green et al., 1994]. Kurth-Nelson and Redish [2009] proposed the modeling of hyperbolic discounting via distributed exponential discounting and Fedus et al. [2019] extended this formulation to deep reinforcement learning by approximating hyperbolic discounting from exponential discounting while using multi-horizon (multi-timescale) learning as an auxiliary task. Xu et al. [2018] present a meta-learning approach and propose to use γ as a learn-able parameter of their network. Sherstan et al. [2020] propose Γ-nets to predict value-function for any discount factor γ by using timescale as an input to the value estimator.

3 Methodology

We aim to facilitate multi-horizon learning for value-based algorithm by incorporating several existing extensions to the DQN algorithm [Mnih et al., 2015], some of which have not been used in similar prior works, and use a deeper network for the function approximation.

3.1 Network Architecture and Rainbow Extensions

The proposed network architecture is presented in Figure 1. We employ the deeper IMPALA-CNN architecture [Espeholt et al., 2018] with 15 convolutional layers instead of the small 3-layer network (Nature-CNN) used in Fedus et al. [2019] for multi-horizon learning. The residual blocks [He et al., 2016] in the IMPALA-CNN architecture keep the optimization process easier while enabling substantially deeper feature learning.

We limit our choice to the six extensions used in the Rainbow [Hessel et al., 2018] as these extensions deal with complementary issues and can easily be combined together. We include five extensions out of that six namely double Q-learning, prioritized experience replay, n-step learning, dueling DQN, noisy net; however, we exclude the Distributional RL as this component requires a high degree of complexity as compared to its benefits, when evaluating for less number of time steps. Hessel et al. [2018] show that optimizing the distribution of returns helps in the long run, such as training beyond 40M time steps. Schmidt and Schmied [2021] note only a marginal performance improvement with the inclusion of Distributional RL and suggest not to include this component when training for 10M time steps. In contrast to Fedus et al. [2019], our approach includes the dueling network and exploration through noisy nets in the multi-horizon learning setting. Our results show that these two components are important factors in the agent's performance, especially when assessing generalizable skills on the Procgen benchmark.

3.2 Advantage-based Action Selection

Hyperbolically-discounted advantage estimation has been recently proposed for actor-critic methods [Nafi et al., 2022], however to the best of our knowledge, the use of such a method for value-based off-policy methods has not been proposed before. While using the dueling DQN [Wang et al., 2016] architecture, we use the centered advantage from the dueling head for action selection. To calculate hyperbolically discounted-advantage, we use the hyperbolic function evaluation $\Gamma_k(t) = \frac{1}{1+kt} = \int_0^1 \gamma^{kt} d\gamma$ and estimate hyperbolically-discounted advantage as:

$$A_\pi^\Gamma(s,a) \approx \sum_{\gamma^i \in G} w(\gamma_i) A_\pi^{\gamma^i}(s,a)$$

Hyperbolic advantage is calculated using the advantages in the dueling head for each gamma value. and approximated using a Riemann sum over the discrete internal $G = [\gamma_0, \gamma_1, ..., \gamma_N]$.

4 Implementation

Rainbow [Hessel et al., 2018] combines several independent improvements on top of the standard Q-Learning framework. It uses six extensions to DQN namely double DQN [van Hasselt et al., 2016], dueling DQN [Wang et al., 2016], noisy nets [Fortunato et al., 2018], Distributional RL (C51) [Bellemare et al., 2017], prioritized experience replay buffer [Schaul et al., 2016] and n-step returns [Sutton, 1988]. Using a fast and data-efficient implementation of Rainbow [Schmidt and Schmied, 2021], we implement multi-horizon learning for both hyperbolic discounting and exponential discounting (using advantage corresponding to largest gamma for action selection). Our multi-horizon learning implementation is inspired by Fedus et al. [2019] but differs in a number of ways. We use hyperbolically discounted advantage estimates instead of hyperbolically discounted Q-values for the action selection. The network architecture of Nature CNN is replaced with an IMPALA large (2 channel) network.

The Rainbow implementation in Fedus et al. [2019] used three of the six improvements proposed in Hessel et al. [2018] namely Distributional RL, prioritized replay buffer and n-step returns, whereas our implementation uses five of the six improvements of Rainbow (double DQN, dueling DQN, noisy

nets, prioritized replay buffer, n-step returns but does not use distributional DQN). In determining our γ interval, we follow the same scheme as suggested by [Fedus et al., 2019].

We use Procgen [Cobbe et al., 2020] to evaluate our approach. The Procgen benchmark is designed to study sample efficiency and generalization in reinforcement learning. The agent's aim is to perform better in unseen scenarios or levels of an environment while trained on a smaller subset of the levels, where the levels generally vary in terms of visual appearances and positions of the entities, environment backgrounds, and dynamics. We model the environment as a Partially Observable Markov Decision Process (POMDP) and each level of the game is a sampled POMDP instance. We provide the hyperparameters used in our implementation in Table 1.

hyperparameters	**values**
Q-target update frequency	32,000 steps
Importance sampling beta for PER	0.45
n-step	3
σ_0 for noisy linear layers	0.5
Learning rate	0.00025
batch size	256
Adam e parameter	0.00002 (0.005/batch size)
parallel environments	64
Replay buffer size	262,144 (2^{18})
Training starts at	80,000 steps
γ (single-horizon)	0.99
γ_{max}	0.99
number of γ	5
γ values	[0.374, 0.608, 0.755, 0.847, 0.904]
k (hyperbolic exponent)	0.1
integral estimate	lower
number of seeds per environment	3
random seed values	[74330, 95762, 801604]

Table 1: Hyperparameters for Rainbow

5 Experiments and Results

We build upon the Rainbow implementation of Schmidt and Schmied [2021] to evaluate multi-horizon learning in off-policy reinforcement learning. The single-horizon agent uses vanilla Rainbow and learns over a fixed horizon with a single γ=0.99 and discounts rewards exponentially. For multi-horizon, the agent simultaneously learns Q-values for N many γ, but has the choice between acting either using a hyperbolically discounted-advantage, $A_\pi^\Gamma(s,a)$, or exponentially-discounted advantage, $A_\pi^{\gamma_n}(s,a)$, corresponding to the largest gamma value (γ_n).

Figures 2 and 3 show the performance of the Rainbow variants over a subset of four Procgen games. Results confirm that agents which can model value estimates over multiple timescales perform better than their single-horizon counterparts. Moreover, advanced discounting functions like hyperbolic that are leveraged through the use of multiple timescales perform even better than exponentially discounted, multi-horizon (largest gamma) variants. As shown in Table 2, when run across the full suite of 16 environments over 25M timesteps, the aggregate score for multi-horizon (hyperbolic) is higher than the single-horizon variant.

Table 2: Aggregated Train and Test scores across all 16 environments after training over 25M environment steps.

Score	**Single-Horizon**	**Multi-Horizon**
Train	75.88 ± 14.97	$\mathbf{85.33 \pm 10.06}$
Test	45.04 ± 12.68	$\mathbf{57.89 \pm 09.50}$

5.1 Performance on subset of 4 environments

In order to test our hypothesis, we initially ran experiments over a subset of 4 Procgen environments. We test the single-horizon and multi-horizon methods (with largest gamma exponential discounting and hyperbolic discounting) to ascertain the performance of the agent. Figure 3 shows that the Multi-Horizon (hyperbolic discounting) performed better than both Multi-Horizon (largest gamma) and Single-Horizon Rainbow variants on three out of the four games, with comparable performance in the other game. Due to the high training cost, subsequent runs were only made for evaluating Multi-Horizon (hyperbolic) against Single-Horizon variant.

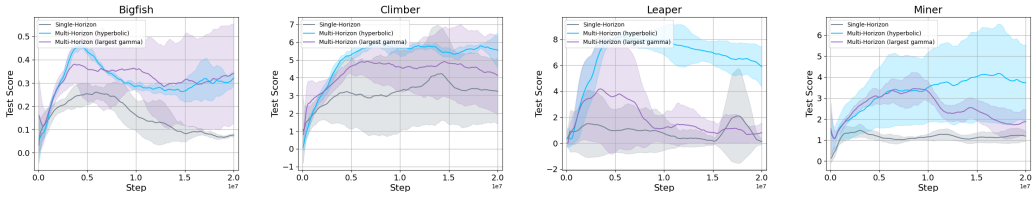


Figure 2: Test performance of Single-Horizon, Multi-Horizon (largest gamma), and Multi-Horizon (hyperbolic discounting) versions of Rainbow over a subset of four Procgen environments trained over 20M timesteps.

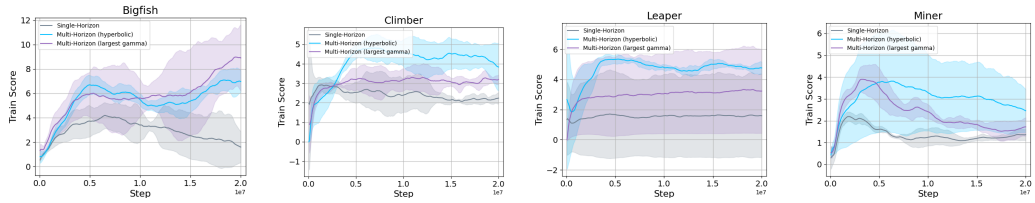


Figure 3: Train performance of Single-Horizon, Multi-Horizon (largest gamma), and Multi-Horizon (hyperbolic discounting) over four Procgen environments. Means and standard deviations are calculated over 3 trials, each with a different seed.

Table 3: Scores on test levels for Single-Horizon Rainbow and Multi-Horizon (hyperbolic) Rainbow after training on 25M environment steps for all 16 environments.

Environment	Single-Horizon	Multi-Horizon (hyperbolic)
Bigfish	0.14±0.02	**0.32±0.019**
Bossfight	0.00±0.00	0.00±0.00
Caveflyer	3.99±0.25	**4.64±0.29**
Chaser	0.82±0.06	**1.18±0.10**
Climber	3.19±1.90	**5.27±0.25**
Coinrun	7.47±0.85	**8.04±0.56**
Dodgeball	**0.06±0.07**	0.05±0.04
Fruitbot	**2.33±0.08**	1.94±0.03
Heist	3.37±1.32	**4.18±1.28**
Jumper	**2.87±0.01**	2.82±0.11
Leaper	0.72±1.25	**6.74±0.75**
Maze	6.62±4.69	**7.62±2.73**
Miner	1.26±0.20	**3.37±1.65**
Ninja	**6.45±0.48**	6.30±0.27
Plunder	1.49±1.29	**1.57±1.38**
Starpilot	**4.18±0.23**	3.79±0.10

5.2 Test performance on all 16 environments

Figure 4 shows the test performance of Single-Horizon and Multi-Horizon (hyperbolic discounting) Rainbow variants on all Procgen games for 25M timesteps. Our proposed extensions outperform the baseline on the majority of the environments, while remaining competitive with the rest, with the exception of an unknown issue in the Bossfight environment during evaluation. Test score values are given in Table 3.

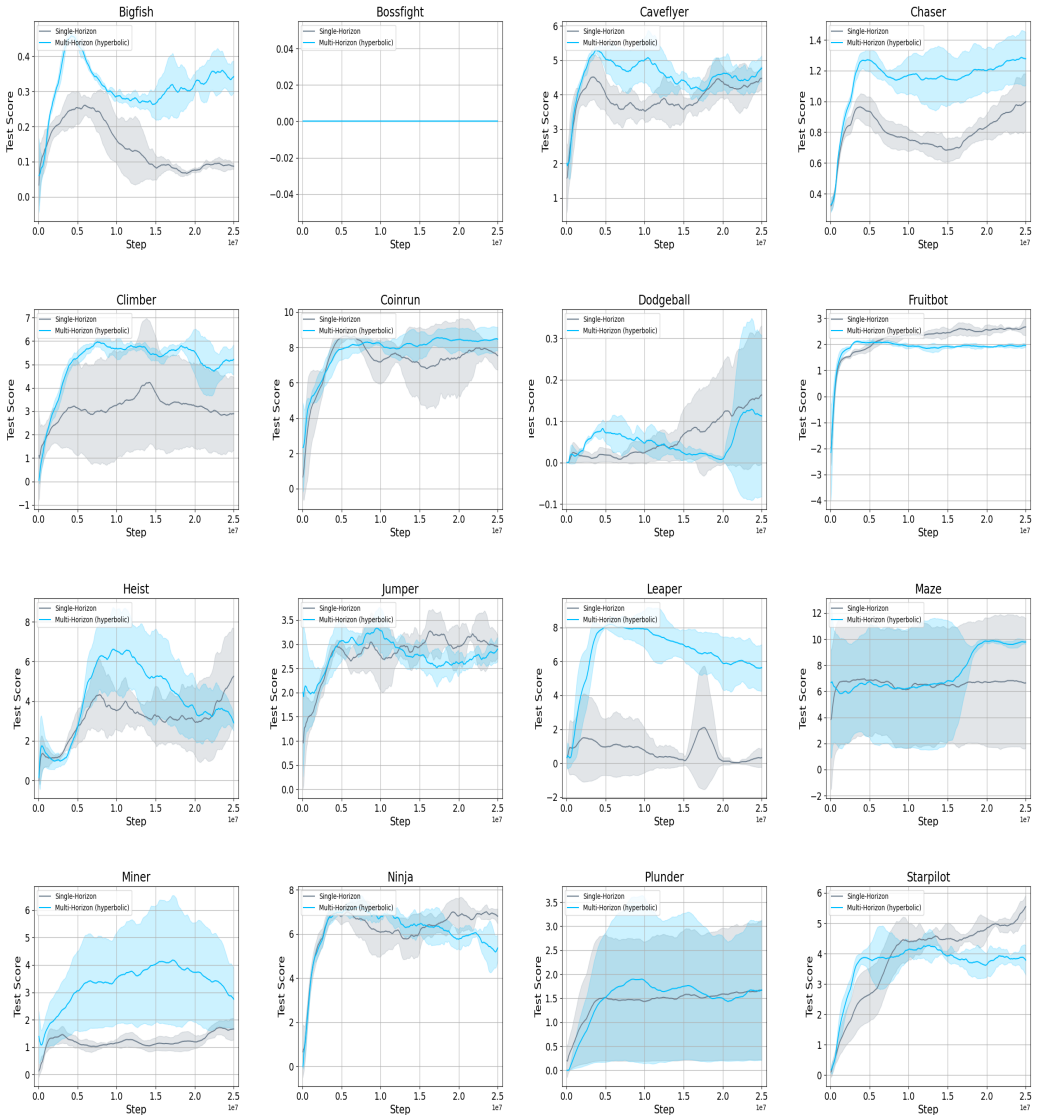


Figure 4: Test performance for Single-Horizon and Multi-Horizon (hyperbolic) variants of Rainbow across the entire Procgen benchmark. Mean and standard deviation are calculated over 3 trials, each with a different seed.

5.3 Train performance on all 16 environments

Figure 5 shows the train performance of Single-Horizon and Multi-Horizon (hyperbolic discounting) Rainbow variants on all Procgen games. Our proposed methods outperform the single-horizon on the majority of the games, while remaining competitive in the rest.

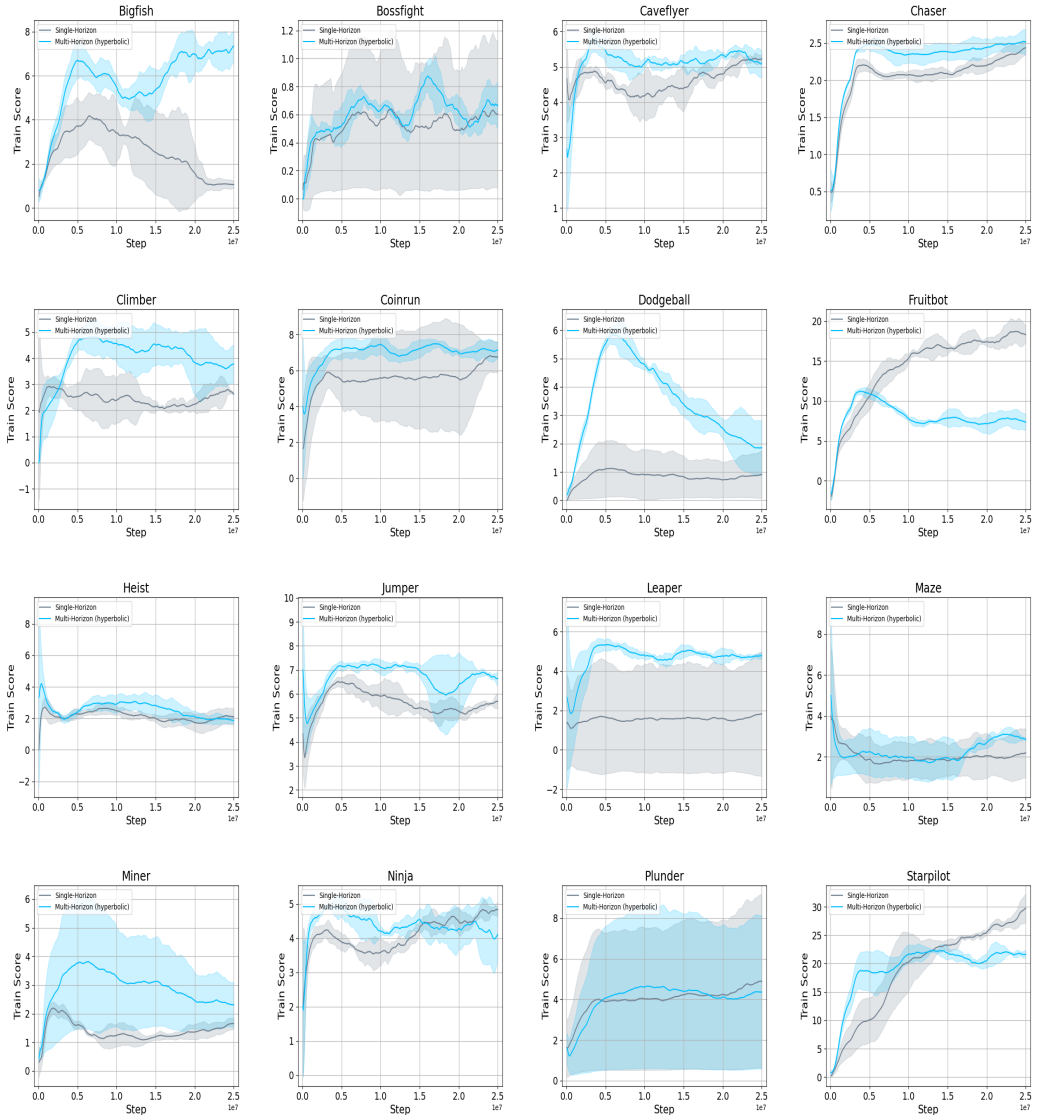
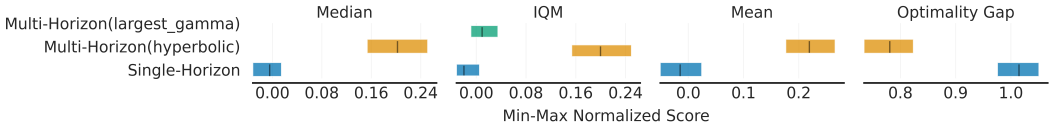


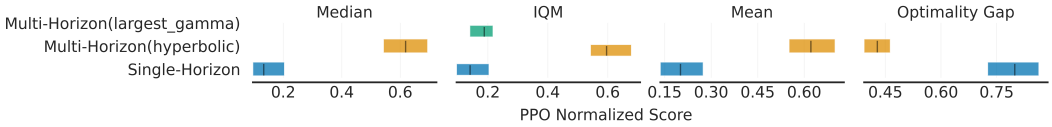
Figure 5: Test performance for Single-Horizon and Multi-Horizon (hyperbolic) variants of Rainbow across the entire Procgen benchmark. Mean and standard deviation are calculated over 3 trials, each with a different seed.

6 rliable: Evaluating statistical uncertainty

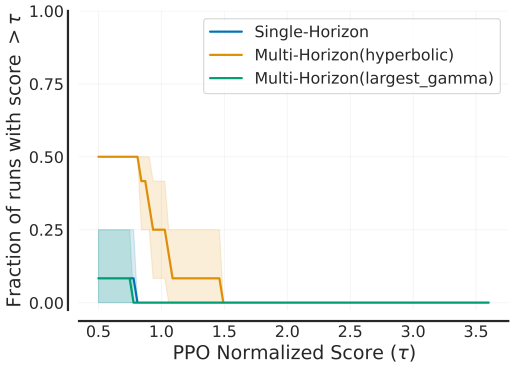
In order to reliably evaluate the results and reduce statistical uncertainty in the comparisons, we analyze the performance of our approach as proposed in Agarwal et al. [2021] through the *rliable* library [Agarwal, 2021]. This was all the more important since our trials used only 3 seeds. In order to perform a robust evaluation of the results, we compute performance statistics across all games within the Procgen benchmark using the metrics as mentioned in the rliable paper. We go over the rliable metrics used in our results. **Interquartile Mean (IQM)** takes the second and third quartiles



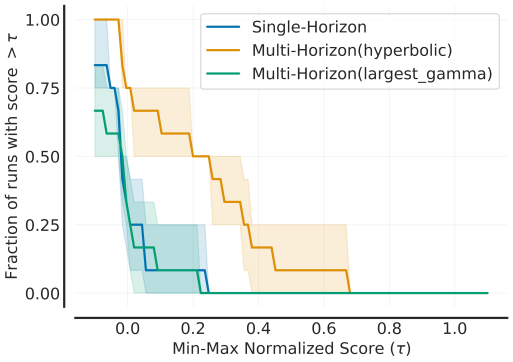
(a) Aggregate Metrics (Min-Max normalized)



(b) Aggregate Metrics (PPO normalized)



(c) Performance Profiles



(d) Performance Profiles

Figure 6: Performance comparison of Single-Horizon, Multi-Horizon (largest gamma), and Multi-Horizon (hyperbolic discounting) over four Procgen environments. **(a)** Aggregate Metrics (Min-Max Normalized) including Mean, Mean, IQM and OG metrics for each method. **(b)** Aggregate Metrics (PPO Normalized) including Mean, Mean, IQM and OG metrics for each method. **(c)** PPO Normalized Performance Profile of the three methods. **(d)** Min-Max Normalized Performance Profile of the three methods.

(the middle 50%) of the runs combined across seeds and environments, and calculates the mean score. IQM is not affected by outliers and reduces the statistical uncertainty even with few runs. **Optimality gap (OG)** measures the proportion of performance that fails to meet a minimum threshold score of $\alpha = 1.0$, such that scores beyond α are not considered very important. For both IQM and OG, instead of taking point estimates, interval estimates using stratified bootstrap confidence intervals are computed. **Performance profile** of an algorithm depicts its score distribution as the fraction of runs above a certain normalized score. These profiles visualize the empirical tail distribution function of a random score, but with stratified bootstrap sampling to produce point-wise confidence bands. The higher the curve of a method, the better it is. The scores are normalized either using PPO mean or through the achievable min-max scores given for each environment as found in Appendix C Normalization Constants of Cobbe et al. [2020].

The results are shown in Figure 6 which confirm our earlier findings of Multi-Horizon (hyperbolic) variant performing better then Multi-Horizon (largest gamma), alluding to the fact that for procedurally generated environments and as such any environment where we want an agent to learn a robust policy (unlike memorizing trajectories), learning over multiple horizons can not be just thought of as an auxiliary task. By discounting hyperbolically and *utilizing value estimates from all time horizons to make decisions (acting policy)* is an important consideration for learning intelligent behavior.

7 Conclusion

We report initial results on the use of multi-horizon learning using advantage-based action selection in procedurally generated environments. Our contribution can be seen as validating the impact of

multi-horizon learning on an agent's policy through learning accurate value estimates, achieving higher sample efficiency and better generalization.

References

Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Proceedings of 34th IEEE conference on decision and control*. IEEE, 1995.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. *arXiv preprint arXiv:1902.06865*, 2019.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 2016.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *ICML*, pages 2048–2056. PMLR, 2020.

Richard S Sutton. Td models: Modeling the world at a mixture of time scales. In *Machine Learning Proceedings 1995*, pages 531–539. Elsevier, 1995.

Saori C Tanaka, Kenji Doya, Go Okada, Kazutaka Ueda, Yasumasa Okamoto, and Shigeto Yamawaki. Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops. In *Behavioral economics of preferences, choices, and happiness*, pages 593–616. Springer, 2016.

James E Mazur. Choice, delay, probability, and conditioned reinforcement. *Animal Learning & Behavior*, 25(2):131–147, 1997.

Leonard Green, Nathanael Fristoe, and Joel Myerson. Temporal discounting and preference reversals in choice between delayed outcomes. *Psychonomic Bulletin & Review*, 1(3):383–389, 1994.

Zeb Kurth-Nelson and A David Redish. Temporal-difference reinforcement learning with distributed representations. *PLoS One*, 4(10):e7362, 2009.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Craig Sherstan, Shibhansh Dohare, James MacGlashan, Johannes Günther, and Patrick M Pilarski. Gamma-nets: Generalizing value estimation over timescale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5717–5725, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI conference on artificial intelligence*, 2018.

Dominik Schmidt and Thomas Schmied. Fast and data-efficient training of rainbow: an experimental study on atari. *Deep RL Workshop NeurIPS*, 2021. URL `https://arxiv.org/abs/2111.10247`.

Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Hyperbolically discounted advantage estimation for generalization in reinforcement learning. In *Decision Awareness in Reinforcement Learning Workshop at ICML*, 2022.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *Proceedings of the International Conference on Representation Learning (ICLR)*, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2016.

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

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *NeurIPS*, 2021.

Rishab Agarwal. rliable. `https://github.com/google-research/rliable`, 2021.

Supplementary Materials

A ML Reproducibility Checklist

A.1 For all models and algorithms presented, check if you include:

- A clear description of the mathematical setting, algorithm, and/or model.
  Yes
- A clear explanation of any assumptions.
  Yes
- An analysis of the complexity (time, space, sample size) of any algorithm.
  No

A.2 For any theoretical claim, check if you include:

- A clear statement of the claim
  Yes
- A complete proof of the claim.
  Not provided. Reader is referred to Fedus et al. [2019] for proofs.

A.3 For all datasets used, check if you include:

- The relevant statistics, such as number of examples
  Publicly available dataset (Procgen used)
- The details of train / validation / test splits.
  Yes
- An explanation of any data that were excluded, and all pre-processing step.
  Not Applicable.
- A link to a downloadable version of the dataset or simulation environment.
  Publicly available dataset/environment (Procgen used)
- For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control.
  Yes

A.4 For all shared code related to this work, check if you include:

- Specification of dependencies
  Yes
- Training and evaluation code.
  Yes, see attached zip file
- (Pre-)trained model(s).
  No
- README file includes table of results accompanied by precise command to run to produce those results
  Yes, see attached zip file

A.5 For all reported experimental results, check if you include:

- The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.
  Yes
- The exact number of training and evaluation runs.
  Yes
- A clear definition of the specific measure or statistics used to report results.
  Yes

- A description of results with a central tendency (e.g. mean) and variation (e.g. error bars).
  Yes
- The average runtime for each result, or estimated energy cost.
  Yes
- A description of the computing infrastructure used.
  Yes

B Setup

The software used included Python (3.8.x) and PyTorch (1.12), with CUDA (11.4). Our computing infrastructure involved using 2 GPU servers, each having an Intel Xeon 4215 processor (3.50GHz), 16 physical cores, 1 TB RAM, and 8 Nvidia A40 GPUs. Each experiment was run using one GPU, and took between around 1 day (about 25 hours), totaling 4.5 days (about 108 hours) for the set of experiments (16 envs x 3 seeds x 2 methods + 4 envs x 3 seeds x 1 method). The experiments did not use vectorized environments.