

Architectural and algorithmic strategies for generalizable
deep reinforcement learning

by

Nasik Muhammad Nafi

B.S., Bangladesh University of Engineering and Technology, 2015

M.S., Kansas State University, 2019

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Abstract

Generalization in Reinforcement Learning (RL) refers to the ability to transfer the learned policy to previously unseen task variations - specifically, state and/or reward distributions. In high-dimensional observation settings, current deep RL agents struggle to generalize even over in-distribution but unseen variations, unlike biological agents which can quickly adapt to new conditions by generalizing over prior learned behaviors. To alleviate this problem, this dissertation focuses on efficiently encoding structural assumptions and inductive bias through *architectural* facet, which determines the computation structure and underlying learning representation; and *algorithmic* facet, which defines the optimization process.

This dissertation investigates the policy and value networks at a high level of design abstraction and the policy and value approximation objectives. The central hypothesis is that robust representation learning and value estimation free of overfitting can facilitate zero-shot generalization to new task instances. First, an attention-based partially decoupled actor-critic architecture is introduced to enhance generalization by addressing policy-value representation asymmetry while remaining computationally efficient. Additional analysis on sensitivity to the degree of decoupling provides a designer-level model selection mechanism. Second, to improve resilience to potential observational variations, RL using augmentation-invariant representation is introduced, learning latent representations through a non-contrastive self-supervised approach with minimal modifications to the underlying RL structure. Third, variations in environmental conditions are shown to significantly impact task completion times or episode lengths, resulting in high variance in value estimation. A novel value approximation technique is therefore introduced that reshapes the value targets considering the possibility of episode length variation. Finally, horizon regularization is introduced to estimate an advantage for policy optimization that leverages multiple horizons, avoiding the implications of a fixed effective horizon imposed by a single discount factor.

Architectural and algorithmic strategies for generalizable
deep reinforcement learning

by

Nasik Muhammad Nafi

B.S., Bangladesh University of Engineering and Technology, 2015

M.S., Kansas State University, 2019

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
William H. Hsu

Copyright

© Nasik Muhammad Nafi 2024.

Abstract

Generalization in Reinforcement Learning (RL) refers to the ability to transfer the learned policy to previously unseen task variations - specifically, state and/or reward distributions. In high-dimensional observation settings, current deep RL agents struggle to generalize even over in-distribution but unseen variations, unlike biological agents which can quickly adapt to new conditions by generalizing over prior learned behaviors. To alleviate this problem, this dissertation focuses on efficiently encoding structural assumptions and inductive bias through *architectural* facet, which determines the computation structure and underlying learning representation; and *algorithmic* facet, which defines the optimization process.

This dissertation investigates the policy and value networks at a high level of design abstraction and the policy and value approximation objectives. The central hypothesis is that robust representation learning and value estimation free of overfitting can facilitate zero-shot generalization to new task instances. First, an attention-based partially decoupled actor-critic architecture is introduced to enhance generalization by addressing policy-value representation asymmetry while remaining computationally efficient. Additional analysis on sensitivity to the degree of decoupling provides a designer-level model selection mechanism. Second, to improve resilience to potential observational variations, RL using augmentation-invariant representation is introduced, learning latent representations through a non-contrastive self-supervised approach with minimal modifications to the underlying RL structure. Third, variations in environmental conditions are shown to significantly impact task completion times or episode lengths, resulting in high variance in value estimation. A novel value approximation technique is therefore introduced that reshapes the value targets considering the possibility of episode length variation. Finally, horizon regularization is introduced to estimate an advantage for policy optimization that leverages multiple horizons, avoiding the implications of a fixed effective horizon imposed by a single discount factor.

Table of Contents

List of Figures	xi
List of Tables	xviii
Acknowledgements	xix
Dedication	xxi
1 Introduction	1
1.1 Problem Statement	2
1.2 Contributions	3
2 Background	8
2.1 Mathematical Formulation	8
2.1.1 Markov Decision Processes	8
2.1.2 Contextual Markov Decision Processes	9
2.2 Reinforcement Learning Methods	9
2.2.1 Value-based Methods	10
2.2.2 Policy-based Methods	11
2.2.3 Actor-Critic Methods	12
2.3 Selected Actor-Critic Algorithms	13
2.3.1 Proximal Policy Optimization	13
2.3.2 Soft Actor-Critic	14
2.4 Relevant Additional Concepts	15
2.4.1 Deep RL and Representation Learning	15

2.4.2	Discount Factor and Effective Horizon	15
2.5	Procedurally Generated Environments	16
2.5.1	Procgen	17
2.5.2	Crafter	17
2.5.3	MiniGrid	18
3	Partial Decoupling of Actor and Critic	19
3.1	Introduction	19
3.2	Related Work	21
3.3	Preliminaries	22
3.4	Attention-based Partially Decoupled Actor-Critic	23
3.4.1	Partial Decoupling of Policy and Value function	23
3.4.2	Relevant Feature Learning using Attention	26
3.5	Experiments and Results	28
3.5.1	Network Architecture	29
3.5.2	Generalization Performance on Test Distribution	29
3.5.3	Assessing the Generalization Gap	32
3.5.4	Ablation	32
3.5.5	Visualization of Value and Policy using Grad-CAM	32
3.6	Analyzing the Generalization Sensitivity to the Extent of Policy-Value De- coupling	35
3.6.1	Instantiating Different Extent of Decoupling	36
3.6.2	Experimental Setup	39
3.6.3	Generalization Across Benchmarks	40
3.6.4	Performance on Individual Procgen Environments	42
3.6.5	Extended Study for Early Separation	45
3.7	Conclusion	47

4	Minimalist Approach to Augmentation Invariant Representation Learning in RL	48
4.1	Introduction	48
4.2	Related Works	50
4.3	Background	51
4.3.1	Contextual Markov Decision Process	51
4.3.2	Proximal Policy Optimization	52
4.4	Augmentation Invariant Representation for RL	53
4.4.1	Augmentation Invariant Representation	54
4.4.2	Combining Representation Learning with RL	56
4.5	Experiments and Discussions	58
4.5.1	Impact of Variance and Covariance with Invariance	60
4.5.2	Relation to the Batch Size	61
4.5.3	Sensitivity to Hyperparameters	62
4.5.4	Importance of Update Sequence	62
4.6	Conclusion	63
5	Value Target Shaping to Address Uncertainty in Episode Length	64
5.1	Introduction	64
5.2	Preliminaries	66
5.2.1	Contextual Markov Decision Process	66
5.2.2	Proximal Policy Optimization	67
5.3	Methodology	68
5.3.1	Intuition	68
5.3.2	Augmented Value Targets (AVaTar)	69
5.3.3	Efficient Computation using Average Augmented Target	71
5.3.4	Network Architecture	73
5.4	Results and Discussions	74
5.4.1	Generalization Performance on Test Distribution	75

5.4.2	Assessing the Generalization Performance	78
5.4.3	Additional Analysis	78
5.5	Related Works	79
5.6	Conclusion	80
6	Policy Optimization with Horizon Regularized Advantage	81
6.1	Introduction	81
6.2	Related Work	84
6.3	Backgrounds	85
6.3.1	Contextual Markov Decision Process	85
6.3.2	Proximal Policy Optimization	86
6.4	Methodology	86
6.4.1	Horizon Regularized Advantage	87
6.4.2	Regularization through Hyperbolic Approximation	89
6.5	Experiments and Results	90
6.5.1	Network Architecture and Training	90
6.5.2	Evaluation Benchmarks	91
6.5.3	Generalization Performance on Test Distribution	92
6.6	Analysis and Ablations	93
6.6.1	Comparing Statistical Uncertainty	93
6.6.2	Analyzing the Computational Overhead	94
6.6.3	Assessing the Generalization Gap	95
6.6.4	Analyzing the Auxiliary Task of Multi-horizon Learning	95
6.6.5	Comparison with Different Gamma	96
6.7	Conclusion	97
7	Conclusion	98
7.1	Limitations	99

7.2	Directions for Future Work	100
7.2.1	Neural Architecture Search for Optimal Separation	100
7.2.2	Enabling Improved Contrastive Learning	100
7.2.3	Connecting Hazardous and Contextual MDPs	101
7.2.4	Mixture-of-Experts for Multi-horizon Advantage Estimates	101
7.2.5	Combining Multiple Improvements	102
	Bibliography	103

List of Figures

3.1	Comparison of architectures: (left) a fully shared network for policy and value, (middle) two explicitly separate networks for policy and value, and (right) our proposed partially separated network for policy and value function.	20
3.2	Details of our proposed architecture. The separated blocks for the policy and value function (at the top) incorporate an attention module (shown as shaded blocks) inside the residual blocks as well as one before the first convolutional layer and one after the max pooling layer. The shared blocks (at the bottom) are similar to the IMPALA CNN architecture ¹ . They do not have an attention module.	24
3.3	Test performance of PPO ² , IDAAC ³ , and our proposed APDAC over eight Procgen environments. Means and standard deviations are calculated over 10 trials, each with a different seed.	28
3.4	Training and test performance of PPO ² , IDAAC ³ , and proposed APDAC over four Procgen environments. Means and standard deviations are calculated over 10 trials, each with a different seed. The dotted lines refer to the training reward while the solid line refers to the test reward. In the case of APDAC, the gaps between the dotted line and solid line (green) are similar to the fully decoupled approach, IDAAC (orange) and both are better than PPO (blue).	31

3.5	Ablation study of the proposed method APDAC with the standard baseline PPO (no decoupling, no attention) and the ablated version PDAC (partially decoupled but without attention) on the test distribution for four Procgen environments. Means and standard deviations are calculated over 10 trials, each with a different seed.	31
3.6	Visualization of the value and policy representation for PPO ² , IDAAC ³ , the ablation, PDAC, and the proposed approach, APDAC, on an input sample from three Procgen environments. The red regions correspond to the higher value of gradients while the blue regions correspond to the lower value. For both policy and value, compared to other approaches, APDAC exclusively prioritizes informative spatial regions (red marked) and constructs asymmetrical representation.	33
3.7	Different network architectures highlighting the extent of decoupling between policy and value networks.	35
3.8	PPO normalized scores for the four architecture variants across all the 16 environments in the Procgen benchmark.	40
3.9	(Left) Crafter score for the four architectures with different degrees of separation for policy and value; (right) Comparison of the reward achieved by each agent during 1M timestep. In terms of both metrics, <i>late separation</i> performs better.	40
3.10	Test performance for fully shared, early separated, lately separated, and fully separated architecture across all 16 environments from the Procgen benchmark. Mean and standard deviation are calculated over 5 trials with different seeds.	43
3.11	Interquartile Mean (IQM) with 95% confidence interval for PPO (fully shared) normalized test rewards for all the 16 environments from the Procgen benchmark.	44

3.12	Learned representation of the policy and value using lately separated and fully separated architectures for the <i>Plunder</i> environment (at two different timesteps). The red-marked regions correspond to the higher value of gradients while blue regions correspond to the lower value of gradients. The learned policy representations using the partial late separation approach get biased with the value function and primarily look at the whole life bar as opposed to the end mark focused by the policy representation learned by the fully separated approach.	45
3.13	Test performance of the <i>early separation</i> architecture and its two variants on four environments from the Progen benchmark. <i>Early separation joint</i> , the baseline, uses the loss function as defined in eq. 3.8, <i>early separation phasic</i> uses separate optimization phases for policy and value loss, <i>early separation adv head</i> incorporates an extra advantage head in the policy subnetwork. Results show that neither of these two extensions alone can improve the performance of the <i>early separation</i> architecture.	46
4.1	Instability in the CURL ⁴ performance with different coefficients for the auxiliary contrastive loss. The experiments are for "crop" data augmentation in Progen. It shows how much the contrastive loss coefficient hyperparameter impacts the performance.	52
4.2	Test performance of non-contrastive approach SODA ⁵ with different combinations of the hyperparameter RL updates per iteration and projection dimension. The experiments are for "crop" data augmentation. This reveals that the performance highly is sensitive to the hyperparameters.	52
4.3	Overview of the proposed non-contrastive RL approach. We directly use the raw sample x for the RL objective and use the augmented one x' only for self-supervised representation learning. RAIR does not use any projector or predictor heads.	53

4.4	Similar positive and negative pair with the anchor (current observation) . . .	54
4.5	Test performance of RAIR (Ours), DrAC ⁶ , CURL ⁴ , SODA ⁵ , and PPO ⁷ for "crop" data augmentation in four Procgen environments during training. Mean and std are calculated over 5 trials.	57
4.6	Test performance of RAIR (Ours), DrAC, and PPO for "color jitter" data augmentation in four Procgen environments.	57
4.7	PPO normalized test score of RAIR (Ours), DrAC, CURL, and PPO for "crop", "grayscale", "color jitter", and "cutout" data augmentation for all 16 Procgen environments. Each agent is trained on 200 training levels for 25M timestep and then evaluated on 100 episodes. Means are calculated over 5 trials for every environment.	59
4.8	PPO normalized test performance of RAIR (Ours), DrAC, CURL, and PPO for "crop", "grayscale", "color jitter", and "cutout" data augmentation across all 16 Procgen environments. Mean and std are calculated over 5 trials with different seeds.	60
4.9	Test performance of our proposed cosine similarity-based invariance learning vs. variance-invariance-covariance based regularization method (VICReg) and its different ablations (removing one term at a time). Means and standard deviations are calculated over 5 trials.	60
4.10	Analyzing the impact of batch size.	61
4.11	Analyzing the impact of encoder learning rate while keeping the learning rate of the optimizer that optimizes RL objective (which is set to 0.0005).	61
5.1	Distribution of task completion time or episode lengths of four Procgen en- vironments for a fully trained PPO agent estimated based on 1000 episodes randomly sampled from the 1 to 200 training levels vs. only level 1, level 100, and level 1000. The difference in the distribution over multiple levels vs. a single level is clearly visible.	65

5.2	Details of the architecture used and the overall approach. Each block shown on the left side is almost identical to the IMPALA CNN architecture ¹ . Our calculation takes place based on the value prediction.	73
5.3	Test performance of our proposed Augmented Value Targets (AVaTar) and the standard PPO on Procgen environments.	75
5.4	Comparison of the generalization-specific SOTA approach UCB-DrAC ⁶ and proposed corresponding Augmented Value Targets (AVaTar) version <i>AVaTar + UCB-DrAC</i> over four Procgen environments.	75
5.5	Comparison of Interquartile Mean (IQM) of the Min-Max normalized score across all 16 Procgen environments. Both AVaTar versions outperform PPO ⁷ and UCB-DrAC ⁶ . The colored region shows the 95% confidence interval. . .	76
5.6	(Left) Crafter score of our AVaTar compared to standard PPO ⁷ and popular Rainbow ⁸ along with other approaches; (right) Comparison of the reward achieved by each agent during 1M timestep. Proposed AVaTar outperforms other approaches considering both metrics. We use 10 different seeds for our experiments.	76
5.7	Test performance of standard PPO with the increasing number of training contexts (levels) and our proposed Augmented Value Targets (AVaTar) with only 200 contexts over eight Procgen environments. It is evident that while training on 200 levels AVaTar can achieve performance gain that is even better than the PPO with 2000 training levels.	77
5.8	Test performance of our proposed Augmented Value Targets (AVaTar) with the varying number of gammas $n_\gamma = \{5, 10, 20\}$. We observe that the performance differences with varying gamma are marginal compared to the performance gain with respect to the fixed gamma (the standard PPO).	77

5.9	Comparison of proposed Augmented Value Targets (AVaTar) with discount regularization. Discount regularization uses a fixed but slightly lower discount factor as a regularizer ⁹	78
6.1	Distribution of episode lengths for a fully trained PPO agent estimated based on 1000 episodes randomly sampled from the test levels. Left: the agent is trained on 200 levels. Right: the agent is trained on 5000 levels; this denotes the inherent uncertainty and variation in completion time across levels (contexts) irrespective of the agent’s expertise.	82
6.2	Details of our proposed architecture. Each <i>Shared Block</i> is identical to the IMPALA CNN architecture ¹ . The <i>Critic Head</i> predicts state-values $V_{\pi}^{\gamma_i}$ for n_{γ} different discount factors. These exponentially discounted state-value predictions are then used to calculate the corresponding advantage estimates and the horizon regularized advantage function.	87
6.3	Test performance of proposed average advantage-based horizon regularization (HRA-Avg) and hyperbolic discounting-based horizon regularization (HRA-Hyp) against PPO on Procgen environments.	90
6.4	Test performance of APDAC and its counterpart that uses horizon regularized advantage through hyperbolic discounting. Means and standard deviations are calculated over 5 trials.	91
6.5	Performance comparison of proposed HRA-Avg and HRA-Hyp with PPO across all 16 environments using IQM and Mean of PPO normalized score.	92
6.6	(Left) Crafter score of our HRA-Avg and HRA-Hyp compared to standard PPO and Rainbow along with other approaches; (right) Comparison of the reward achieved by each agent during 1M timestep. In terms of both metrics, both versions of our proposed approach perform significantly better.	93
6.7	Evaluation of PPO and proposed HRA on two Minigrid environments. HRA significantly outperforms standard PPO.	93

6.8	Train and test performance of PPO, HRA-Hyp, and HRA-Avg for four Proc- gen environments. Means and standard deviations are calculated over 5 trials, each with a different seed.	94
6.9	Test rewards of proposed HRA-Avg compared to traditional exponential discounting- based PPO trained on the increased number of levels. HRA-Avg has been trained on only 200 levels.	95
6.10	Test performance of the auxiliary task of learning over multiple horizons that calculates the advantage only with the largest gamma vs. our hyperbolic advantage estimation that combines multiple advantage estimates.	96
6.11	Analysis of test scores for exponential discounting using different discount factors γ vs. horizon regularization.	96

List of Tables

3.1	Scores on test levels after training on 25M environment steps for all 16 environments. Values are averaged over 10 trials, each with a different seed. . . .	30
3.2	Comparison of number of required convolutional layers and computational time for different approaches	30
3.3	Comparison of computational time and the number of convolutional layers .	42
4.1	Components used in proposed RAIR compared to CURL and SODA.	55
4.2	Average PPO-Normalized Return (%) of different approaches on test levels across all 16 environments. Evaluated with 5 trials for each environment, each with a different seed.	58
5.1	List of hyperparameters used for AVaTar	74

Acknowledgments

First of all, I would like to express my deep gratitude to the Almighty for providing me with the opportunity and strength to complete this PhD. I am immensely grateful to many individuals who have supported and inspired me throughout this challenging journey.

I would like to thank my advisor, William Hsu, for his unwavering support and guidance during my PhD. I am deeply grateful for the independence he allowed me in selecting research directions and pursuing my passions. Over the past six years, his continuous encouragement and invaluable advice immensely helped me to make progress in my research. His generosity, kindness, and empathy have profoundly influenced me, both as a scholar and as a human being. I would also like to express my gratitude to my dissertation committee members Doina Caragea, Mitchell Neilsen, and Bala Natarajan for their constructive feedback that has been instrumental in shaping my dissertation.

I would like to thank my collaborators and labmates from the K-State Knowledge Discovery in Databases (KDD) lab. I would like to thank Avishek Bose who has always been there to listen to my problems and provide suggestions. I would like to thank Raja Farrukh Ali for his collaboration, guidance, and inspiration. I greatly admire his intelligence and organizational skills, which positively impacted my own development. I would also like to thank my peers Huichen Yang, Ademola Okerinde, Rushrukh Rayan, Lei Luo, and Vahid Behzadan for their support and advice. I had the opportunity to mentor and collaborate with several talented undergraduates at the KDD Lab. I would like to thank Creighton Glasscock, Ashley Rediger, Giovanni Poggi-Corradini, Kevin Duong, Ethan Khoury, Mason Vick, and Tatenda Sekabanja for their contributions to several research projects that are part of this dissertation.

Furthermore, I would like to express my heartfelt gratitude to some incredible mentors who have guided me in shaping my research and career paths. I was fortunate to have Peter Stone as my mentor at the AAMAS Doctoral Consortium, and I deeply appreciate his

valuable insights. I also extend my thanks to Mohammad Khodabakhsh, who advised me during my internship at DEKA Research and Development. His support and trust offered me a perfect environment for growth. Additionally, I am thankful to my friends at K-State and UTA who have made the journey much more enjoyable.

I am especially grateful to my mother who instilled in me a passion for a PhD. I am forever indebted to my mother and father for fostering my desire to learn and for providing me with the best education at the early stage of my life. My heartfelt thanks also go to my elder sister and younger brother, who have supported me in countless ways since childhood. I am equally thankful to my extended family members for their encouragement.

Finally, I would like to thank my wonderful wife who has been my strongest support during this grueling phase of my life. Her willingness to discuss research ideas, share insights, and offer advice has been invaluable. Her unconditional love and care have made this journey much easier for me. And to my son, I am eternally grateful. Your smile has been a remedy for my distress and a constant source of inspiration and revitalization. I wish I could spend more joyful time with you. Thank you for being part of this incredible journey.

Dedication

To my loving mom, dad, wife, and son Aronno.

Chapter 1

Introduction

Reinforcement learning (RL) provides a framework for sequential decision-making in complex systems. RL agents distinguish themselves from other Artificial intelligence (AI) and machine learning (ML) agents by learning tasks in a manner akin to humans and animals, through interaction with their surroundings and achieved feedback Sutton and Barto¹⁰. In recent years, RL agents have proved more capable than anticipated. RL agents have performed at a human level on a variety of control tasks such as object manipulation and playing Go^{11–13}. They can master complex tasks by exploring and learning in a training environment given a large number of samples. Once deemed totally impractical and unsafe, advances now suggest that well-trained RL agents can be a plausible alternative to agents using traditional learning and control techniques.

Humans possess a remarkable ability to generalize knowledge and apply it in new contexts, enabling them to tackle unfamiliar tasks effectively. This capability stems from cognitive flexibility, which allows humans to extract underlying principles from past experiences and recognize patterns across different situations. When faced with new tasks or novel challenges, humans draw upon previously learned strategies and concepts, adapting them to fit the current context. Using language, memory, and problem-solving abilities facilitates the human ability to analyze, think abstractly, and transfer skills from one domain to another. This cognitive versatility not only aids in learning to complete unfamiliar tasks but also allows

humans to navigate the complexities of ever-changing environments. As opposed to humans who can transfer their learning from one context to another, RL agents are still in their infancy¹⁴. Deploying artificially intelligent agents in real-world applications will require significant generalization and faster adaptation capabilities for similar but unseen scenarios or conditions¹⁵. However, the generalizability of this magnitude has yet to be achieved in standard RL algorithms^{2;16–18}. This dissertation aims to develop efficient reinforcement learning agents that can generalize to new contexts beyond the training ones.

Deep reinforcement learning (DRL) parameterizes the learning agent through deep neural networks. Much like any deep neural network-based function approximator, deep RL agents are prone to overfitting^{19;20}. They tend to memorize the training data distribution, particularly the training episodes in RL^{3;14}. Until recently, deep RL algorithms were trained and tested on the same singleton environment. Thus, the issue of overfitting was not consistently observed and measured, but instead tacitly tolerated. Recent research reveals the potential side effects of such a limited approach to evaluation in assessing generalization²¹. These findings have inspired the development of benchmarks that provide a better way to quantify an agent’s ability to generalize^{2;22}. The emergence of such benchmarks does not require real-world deployment of the RL agents but offers an opportunity to train and test RL agents on different sets of similar instances of the same task to effectively evaluate generalization^{3;16}.

1.1 Problem Statement

Generalization is aimed at enabling real-world application of deep RL²³. The real world is dynamic and ever-changing, and training on all possible variations of a task is infeasible, so training strategies must allow RL agents to learn how to perform well in new situations without prior exposure to that exact situation. Thus, the core of the objective requires mitigating overfitting to the training environments^{14;21}. We must design RL methods that are robust to the variations in the environments and can transfer learning to unseen but similar scenarios during deployment.

In this thesis, we consider the problem of zero-shot generalization in RL. The goal is

to learn a policy that can perform better in a zero-shot manner on a collection of new environments that differ from those on which the policy was trained. The term *zero-shot* signifies that no additional training or data from the new environment is allowed¹⁴. Thus, the solution should exclude any domain adaptation or any type of meta-RL approach that goes through an adaptation phase during or just before deployment.

Recent benchmarks characterized by the Contextual Markov Decision Process (CMDP) framework can generate different episodes corresponding to different variations of the environment while keeping the task objective the same^{2;24}. This dissertation considers the problem of generalization to unseen scenarios or levels of procedurally generated environments given exposure to a limited number of levels during training. The levels vary in terms of background, dynamics, game assets, and the attributes of the entities such as position, spawn time, shape, and color; however, all levels share the same goal. Thus, the RL agent must have significant generalization capability to ensure a robust policy that performs well on levels, episodes, or scenarios that are not encountered during training.

1.2 Contributions

Fundamentally, a deep RL agent comprises two aspects: the architecture or structure of the agent and the learning strategy or algorithm used by the agent. The architectural aspect may include the design of the neural network that represents the policy and value functions such as the number of layers in the network, the activation functions used, or the type of the neural network (Convolutional Neural Network (CNN) versus Recurrent Neural Network (RNN)). Architectural aspects may also consider high-level architectural issues such as modularity of the architecture or incorporating inductive bias through network architecture. On the other hand, the learning algorithm doesn't necessarily denote only the training regime to update the network parameters' weights. Rather algorithmic aspects may include uncertainty modeling, variance reduction in value estimation, auxiliary losses, exploration techniques, reward shaping, etc. that improve the agent's ability to learn. In this dissertation, we seek solutions to learn generalizable policies from both aspects.

This research contributes new architectural and algorithmic advances from two important dimensions of reinforcement learning: representation learning and value estimation. Both are pivotal to developing effective RL methods. In high-dimensional state space settings, the latent representations or features learned by the deep neural network are crucial. The policy is optimized on top of learned representations, so generating good latent representation is essential. In RL, representation learning doesn't entail merely encoding the information within a state but learning the dynamics of the state transitions and features conducive to decision-making. While rewards are the primary feedback in the RL system, value estimates (the predictions of expected long-term rewards) are significant in making and evaluating decisions. Value function defines what is good in the long run, not immediate rewards. The value function specifies the total cumulative reward that the agent can expect to achieve starting from a particular state. Based on the perception of value estimates, action choices are made. In developing new methods, this research pays central attention to the intricate relationship between the policy and the value function. The specific contributions of this dissertation are as follows:

Efficient Neural Architecture for Actor-Critic: I show that partial decoupling of policy and value networks effectively and efficiently learns representation that improves generalization in RL. I leverage a layer block level of abstraction to partially decouple the policy and value networks. This work reveals that partial decoupling and associated attention mechanisms help to achieve the necessary policy-value representation asymmetry needed for generalization in RL. I simultaneously address the issue of limited generalization due to overfitting in a fully shared actor-critic network and the added complexity of fully decoupled ones. I conduct a comprehensive analysis to investigate the extent of decoupling necessary to partial separation. Further, I analyze the quality of the learned representations.

Augmentation Invariant Representation Learning: I develop a representation learning framework for RL that distinctly considers both the augmentation invariant representation learning task and the downstream RL task. More specifically, this research com-

bines non-contrastive self-supervised embedding learning and RL objective optimization in a minimalist way cf. Fujimoto and Gu²⁵, seeking to focus on contrastive and non-contrastive hyperparameters, simplify the architectural framework, and more clearly attribute reward gains. I demonstrate the brittleness of the contrastive and non-contrastive loss-based RL algorithms and how slightly modifying the hyperparameters significantly impacts these reward gains. To alleviate this problem, I reduced the dimension of the hyperparameter space by discarding redundant architectural components and improved the stability of the algorithms by altering the invariant loss update sequence. This presents a simpler way of leveraging data augmentation that can effectively compare to similar existing methodologies.

Augmented Values Targets: I identify a high degree of variation and uncertainty in the episode length or task completion time in the case of procedurally generated environments, a characteristic that is also prevalent in many real-world domains. This negatively affects value function generalization especially when a fixed discount factor is used in the RL algorithm. I argue that a fixed discount factor introduces a very restricted inductive bias. Thus, I propose generating multiple value targets using randomly sampled multiple discount factors and optimizing the value function towards the average of those targets.

Horizon Regularized Advantage for actor-critic: I introduce a novel advantage estimation technique that considers advantage estimates learned over multiple horizons. I draw motivation from uncertainty in task completion time identified in the previous contribution. While learning from a limited number of task instances with little variation in task duration, the learned value estimates get biased to those episode lengths observed during training. The proposed approach thus learns value estimates for different horizons using multiple discount factors and combines the corresponding advantage estimates through simple averaging or non-linear mixing that approximates hyperbolic discounting.

This dissertation is organized as follows. Chapter 2 presents basic mathematical concepts and notations of RL, general algorithms prevalent in RL, and an overview of procedurally

generated environments. Chapter 3 introduces the hybrid actor-critic architecture that partially decouples the policy and value function and deploys an attention mechanism. Chapter 4 investigates representation learning through data augmentation for RL. It introduces a minimalist approach to learning augmentation invariant representation amenable to generalization. Chapter 5 studies the problem value estimation under uncertainty in task completion time and introduces the value target augmentation approach to learning robust value estimates. Chapter 6 introduces a novel advantage estimation technique that uses a mixture of multiple advantages learned over multiple horizons. Finally, Chapter 7 summarizes the key findings, discusses limitations, and suggests direction for future research. The contributions presented in chapters 3, 5, and 6 are already published in peer-reviewed conferences. The research results in Chapter 4 have been presented at a peer-reviewed workshop, and a manuscript covering those results is currently under review at a conference. The list of published articles follows.

List of Contributions

- **Nasik Muhammad Nafi**, Creighton Glasscock, and William Hsu. Attention-based partial decoupling of policy and value for generalization in reinforcement learning. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 15–22, 2022.²⁶
doi: <https://ieeexplore.ieee.org/abstract/document/10069416>.
Also presented at the Deep RL workshop at NeurIPS 2021²⁷.
- **Nasik Muhammad Nafi**, Raja Farrukh Ali, and William Hsu. Analyzing the sensitivity to policy-value decoupling in deep reinforcement learning generalization. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2024.
doi: <https://www.ifaamas.org/Proceedings/aamas2023/pdfs/p2625.pdf>
IJCNN 2024 proceedings are in press. Also presented at AAMAS 2023 as an Extended Abstract²⁸ and at the Deep RL workshop at NeurIPS 2022²⁹.

- **Nasik Muhammad Nafi** and William Hsu. Reinforcement Learning with Augmentation Invariant Representation: A Non-contrastive Approach. In *NeurIPS 2023 Workshop on Generalization in Planning (GenPlan)*, 2023.³⁰
doi: <https://openreview.net/forum?id=fYUWJHNkdA>
- **Nasik Muhammad Nafi**, Giovanni Poggi-Corradini, and William Hsu. Policy optimization with augmented value targets for generalization in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2023.³¹
doi <https://ieeexplore.ieee.org/abstract/document/10191507>
- **Nasik Muhammad Nafi**, Raja Farrukh Ali, William Hsu, Kevin Duong, and Mason Vick. Policy Optimization using Horizon Regularized Advantage to Improve Generalization in Reinforcement Learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1427–1435, 2024.³²
doi: <https://dl.acm.org/doi/abs/10.5555/3635637.3663002>
An earlier version was presented at Decision Awareness in RL workshop, ICML 2022³³.

In addition to the publications directly included in this dissertation, I note here some other contributions from my PhD studies. While these additional publications are not part of this dissertation, they are relevant to the investigation of RL and its application.

- Raja Farrukh Ali, Kevin Duong, **Nasik Muhammad Nafi**, and William Hsu. Multi-horizon learning in procedurally-generated environments for off-policy reinforcement learning (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 13, pages. 16150-16151. 2023.³⁴
- **Nasik Muhammad Nafi**, Avishek Bose, Sarthak Khanal, Doina Caragea, and William H. Hsu. Abstractive text summarization of disaster-related documents. In *ISCRAM 2020 Conference Proceedings–17th International Conference on Information Systems for Crisis Response and Management*, 2020.³⁵

All source codes can be accessed through the author’s GitHub account.¹

¹<https://github.com/NasikNafi/>

Chapter 2

Background

In this chapter, we briefly present the mathematical foundations of RL, general categories of RL methods, and an overview of modern procedurally generated environments. This will serve as a basic introduction to the problem settings, concepts, techniques, and benchmarks used in the subsequent chapters.

2.1 Mathematical Formulation

2.1.1 Markov Decision Processes

Markov Decision Process (MDP)³⁶ is the mathematical framework that formalizes the underlying environment in which an RL agent makes decisions³⁷. MDPs can be represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(s_{t+1}|s_t, a_t)$ represents the transition probability to reaching state $s_{t+1} \in \mathcal{S}$ from state $s_t \in \mathcal{S}$ by taking action $a_t \in \mathcal{A}$, $\mathcal{R}$ is the reward function that defines $\mathcal{R}(r_{t+1}|s_t, a_t)$ as the reward at state s_t after taking action a_t , and $\gamma \in [0, 1)$ is the discount factor. The objective of the RL agent is to maximize the expectation of the total discounted return $\sum_t \gamma^t \mathcal{R}(s_t, a_t)$ over the observed trajectories generated following the agent policy $\pi(a|s)$, that maps states to actions, and the transition function $\mathcal{T}$. The optimal policy $\pi^*(a|s)$ is considered as the solution of the MDP.

2.1.2 Contextual Markov Decision Processes

Contextual Markov Decision Process (CMDP) is an augmented form of the basic MDP¹⁴. CMDPs are represented as tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, \mathcal{R}, \mu_C, \mu_S)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{C}$ is the context space, $\mathcal{T}(s'|s, a)$ is the transition function, $\mathcal{R}$ is the reward function, μ_C is the context distribution, and μ_S is the context-dependent initial state distribution. At the beginning of each episode, a context is sampled according to $c \sim \mu_C$. Then an initial state is sampled according to $s_0 \sim \mu(\cdot|c)$ and the successive states within that episode are sampled based on $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t, c)$. Let d_π^c denote the state distribution resulting from acting with the policy π in context c .

In a CMDP setting, the goal of an RL agent is to learn a policy π that maximizes the expected return over all possible contexts such that $\mathcal{G} = \mathbb{E}_{c \sim \mu_C, s \sim d_\pi^c, a \sim \pi(s)}[\mathcal{R}(s, a)]$. The context parameters generally take the form of an initial seed, identifier, or parameter vector that determines the variation of an episode. Those parameters are not observable by the agent. The context remains the same within an episode but varies between episodes. During training, the agent is exposed to a limited number of contexts that is far smaller than the whole set of contexts on which the agent is being tested. The challenge is learning from those limited contexts and generalizing over unseen ones.

2.2 Reinforcement Learning Methods

The fundamental goal of an RL agent is to learn to make the optimal decision or action in a certain state or situation. The agent gradually refines its behavior to favor optimal actions. It does so by updating its behavior so that actions leading to better outcomes are selected more frequently and actions leading to worse outcomes are selected less frequently. A set of "rules" that defines which action should be selected in each state is referred to as the agent's policy. The outcomes are governed by a task-specific reward function. The agent receives scalar feedback, known as reward, at each time step according to the reward function. The core objective of an RL agent is to maximize the expected cumulative rewards over a given

time frame. It is interesting to note that optimizing this single objective is sufficient to acquire the intended behavior. If the agent optimizes its policy such that the rewards are maximized "eventually" instead of "greedily", then it can be close to being optimal.

2.2.1 Value-based Methods

As the name suggests, this category of method performs action selection directly based on value judgments, i.e., how good a state or action is in the long run¹⁰. Thus, the core of these methods lies in estimating the value function. The value function $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$ is defined as the expected cumulative reward (also called expected return) over the trajectory starting from a particular state and following policy π :

$$V^\pi(s) = \mathbb{E} \left[\sum_{t=0}^T \gamma^t r_t | s_0 = s \right]. \quad (2.1)$$

This is referred to as the state value function because this specifies the long-term desirability of a state. Similarly, the state-action value function, known as the Q-function $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, can be defined as

$$Q^\pi(s, a) = \mathbb{E} \left[\sum_{t=0}^T \gamma^t r_t | s_0 = s, a_0 = a \right] \quad (2.2)$$

Further, the advantage function $A^\pi(s, a)$ for a state-action pair can be defined as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. All the value functions can be recursively defined using Bellman's equation. For state value function: $V^\pi(s_t) = r_t + \gamma V^\pi(s_{t+1})$. Similarly, for the Q^π function, it can be defined as:

$$Q^\pi(s_t, a_t) = r_t + \gamma \max_{a \in \mathcal{A}} Q^\pi(s_{t+1}, a) \quad (2.3)$$

Leveraging this recursive definition, the state-action value function can be iteratively estimated from the sequences of observations an agent makes in an environment. Using a batch of state transition data, a Q-function $\hat{Q}(\cdot, \theta)$, parameterized by θ , can be approximated by minimizing the loss $\mathcal{L} = (\hat{Q}(s_t, a_t, \theta) - Y_t)^2$. where $Y_t = r_t + \gamma \max_a Q(s_{t+1}, a, \theta)$. Such a temporal-difference (TD) approach¹⁰ to learning the optimal state-action value is known as

Q-learning³⁸. The value update equation used in the algorithm follows.

$$Q^{new}(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (2.4)$$

Given enough samples and time, the state-action value function will converge to the optimal $Q^*(S_t, A_t)$. This optimal state-action value function at each state defines the optimal policy that selects action based on the best value:

$$\pi^*(\cdot|s) = \arg \max_a Q^*(s, a) \quad (2.5)$$

These methods are off-policy as they can use samples collected using any earlier policy to update the value function. This characteristic often makes them more sample-efficient than the on-policy policy-based approaches.

2.2.2 Policy-based Methods

The policy-based methods directly learn a parameterized policy π_θ to select actions without requiring a value function¹⁰. The policy is defined as a function that maps states to actions to maximize the expected cumulative reward, which is the basic objective of RL. The expected return objective is defined as

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{eta}}}} \left[\sum_{t=0}^T \gamma^t r_t \right] \quad (2.6)$$

where τ represents a trajectory (sequence of states, actions, and rewards) generated following policy π_θ , γ is the discount factor, r_t is the reward at time step t , and T is the time horizon. The policy gradient theorem provides the gradient of the expected return with respect to the policy parameters θ :

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right] \quad (2.7)$$

where G_t is the return from time step t . Since it is a maximization problem, the policy parameters θ can be updated using the gradient ascent: $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\pi_{\theta})$, where α is the learning rate. The basic policy-gradient approach REINFORCE³⁹ uses the following update rule:

$$\theta_{t+1} = \theta_t + \alpha G_t \left[\frac{\nabla_{\theta} \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)} \right] \quad (2.8)$$

$\pi(A_t | S_t, \theta_t)$ denotes the probability that the policy parameterized by θ_t takes action A_t at state S_t . The policy-based approaches that directly use the expected return G_t suffer from high variance. A generic solution to reduce the variance in REINFORCE is to use a baseline $b(s_t)$:

$$\theta_{t+1} = \theta_t + \alpha (G_t - b(S_t)) \left[\frac{\nabla_{\theta} \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)} \right] \quad (2.9)$$

2.2.3 Actor-Critic Methods

As discussed in the previous section, policy-gradient methods often must use a baseline function to tackle the high variance issue. The baseline can be any function of the states. A useful baseline is the estimate of the state value and can additionally be learned as a separate parameterized function following a value-based method. Such an approach that learns both the policy (actor) and value (critic) functions is called the Actor-Critic method¹⁰, which is a hybrid method in the sense that it uses both value-based and policy-based paradigms. The update rule for actor-critic methods can be defined as

$$\theta_{t+1} = \theta_t + \alpha (G_t - \hat{V}(S_t)) \left[\frac{\nabla_{\theta} \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)} \right] \quad (2.10)$$

where $\hat{V}(\cdot)$ is the value function estimate. This equation can be written as:

$$\theta_{t+1} = \theta_t + \alpha \left(R_{t+1} + \gamma \hat{V}(S_{t+1}) - \hat{V}(S_t) \right) \left[\frac{\nabla_{\theta} \pi(A_t | S_t, \theta_t)}{\pi(A_t | S_t, \theta_t)} \right] \quad (2.11)$$

2.3 Selected Actor-Critic Algorithms

The previous section presents different types of RL methods. As described, the actor-critic method combines the strengths of both policy-based (actor) and value-based (critic) methods. Two commonly used actor-critic RL algorithms are Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO). SAC is an Off-policy algorithm that can learn from old samples, making it more sample-efficient. On the other hand, PPO is an On-policy algorithm that relies on the most recent samples collected by the current policy. This online nature of PPO makes it less sample-efficient. However, being an on-policy algorithm PPO offers better convergence. In this work, we use PPO as a baseline and mostly build our approach on top of the PPO. However, the proposed solutions can easily be integrated with SAC.

2.3.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a widely used actor-critic method⁷. While learning from high-dimensional image observation, as the policy and value function approximator, PPO generally leverages a shared neural network. Given that the network is parameterized by θ , PPO optimizes the following joint objective:

$$J_{PPO}(\theta) = J_{\pi}(\theta) - \alpha_v L_V(\theta) + \alpha_s S_{\pi}(\theta) \quad (2.12)$$

where $J_{\pi}(\theta)$ is the policy gradient objective, $L_V(\theta)$ is the value loss, α_v is the coefficient for value loss, $S_{\pi}(\theta)$ is the entropy bonus, and α_s is the coefficient for entropy bonus. PPO is primarily based on the Trust Region Policy Optimization (TRPO)⁴⁰ method. Both maximize the following surrogate policy objective:

$$J_{\pi}(\theta) = \hat{\mathbb{E}}_t[r_t(\theta)\hat{A}_t] \quad (2.13)$$

where $r_t(\theta) = \frac{\pi_{(\theta)}(a_t|s_t)}{\pi_{(\theta)_{old}}(a_t|s_t)}$ is the probability ratio between the new policy and the old policy, and $\hat{A}_t$ is the advantage estimate at timestep t . Unlike TRPO, PPO prevents excessively

large policy updates by clipping the value of $r_t(\theta)$ to the intervals of $[1 - \epsilon, 1 + \epsilon]$. The minimum between this clipped value and the original value of $r_t(\theta)$ is then selected as $r_t(\theta)$. Thus, PPO optimizes the following clipped surrogate objective for policy optimization:

$$J_\pi(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right] \quad (2.14)$$

2.3.2 Soft Actor-Critic

Soft Actor-Critic (SAC) is a popular RL algorithm suitable for solving continuous control tasks¹³. It is an off-policy approach that combines the benefits of entropy-based methods. SAC tackles the challenge of balancing exploration and exploitation by incorporating a maximum entropy framework. This encourages the agent to try out various actions by maintaining a level of randomness in its action selection. Thus, the agent can explore the environment more thoroughly. SAC simultaneously maximizes the expected cumulative reward objective and the entropy of the policy as follows.

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} (\gamma^t r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t))) \right] \quad (2.15)$$

where $\mathcal{H}(\pi(\cdot | s_t)) = -\sum_{a \in \mathcal{A}} \pi(a | s_t) \log \pi(a | s_t)$ denotes the entropy of the policy and α is the temperature parameter to balance reward and entropy maximization. SAC uses an adaptive scheme to control the exploration and exploitation trade-off through the temperature parameter. In practice, the KL-divergence between the policy and an exponential of the Q-value is minimized as the policy objective. SAC utilizes two critic networks (Q-function) that learn a soft Q-value. Two networks help to reduce overestimation bias. The minimum of the two Q-values is used for the value gradient and policy gradient.

2.4 Relevant Additional Concepts

2.4.1 Deep RL and Representation Learning

At the core of the RL algorithms lies the approximation of the value and policy function. These functions can be approximated using traditional linear or polynomial functions. However, advances in modern deep learning theory and the emerging necessity to learn from high-dimensional state space have led to the adoption of deep neural network-based function approximators in RL¹¹. The use of multiple layers of nonlinear transformations makes the neural network a powerful tool to represent and learn complex patterns from data. Thus, the use of deep neural networks allows deep RL to effectively handle the high-dimensional state and action spaces prevalent in real-world applications. Representation learning appears as the key strength of deep RL as it enables automatic learning of meaningful feature representations from raw data such as images, sensor data, or natural language. This ability to extract key features and organize them hierarchically enhances the decision-making and generalization capability of the deep RL agent. Using functional notation, we can write the function we want to approximate as $g : \mathcal{X} \rightarrow \mathcal{Y}$. It can be decomposed as $y = g_\phi(z)$, where z is the latent representation of the input x and the network is parameterized by ϕ . Then, $z = f_\theta(x)$ is the encoder function parameterized by θ that encodes the input x into low-dimensional feature representation z reducing noise and redundancy.

2.4.2 Discount Factor and Effective Horizon

In standard RL algorithms, the agent's objective is to maximize the sum of the discounted rewards over the future. Formally, the discounted return G_t is:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots, \quad (2.16)$$

where $0 \leq \gamma < 1$ is called the discount factor⁴¹. The discount factor γ determines the current value of the future rewards. If the immediate reward at time step k is r_k , then that reward

will be perceived as $\gamma^{k-1}r_k$ at the current time step. The horizon refers to how many steps into the future the agent takes into account and values the reward it receives. If $\gamma = 0$, then the agent becomes myopic and only maximizes immediate rewards. Conversely when $\gamma \rightarrow 1$, the agent values future rewards as well, making the agent far-sighted. A particular value of γ sets a fixed effective horizon till which point the rewards are considered significant⁴².

2.5 Procedurally Generated Environments

Popular benchmarks like the Arcade Learning Environment⁴³, MuJoCo⁴⁴, and DeepMind Control Suite⁴⁵ have driven significant algorithmic advancements in RL. These benchmarks provide singleton environments posing challenges like credit assignment, exploration in sparse reward settings, and learning from high-dimensional observations. A singleton environment such as this one means the environment does not change during training and testing. Thus, training in a singleton environment makes the RL agent susceptible to overfitting and failing to generalize when simple variations or perturbations are applied²¹. Generalization or the ability to perform in diverse situations is, however, a crucial attribute for real-world applications of RL such as industrial automation, human-robot interaction, and autonomous driving.

Procedurally generated environments have opened up a new avenue of research by providing the opportunity to train and test RL agents in diverse environments². Procedurally generated environments are often called multi-environments because they offer several different environments for the same task⁴⁶. All the variations in the environments are induced through procedural content generation (PCG). PCG algorithmically creates diverse and scalable content across episodes. Procedurally generated environments also provide a systematic way to evaluate generalization through the separation of the training and testing episodes by controlling the algorithmic parameters (such as seed or content vector). Examples of such benchmarks include recently released environments such as Sobokan⁴⁷, MiniGrid⁴⁸, CoinRun²¹, Obstacle Tower⁴⁹, Procgen², and Crafter²⁴.

2.5.1 Procgen

The Procgen benchmark², Developed by OpenAI, is designed to evaluate the generalization performance of RL agents. It provides a suite of 16 procedurally generated environments, each offering an infinite variety of levels. The procedural content generation leveraging random seeds ensures that levels are not identical. The levels differ in aspects such as visual appearance, dynamics, game assets, and the attributes of entities. Despite these variations, all levels share the same task objective within an environment. The variations and the unique challenges presented by each level make it harder for the agent to solve the task by memorizing the training levels. Thus, agents must develop robust policies to generalize across levels. The evaluation protocol is training the agent on 200 levels and testing the agent on the full distribution of levels. This evaluation strategy is effective in assessing the agent’s ability to adapt and perform well in unseen scenarios. All environments have a discrete 15 dimensional action space and the observation space is a $64 \times 64 \times 3$ RGB image. There are two training modes: *easy* and *hard*. The difference lies in the content generation and the hardness of the levels. According to the suggested evaluation protocol, in easy mode agents need to be trained for 25M timesteps before testing while the requirement is 200M timesteps for hard settings.

2.5.2 Crafter

Crafter²⁴ is a challenging benchmark that simulates an open-world survival game environment, providing a rich and diverse landscape for evaluating RL algorithms. In Crafter, each episode is uniquely configured with a distinct sequence of resources, terrain types, and creatures, requiring agents to develop generalization and long-term reasoning skills to succeed. The primary objective in Crafter is to accomplish multiple achievements such as crafting tools, collecting resources, building shelters, finding food, waking up after sleep, and surviving encounters with various creatures. Crafter offers a sparse reward signal composed of two main components. The primary component is a +1 reward when the agent unlocks a new achievement for the first time in the current episode. The secondary component pro-

vides a reward of -0.1 for losing each health point and +0.1 for each regaining a health point. The maximum number of health points is bounded by 9. Thus, this secondary reward component has a marginal impact on the episode’s total return. Other than generic expected return, Crafter introduces another evaluation metric called Crafter score which is the aggregated success rate for the individual achievements. Success rates are calculated for each achievement. It is defined as the fraction of the episodes in which the agent has successfully unlocked the achievement. The training protocol for Crafter is to train for 1 million timesteps, allowing them to explore the diverse possibilities within the environment. The evaluation is carried out based on the reward and score achieved within this budget of 1M timesteps. The observation space of the Crafter is $64 \times 64 \times 3$ image and it has 17 actions in the action space.

2.5.3 MiniGrid

MiniGrid⁴⁸ is a minimalist grid-based environment specially designed to test navigation and planning capabilities through goal-driven tasks. The procedural content generation in MiniGrid ensures diverse maze maps and position distribution of objects such as doors, keys, and boxes. This dissertation uses two specific MiniGrid environments: Multi Room and Four Rooms. These environments present varying levels of complexity, requiring agents to navigate through multiple connected rooms. Randomly generated goal locations and initial positions in each episode offer more challenges for the agent. The agent achieves rewards only for succeeding in the environment. In MiniGrid, agents are generally trained for 1M timesteps. It has a discrete action space with 7 actions. The default observation is a dictionary with three items: “image”, “direction”, and “mission”. This dissertation only considers the image item of the observation.

Chapter 3

Partial Decoupling of Actor and Critic

3.1 Introduction

In recent years, deep reinforcement learning (RL) algorithms have shown significant performance improvement on a variety of control tasks^{11–13}. They can master any complex tasks by exploring and specializing in a training environment given a large number of samples. However, current RL algorithms often suffer from poor generalization performance when applying the learned policy to an unseen scenario^{2;16–18}. Raileanu and Fergus³ show that policy-value representation asymmetry is an underlying cause for poor generalization performance in shared actor-critic architectures. Value estimation in a particular state depends on instance-specific features; however, learning the policy only depends on task-specific features³. Ignoring this asymmetry while learning a joint representation of both policy and value through a shared network limits the agent’s capacity to learn a generalizable policy. Combining the two representations through a shared network makes the policy to be unnecessarily biased toward the value features. Policy-value representation asymmetry also suggests that value estimation requires more information than what is needed to learn an optimal policy³. Thus the learned policy overfits the training instances and performs poorly in terms of generalization.

As the shared representation of policy and value function may lead to overfitting³, an

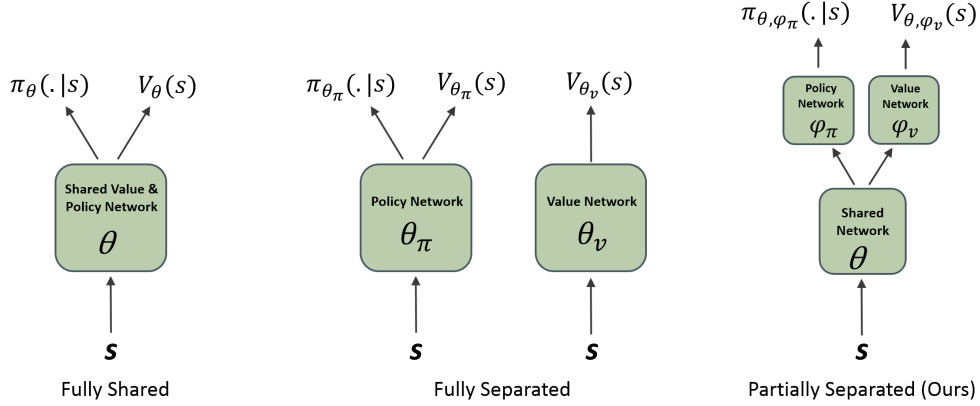


Figure 3.1: Comparison of architectures: (left) a fully shared network for policy and value, (middle) two explicitly separate networks for policy and value, and (right) our proposed partially separated network for policy and value function.

alternative to shared representation is to separate the policy and the value networks^{3,16}. This helps to disentangle the features necessary to properly approximate the value and policy function. However, the policy function cannot be learned in isolation, via stand-alone training. It requires gradients from the value function to learn the optimal policy³. Thus, additional measures must be taken to improve the policy network, which increases the complexity of overall training. Moreover, the two networks entail increased memory requirements and training time.

As shown in Figure 3.1, we design a hybrid actor-critic architecture that provides an excellent alternative to the two extreme approaches, fully shared and fully separate networks, by combining the benefits of both while mitigating their disadvantages. We propose *Attention-based Partially Decoupled Actor-Critic (APDAC)* that shares some early layers of the network while separating the later (downstream) ones into policy and value sub-networks fed by the shared blocks. This partial decoupling recognizes the asymmetry in the policy-value representation and enables distinct high-level feature learning for both policy and value. Additionally, we deploy an attention mechanism in the two separate branches to facilitate the extraction of relevant features for the policy and value functions. In order to perform better on new unseen task episodes, learning minimal and compact representations is crucial³. A minimal set of features allows a deep RL network to avoid spurious correla-

tion between generic attributes of the episodes (e.g., background color) and the value/policy function. Consequently, this minimal representation helps to better capture the relation between task-relevant attributes and the value/policy function. Our ablative experiments demonstrate how attention within these sub-networks accounts for gains over the state-of-the-art baseline by focusing on a minimal set of spatial features. We attribute the benefits of our approach to the hierarchical representation of features and the ability of attention mechanisms to effectively identify the components of an input pertinent to the optimization task. Further, we produce visualizations of the policy and value that reveal crucial insights into the learned policies and value functions.

Key novel contributions of this work are as follows: (i) we introduce a new hybrid architecture, as shown on the far right of Fig. 1, that consists of decoupled policy and value subnetworks (downstream layers) with a shared set of upstream or early layers; (ii) we develop an integrated attention mechanism to encourage policy-specific and value-specific feature learning with minimum overhead; (iii) we demonstrate qualitative and quantitative performance improvement compared to state-of-the-art (SOTA) methods on the Procgen benchmark.

3.2 Related Work

Many recent works have established that lack of generalization is a systemic problem in deep reinforcement learning and popular algorithms tend to overfit to the environment, resulting in models that seem merely to memorize surface-level details of the environment rather than learn generalizable skills^{17;18;50;51}. Existing solutions to the generalization problem include L2 regularization⁵², dropout⁵³, data augmentation⁵⁴, batch normalization⁵³, and hyperbolic discounting³³. It has been shown that the performance of generalization in RL increases with the number of convolutional layers while learning directly from image observation². As established in⁵⁴, Procgen is a testing suite that uses procedural content generation to benchmark generalization to greater effect than traditional benchmarks. Procgen became popular in recent works advancing generalization in deep reinforcement learning^{3;55-57}. As

discussed in³, sharing features between policy and value functions can lead to overfitting, reducing a model’s ability to generalize to new, unseen environments. In contrast to the previous methods,^{3;16} make use of fully disconnected policy and value functions. This provides greater generalization and sample efficiency than earlier counterparts. However, this performance comes at a price; more parameters than previous approaches require more computing power. In addition, certain Procgen environments require specific hyperparameters to produce reported performances. Our method provides results consistent with those in³ with fewer convolutional layers while reducing the need for hyperparameter tuning.

Literature exploring the potential of attention mechanisms in neural networks has found success across a wide array of domains, including natural language processing and vision, both as part of convolutional layers and as stand-alone layers^{58;59}. Attention has also been utilized in vision models with great success, yielding strong performance while requiring less computing power and fewer input parameters, with self-attention and dual attention models being used in pursuits such as image classification and scene segmentation^{59;60}. The use of attention mechanisms in deep reinforcement learning, however, is less prevalent. Some variations of A2C incorporate a shared attention mechanism^{61;62}. Tang et al.⁶³ proposed a self-attention mechanism to rank important patches of the observation that influence the policy. Manchin et al.⁶⁴ presented a spatio-temporal self-attention mechanism for reinforcement learning in the case of RNN-based policy. However, our work differs by combining the attention mechanism with a partially split policy and value network designed to prevent overfitting and achieve generalization.

3.3 Preliminaries

While generic RL algorithms deal with a single (PO)MDP, formulating the problem of generalization considers a distribution of (PO)MDP that generates similar but distinct instances of (PO)MDPs with the overall goal of performing better on the entire distribution of (PO)MDPs. Here, we consider a distribution of POMDPs, denoted by $p(m)$ where $m \in M$, and each instance m is defined by a tuple $(\mathcal{S}_m, \mathcal{O}_m, \mathcal{A}, \mathcal{T}_m, \mathcal{R}_m, \Omega_m, \gamma)$, where $\mathcal{S}_m$ is the set

of states, $\mathcal{O}_m$ is the set of observations, $\mathcal{A}$ is the set of actions, $\mathcal{T}_m(s'|s, a)$ are the transition probabilities, $\Omega_m(o|s', a)$ are the conditional observation probabilities, $\mathcal{R}_m(s, a)$ is the reward function, and γ is the discount factor. If the agent is trained with a limited number of POMDPs, say n , then we have $\mathcal{M}_{train} = \{m_1, m_2, \dots, m_n\}$, where $m_i \sim p$ and $i \in \{1, 2, \dots, n\}$. The ultimate target is to optimize the policy π_θ over the full distribution of POMDPs where the objective function is defined by $J(\pi_\theta) = \mathbb{E}_{p, \pi, \mathcal{T}_m} \left[\sum_{t=0}^T \gamma^t \mathcal{R}_m(s_t, a_t) \right]$. In our experiment, each game environment from our testbed corresponds to a distribution of POMDPs $p(m)$ while each procedurally generated level within that game corresponds to a sampled POMDP instance according to that distribution. The model is tested on the full distribution while learning from a limited number of levels ($n = 200$ in our case), thus providing an opportunity to evaluate how the model can generalize beyond the levels it encounters during training. This collection of POMDP can be considered similar to the CMDP discussed in section 2.1.2 where every instance of the POMDP denotes a new context.

3.4 Attention-based Partially Decoupled Actor-Critic

In Attention-based Partially Decoupled Actor-Critic (APDAC), we modify the traditional shared representation of the actor-critic model by partially separating the policy and the value functions followed by a shared component. Each of the partially separated policy and value sub-networks is enhanced by including attention modules.

3.4.1 Partial Decoupling of Policy and Value function

Decoupling the policy and the value functions is crucial to overcome the problem of overfitting, which is the main drawback of a shared representation³. However, simply using two explicitly separated networks has serious inherent disadvantages because the policy function approximation depends on the gradient of the value function. Cobbe et al.¹⁶ shows that the straightforward method of just using two separate networks for policy and value functions causes a performance decrease when compared to the shared network architecture. To

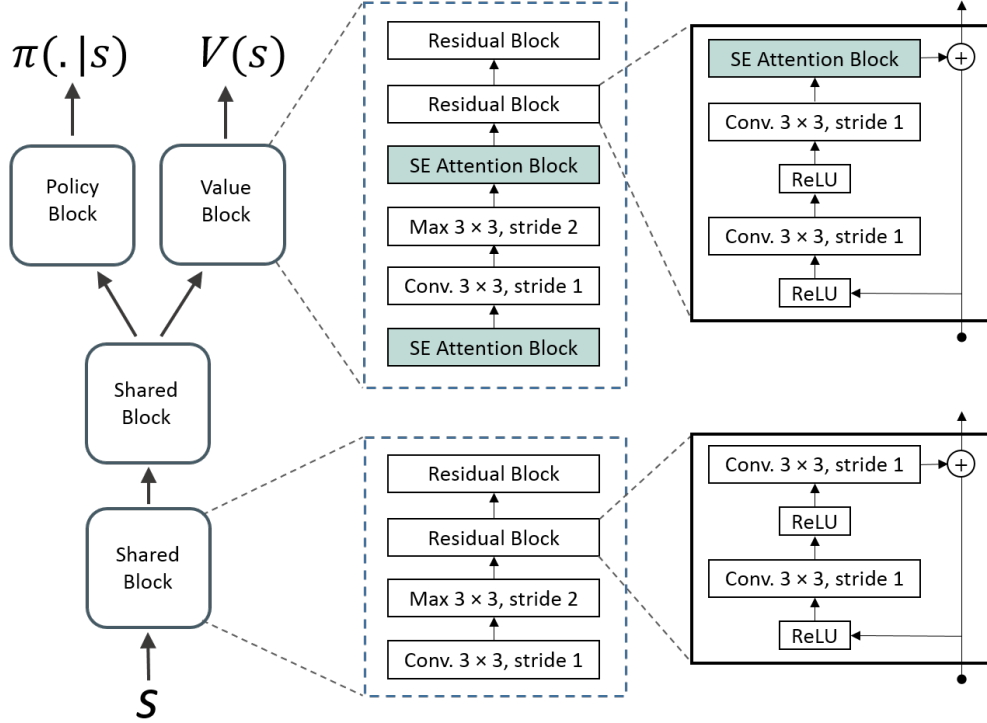


Figure 3.2: Details of our proposed architecture. The separated blocks for the policy and value function (at the top) incorporate an attention module (shown as shaded blocks) inside the residual blocks as well as one before the first convolutional layer and one after the max pooling layer. The shared blocks (at the bottom) are similar to the IMPALA CNN architecture¹. They do not have an attention module.

address this issue, research that uses a separate network approach to optimize policy and value functions often also uses an auxiliary value¹⁶ or advantage head³ in the policy network (See the policy network at the middle in Figure 3.1). This auxiliary head provides a helpful gradient to the separate policy network to learn better task-relevant policy representation, whereas the separate value network optimizes the value function that plays the original role of the critic. Moreover, the two network models introduce additional hyperparameters such as update frequency of the policy network, update frequency of the value network, and coefficients for the advantage loss.

We leverage the hierarchical representation of image features to design our network. Generally, the low-level features include minor details such as lines, edges, dots, and curves, whereas the high-level features are composed of multiple low-level features. Based on this,

we hypothesize that, although the high-level features responsible for accurate estimation of the policy and value functions may differ, the low-level features that constitute the high-level features are almost similar for both. Deep convolutional neural networks (CNN) can learn the feature representations hierarchically using a sequence of convolutional and pooling layers. Initial convolutional layers in a neural network learn the filters to capture low-level features while the later layers in the pipeline learn to identify larger objects and shapes. Therefore, we bifurcate the network earlier, at the convolutional layer level, instead of merely separating the policy and the value head as in the case of a fully shared network presented in². This helps to decouple the high-level feature learning for policy and value on top of the same features learned by the shared network. Thus, our network comprises three parts: the part of the network shared between policy and value parameterized by θ , a second part dedicated to policy learning parameterized by ϕ_π , and a third part dedicated to value function approximation, which is parameterized by ϕ_v . The overall network is trained all together to optimize the following objective:

$$J_{APDAC}(\theta, \phi_\pi, \phi_v) = J_\pi(\theta, \phi_\pi) - \alpha_v L_V(\theta, \phi_v) + \alpha_s S_\pi(\theta, \phi_\pi) \quad (3.1)$$

where $J_\pi(\theta, \phi_\pi)$ is the policy gradient objective, $L_V(\theta, \phi_v)$ is the value loss, $S_\pi(\theta, \phi_\pi)$ is an entropy bonus that enables efficient exploration, and α_v and α_s are the coefficients denoting relative weight of the corresponding terms.

We optimize the same clipped surrogate policy objective as used in PPO⁷:

$$J_\pi(\theta, \phi_\pi) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta, \phi_\pi) \hat{A}_t, \text{clip}(r_t(\theta, \phi_\pi), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right]$$

where $r_t(\theta, \phi_\pi) = \frac{\pi(\theta, \phi_\pi)(a_t|s_t)}{\pi(\theta, \phi_\pi)_{old}(a_t|s_t)}$, and $\hat{A}_t$ is the estimation of the advantage function at timestep t . The only difference is that the parameters ϕ_π are not affected by the value loss L_V while the parameters θ are affected. The value loss L_V is a squared error loss and is defined as follows:

$$L_V(\theta, \phi_v) = \hat{\mathbb{E}}_t \left[(V_{\theta, \phi_v}(s_t) - \hat{V}_t^{targ})^2 \right] \quad (3.2)$$

where $\hat{V}_t^{tar}$ is the value function target.

In the case of a fully decoupled value and policy network architecture, significant additional overheads are imposed by: (1) the reliance of policy optimization on the learned value gradient; (2) an increased number of hyperparameters; and (3) the consequent higher memory footprint. We show that these overheads can be overcome by a single network architecture that just *partially* separates the policy and the value. At the same time, our experimental results show that this partial separation prevents the model from being trapped in the common pitfalls of the fully shared network.

3.4.2 Relevant Feature Learning using Attention

Attention is an effective means to learn high-quality and meaningful representation. To ensure more discriminative and relevant feature learning by the partially separated policy and value sub-networks, we propose to incorporate individual attention mechanisms within the corresponding blocks of the network. We apply attention modules similar to the Squeeze and Excitation (SE) network, which explicitly models the interdependencies between the channels of its convolutional features⁶⁵. SE block leverages global information to put relative importance on more useful features than less useful ones. We use SE blocks only in the later layers (split value and policy sub-networks). Using a SE block in the later layers of a deep network enables distinct feature learning in a highly class-specific manner, while in the initial layers, it learns in a class-agnostic manner. We argue that this characteristic of the SE block makes it a suitable choice for our task, where distinct features relevant to policy and value must be learned.

The placement of the attention block within the sub-networks is a crucial design choice. We identify multiple key positions in the base IMPALA-CNN architecture¹ and conduct extensive experiments to determine the proper positions for the attention block. Based on the empirical analysis, we incorporate one SE attention block in each Residual Block just before the residual connection as shown in Figure 3.2. We also add extra SE attention blocks outside of the Residual Block: one at the beginning of the sub-networks to initially prioritize

the channels and another just after the max pooling layer for re-weighting the downsampled channels (See Section 3.5.1 and Figure 3.2 for details). We have found that the placement of the attention block either after the pooling layer or before the pooling layer makes a significant difference in terms of learning improved representation.

Following⁶⁵ to utilize global information beyond the local receptive field of filters, in our implementation, the squeeze operation in the attention block first encodes a channel descriptor $z \in \mathbb{R}^C$ through global average pooling. Each element of z is defined as:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W u_c(i, j), \quad (3.3)$$

where $u_c \in \mathbb{R}^{H \times W}$ and $U = [u_1, u_2, \dots, u_c]$ denotes the convolved feature output produced by the previous convolutional layer. In the next phase, the excitation operation attempts to capture channel-wise nonlinear dependencies based on the channel descriptor z . The excitation operation is realized by two fully connected (FC) layers around a non-linear function:

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z)), \quad (3.4)$$

where σ refers to the sigmoid activation function, δ refers to the ReLU function, $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$, and $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$; r denotes the reduction ratio parameter between the two FC layers. Finally, the input feature map $\mathbf{U}$ is rescaled as follows using the learned activation s :

$$\bar{x}_c = F_{scale}(u_c, s_c) = s_c u_c, \quad (3.5)$$

where $\bar{X} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_c]$. This way, a set of channel weights is learned to recalibrate the channel response. Thus, the attention block on the policy and value branches enables attended feature learning specific to the policy and value functions respectively. From our experiments, it is evident that the attention mechanism coupled with the contribution of Section 3.4.1 helps in attaining the high-level generalization.

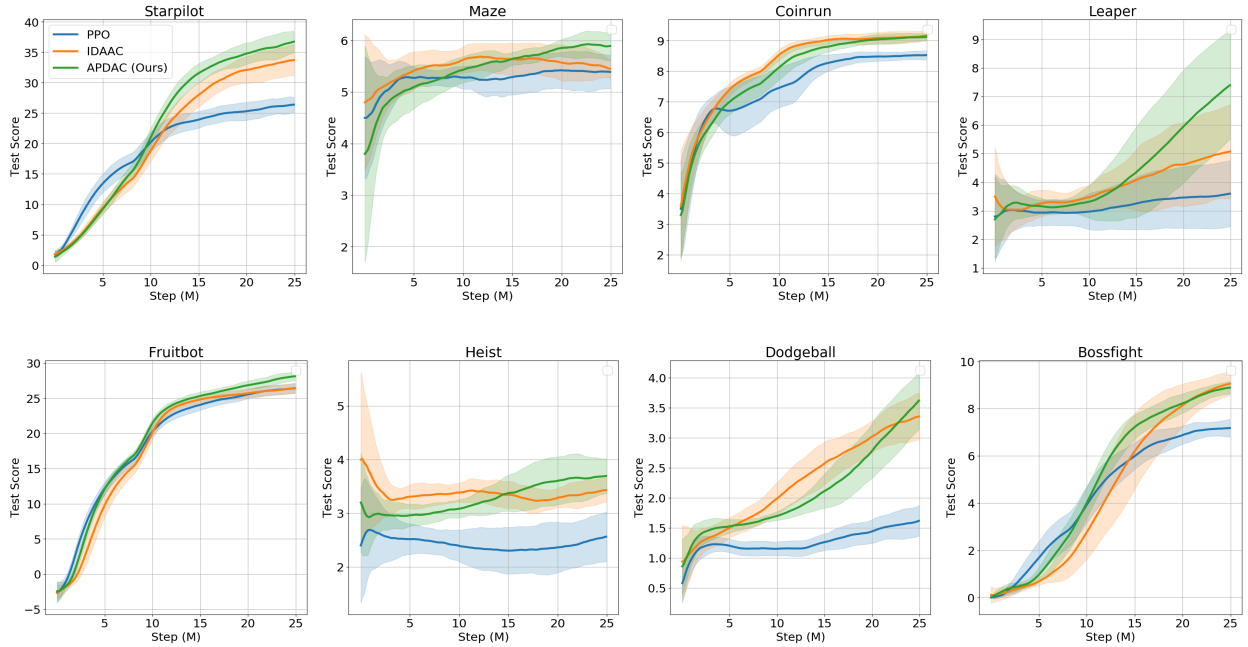


Figure 3.3: Test performance of PPO², IDAAC³, and our proposed APDAC over eight Procgen environments. Means and standard deviations are calculated over 10 trials, each with a different seed.

3.5 Experiments and Results

The availability of highly diverse procedurally generated levels across a wide variety of game environments has motivated us to choose Procgen as our testbed. We evaluate our proposed architecture on the complete Procgen benchmark presented in², which consists of 16 environments. We experiment on the easy difficulty setting for 25 million total timesteps as recommended in². We train the model on 200 levels and test on the full distribution of the levels. Procgen environments are designed to have a discrete 15-dimensional action space. The agent is required to learn the optimal policy directly from image observations. The observation space is of the size $64 \times 64 \times 3$. No frame stacking is used implying that the next state depends only on the present observation.

3.5.1 Network Architecture

Following previous works involving Procgen, we chose IMPALA’s deeper residual CNN architecture as our backbone^{2;3;16}. This model strikes a good balance between the achieved reward and the required computational power². This particular IMPALA CNN architecture has 15 convolutional layers divided into three blocks¹. Each block has a configuration similar to Conv - Pooling - Residual Block - Residual Block, as shown in the bottom-middle part of Figure 3.2. Each residual block includes two Conv layers with a ReLU activation layer. APDAC shares the first two blocks (10 convolutional layers) of the IMPALA CNN, then branches out for the third block. Thus, APDAC employs five separate convolutional layers each for policy and value functions in order to learn features distinctly. Finally, the separated policy and value blocks incorporate one attention unit per residual block along with one at the very beginning of the separated blocks and another just following the max pooling layer. Figure 3.2 shows the details of the architecture.

We extend the implementation of IDAAC³ to experiment with our APDAC architecture. We conducted a hyperparameter search over the number of epochs per rollout $E \in [1, 3, 6]$ and found $E = 1$ works best with APDAC. When training PPO and IDAAC, whenever applicable, we followed the same hyperparameter setup from³. The only difference is that we reduced the number of mini-batch sizes to minimize the required computational power. We use a reduction ratio $r = 8$ for the SE attention block. The source code is publicly available via a GitHub repository.¹

3.5.2 Generalization Performance on Test Distribution

In our experiments, we compared the performance of our network architecture, APDAC, with two representative state-of-the-art methods for all 16 Procgen environments. PPO serves as a representative of the models that use fully shared policy and value networks whereas IDAAC represents those with separate policy and value networks^{2;3;7}. In addition, IDAAC uses a discriminator loss to decrease the dependency on irrelevant instance-specific aspects

¹<https://github.com/NasikNafi/apdac>

Table 3.1: Scores on test levels after training on 25M environment steps for all 16 environments. Values are averaged over 10 trials, each with a different seed.

Game	PPO	IDAAC	PDAC(ours)	APDAC(ours)
Starpilot	26.4±1.2	34.5±2.9	36.2±1.7	36.9±1.7
Bossfight	6.5±0.4	8.9±0.5	8.9±0.4	8.9±0.3
Caveflyer	4.7±0.4	4.6±0.5	5.8±0.8	5.0±0.5
Chaser	4.7±1.0	6.3±0.9	6.6±1.4	5.9±1.1
Climber	5.8±0.4	8.1±0.5	7.9±0.3	7.7 ±0.3
Coinrun	8.54±0.2	9.1±0.1	9.1±0.2	9.1±0.2
Dodgeball	1.7±0.3	3.4±0.4	3.4±0.5	3.7±0.5
Bigfish	3.7±0.6	18.1±1.6	11.0±1.8	14.1±2.3
Fruitbot	23.9±0.8	26.4±0.5	25.7±0.7	28.3±0.6
Heist	2.6±0.5	3.4±0.2	3.5±0.6	3.6±0.3
Plunder	5.4±0.3	17.5±2.4	5.8±0.6	6.3±0.7
Jumper	5.8±0.3	6.1±±.2	6.0±0.2	6.0±0.2
Leaper	3.6±1.2	5.2 ± 1.7	7.4 ± 1.4	7.6 ± 1.4
Miner	9.1±0.5	6.9±2.2	8.7±0.5	8.9±0.6
Maze	5.4±0.3	5.4±0.2	5.7±0.3	5.7±0.4
Ninja	6.2±0.2	7.1±0.3	6.4±0.2	6.3±0.6

Table 3.2: Comparison of number of required convolutional layers and computational time for different approaches

Methods	No of Conv. Layer	GPU Hours
PPO ²	15	3-4
IDAAC ³	30	5-12
APDAC (Ours)	20	3-5

of the environment. Figure 3.3 shows the rolling mean test scores averaged over ten trials for each of the eight environments out of sixteen from the Procgen benchmark. The rolling standard deviations between trials are calculated as well, with confidence intervals bounding one standard deviation above and below each curve. Table 3.1 presents the scores on test levels after training for 25M environment steps for all sixteen environments. In these results, APDAC attains significant gains in performance compared to the standard shared network approach, PPO^{2,7}. Furthermore, APDAC performs better or competitively compared to the existing state-of-the-art, IDAAC, and does so with fewer convolutional layers and less

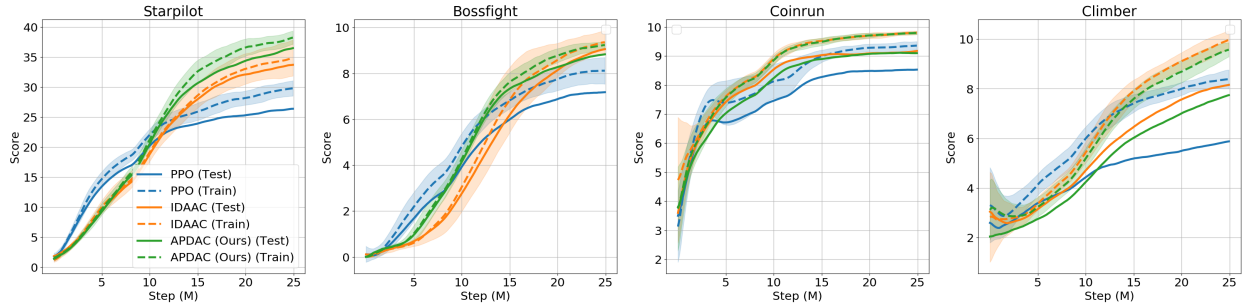


Figure 3.4: Training and test performance of PPO², IDAAC³, and proposed APDAC over four Progen environments. Means and standard deviations are calculated over 10 trials, each with a different seed. The dotted lines refer to the training reward while the solid line refers to the test reward. In the case of APDAC, the gaps between the dotted line and solid line (green) are similar to the fully decoupled approach, IDAAC (orange) and both are better than PPO (blue).

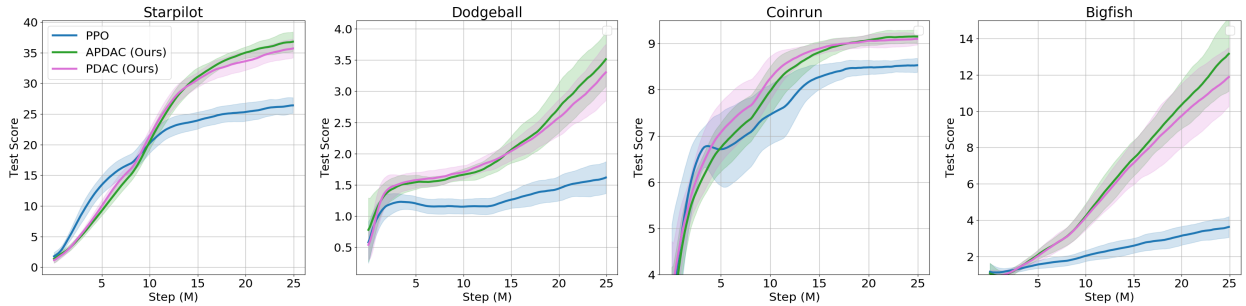


Figure 3.5: Ablation study of the proposed method APDAC with the standard baseline PPO (no decoupling, no attention) and the ablated version PDAC (partially decoupled but without attention) on the test distribution for four Progen environments. Means and standard deviations are calculated over 10 trials, each with a different seed.

computing time. In our particular instantiation, while two separate networks in IDAAC require 30 (15 for policy; 15 for value) convolutional layers, our approach requires only 20 (10 shared; 5 for policy; 5 for value). The number of parameters added for attention blocks is not significant. Table 3.2 shows this comparison. Also, APDAC requires a run time similar to the shared network approach (PPO) which is far less than the fully decoupled approach (IDAAC) in most cases. The reported hours are based on experiments on an Nvidia GeForce GTX 2080 GPU. Thus, we conclude that APDAC succeeds as an efficient hybrid architecture that improves upon its constituent network topologies.

3.5.3 Assessing the Generalization Gap

In addition to analyzing the performance on the test distribution, it is crucial to investigate the performance gap between training and test tasks or levels. This provides an indication of how well the model performs on unseen tasks compared to ones that belong to the training set. It is obvious from Figure 3.4 that APDAC is capable of reducing the train-test gap (solid vs. broken line) at the same level of a fully decoupled approach, IDAAC and far better than the fully shared approach, PPO.

3.5.4 Ablation

To evaluate the contributions of each proposed component, we further experiment with an ablated version of our proposed approach, which eliminates all attention blocks. Thus, this network includes only the partially separated policy and value representations and does not incorporate the contribution mentioned in Section 3.4.2. We denote this model as the Partially Decoupled Actor-Critic (PDAC). To determine the difference in performance brought by this ablation, we compare the results of PPO, APDAC, and the ablation, PDAC, using the same experimental setup as before. Figure 3.5 shows the ablation results from four example environments. APDAC performs better than PDAC in most cases. Indeed, the comparison with PPO clearly shows that the main performance gain of APDAC comes from the partial separation of policy and value networks. Thus, sharing the initial part of the network does not harm performance and, in fact, reduces the required number of convolutional layers compared to an architecture with fully separate value and policy networks. The marginal performance gains between APDAC and the ablation are an incremental aspect of this work; however, attention is still a growing field of study in RL, and our research shows an opportunity to use attention to achieve generalization (Also, see Figure 3.6).

3.5.5 Visualization of Value and Policy using Grad-CAM

To understand the agent’s ability to reason about the environment, we visualize where the agent is attending while making its decisions. This highlights the important spatial regions

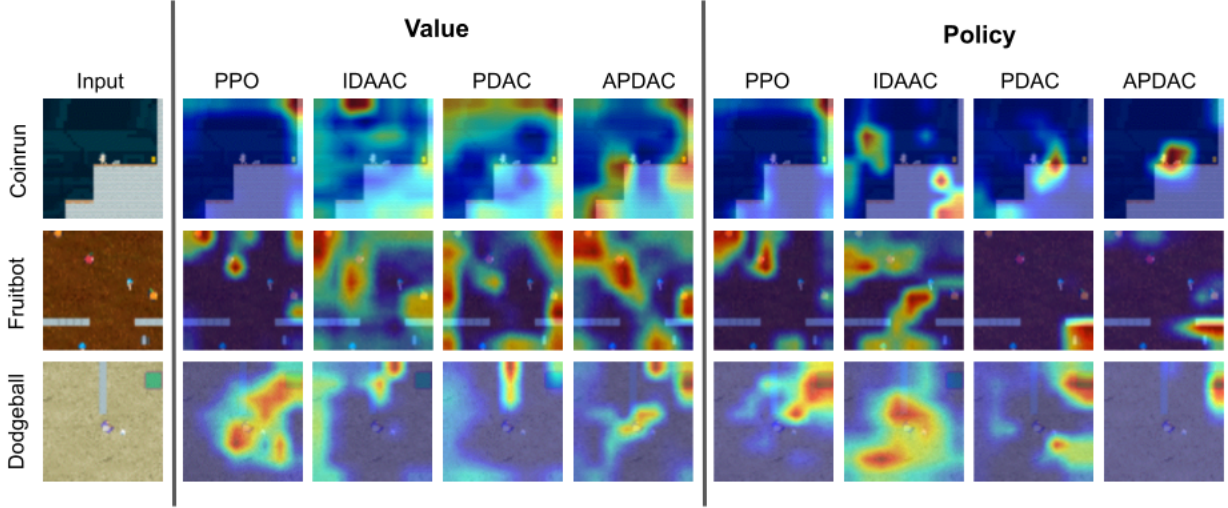


Figure 3.6: Visualization of the value and policy representation for PPO², IDAAC³, the ablation, PDAC, and the proposed approach, APDAC, on an input sample from three Procgen environments. The red regions correspond to the higher value of gradients while the blue regions correspond to the lower value. For both policy and value, compared to other approaches, APDAC exclusively prioritizes informative spatial regions (red marked) and constructs asymmetrical representation.

where exactly the agent is focusing on and helps us to understand in-depth about the key information that is influencing the agent’s behavior. We generate activation maps for both value and policy using the Grad-CAM technique, which uses gradients back-propagated from the final layer⁶⁶. Specifically, we set the last layer before the fully connected part as the target layer and set the final value and policy head outputs as the target concept. We visualize the policy in an action-indiscriminative way as we are more interested in learning the difference between policy and value activations rather than action-specific policy.

Figure 3.6 presents visualizations of the value and policy for PPO, IDAAC, proposed APDAC, and the ablation PDAC. The red regions correspond to the higher value of gradients while the blue regions correspond to the lower value. APDAC produces more fine-grained and explainable visualization compared to other approaches. Figure 3.6 shows that APDAC policy gradients are highly concentrated in specific spatial regions (see the red regions). A policy can be optimal by encoding the minimal task-relevant information from the observation³. The visualization reveals that APDAC policy attempts to focus only on the key information required to act immediately on the environment. Compared to the ablation

PDAC, APDAC produces much clearer gradients and consequently concise representation. We observe that APDAC value and policy gradients are highly asymmetrical. This confirms that an asymmetry truly exists between the policy and value function representations that need to be addressed to achieve better generalization. PPO produces identical activation maps for policy and value, thus failing to encode distinct features for policy and value owing to its shared representation. While IDAAC representations are asymmetrical, it is evident that these are not plausible enough to reason about the action in the environment.

In Figure 3.6, the first row is from the game of Coinrun where the goal of the agent is to collect the coin at the far right of the level while avoiding the stationary saw obstacles, moving enemies, and chasms. In the sample input, the yellow coin is at the right, and the blueish saw obstacle is just in front of the agent. We can see the APDAC value function places more attention on the coin and the specific edges of the terrain. On the other hand, the policy puts attention on the obstacle coming ahead, a saw, as this needs to be bypassed.

In the game Fruitbot (the second row in Figure 3.6), a robot needs to navigate through the gaps in the walls to collect fruit along the path. It receives a positive reward for collecting fruit, however, receives a negative reward for a non-fruit object. The APDAC value column highlights all the fruits in red while the non-fruit objects are in the blue region. In contrast, the policy highlights the wall just in front of the robot as this wall must be avoided.

For Dodgeball (the third row in Figure 3.6), the player spawns in a closed room with randomly configured enemies. The player must kill all the enemies by throwing the ball while avoiding hitting the walls. When all the enemies are killed, the player can go to the newly unlocked platform (previously locked) and receive a large level completion bonus. In the given example in the third row, all the enemies have been killed, and the platform turns from red to green showing it is now unlocked (appears red when locked). APDAC value function focuses on the platform and the path to the platform. The policy also highlights the platform because the player must move closer to it.

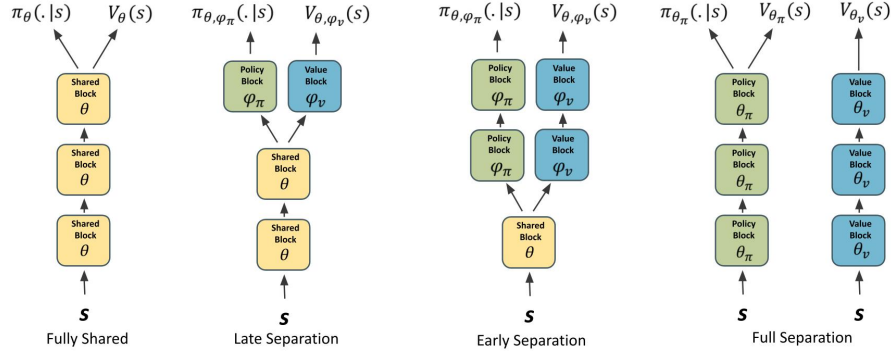


Figure 3.7: Different network architectures highlighting the extent of decoupling between policy and value networks.

3.6 Analyzing the Generalization Sensitivity to the Extent of Policy-Value Decoupling

The question remains open as to what extent of decoupling should be used if a new environment is encountered. Unfortunately, there is no guideline for deciding how much decoupling is enough for policy and value networks in order to achieve a reasonable performance while keeping the computational burden low. In most cases, substantial compromise in generalization performance is not acceptable to reduce the computational overhead. This leads to the question *”Does separating the network early (having most of the layers separated) in partial separation improve generalization while avoiding the drawbacks of full separation?”*. In this section, we investigate the effect of different degrees of decoupling of the actor and critic network on the agent’s generalization performance and systematically answer such research questions. Here are the specific contributions covered in this section:

- We instantiate four architectures with different degrees of separation: fully shared (no separation), early separation, late separation, and full separation (see 3.7); and evaluate the performance aggregating across all 16 environments from the Procgen benchmark and the Crafter.
- We independently analyze the results for 16 individual environments of Procgen. Further, we delve into the learned representation for policy and value networks in all these

settings to identify whether and how the separation helps and draw network design insights from this process.

- We demonstrate that while separation helps, separating too early in a partially separated architecture negatively impacts the generalization performance, in addition to incurring higher computation costs.
- We find that full separation is a must when there is a clear distinction between the source of value and policy features. Otherwise, partial *late separation* suffices and can be the best choice.

3.6.1 Instantiating Different Extent of Decoupling

To analyze the sensitivity to the decoupling of policy and value, we create four different architectures (1) there is no separation of the policy and value network other than the final policy and value heads (fully connected layers) at the end, (2) the network is separated early in the layers (3) the network is separated in the more downstream layers (4) two separate networks. Figure 3.7 depicts these four architectures. Our specified degrees of separation are exhaustive: by category, they are defined as a partition of value/policy networks that exhibit decoupling; by exact separation point, each mutually exclusive and exhaustive category is represented, rather than combinatorially enumerating all k separation points for a deep model with k layers (e.g., $k = 15$). We use the large IMPALA-CNN architecture¹ as the base network, following the previous state-of-the-art works^{3;16}, as well as the original results of Progen benchmark². This deeper IMPALA CNN architecture has three blocks, each having a configuration of Convolution Layer - Pooling Layer - Residual Block - Residual Block, where each residual block has two convolution layers, with a total of 15 convolutional layers. In the rest of this section, we describe the details of these four architectures, their loss functions, the optimization process, and their implementation through the IMPALA-CNN base network.

Fully Shared

By *fully shared*, we denote the traditional architecture that shares the network for the actor (policy) and the critic (value). Specifically, we share the IMPALA-CNN base network (including all convolution layers) between policy and value, however, we only separate the final fully connected layers that represent the policy and value heads (see the left one from Figure 3.7). This is similar to traditional Proximal Policy Optimization (PPO)⁷ implementations e.g.^{2,3}. Given the network parameter θ , we optimize the standard objective function used in policy gradient approaches:

$$J_{NS}(\theta) = J_{\pi}(\theta) - \alpha_v L_V(\theta) + \alpha_s S_{\pi}(\theta) \quad (3.6)$$

where $J_{\pi}(\theta)$ is the policy gradient objective, $L_V(\theta)$ is the value loss, $S_{\pi}(\theta)$ is the entropy bonus for exploration, and α_v and α_s are the corresponding coefficients. PPO builds upon the Trust Region Policy Optimization (TRPO)⁴⁰ method, which maximizes a surrogate objective function defined as: $J_{\pi}(\theta) = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right]$ where $r_t(\theta) = \frac{\pi_{(\theta)}(a_t|s_t)}{\pi_{(\theta)_{old}}(a_t|s_t)}$ is the probability ratio between the new policy and the old policy, and $\hat{A}_t$ refers to the advantage estimate at timestep t . To avoid excessively large policy updates, PPO clips the value of $r_t(\theta)$ between the intervals of $[1 - \epsilon, 1 + \epsilon]$ and takes the minimum between the original value of $r_t(\theta)$ and the clipped one. Thus, the final clipped surrogate objective function is as follows:

$$J_{\pi}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right] \quad (3.7)$$

For the fully shared architecture, we optimize the same clipped surrogate objective as part of Equation 3.6.

Early Separation

To implement early separation, we share the first block of the IMPALA-CNN architecture and separate the next two for policy and value. Thus, only 5 convolutional layers are shared. Consider that the part of the network shared between policy and value is parameterized by θ ,

the separated policy subnetwork is parameterized by ϕ_π , and the separated value subnetwork is parameterized by ϕ_v . Then, we optimize the objective jointly for all parameters, similar to the PPO loss (Equation 3.6):

$$J_{ES}(\theta, \phi_\pi, \phi_v) = J_\pi(\theta, \phi_\pi) - \alpha_v L_V(\theta, \phi_v) + \alpha_s S_\pi(\theta, \phi_\pi) \quad (3.8)$$

where $J_\pi(\theta, \phi_\pi)$ is the policy gradient objective, $L_V(\theta, \phi_v)$ is the value loss and $S_\pi(\theta, \phi_\pi)$ is the entropy bonus.

Late Separation

In this variant, we share the first two blocks of the IMPALA-CNN architecture and separate the third block for each of the policy and value network components. As a result, the first 10 convolutional layers are shared while there are two different sets of 5 convolutional layers for each of the policy and value subnetworks. If the shared network part is parameterized by θ , the separated policy network and value network are parameterized by ϕ_π and ϕ_v respectively, then we optimize the objective jointly, similar to the PPO loss (Equation 3.6):

$$J_{LS}(\theta, \phi_\pi, \phi_v) = J_\pi(\theta, \phi_\pi) - \alpha_v L_V(\theta, \phi_v) + \alpha_s S_\pi(\theta, \phi_\pi) \quad (3.9)$$

where $J_\pi(\theta, \phi_\pi)$ is the policy gradient objective, $L_V(\theta, \phi_v)$ is the value loss and $S_\pi(\theta, \phi_\pi)$ is the entropy bonus. This is similar to the partial separation proposed by Nafi et al.²⁶. The difference between early separation and late separation is that the number of shared parameters represented by θ is higher in late separation.

Full Separation

As the name suggests, we use two fully separate networks for policy and value. This approach is similar to the one proposed by Raileanu and Fergus³ that fully disentangles the policy and value representations. We also keep the extra advantage head in the policy network, as without any sort of value or advantage gradient the policy network remains isolated, resulting

in the failure of the policy optimization process¹⁶. Since there are two different networks, the optimization takes place in two phases with the policy network optimized first, followed by the value network. The policy network parameterized by θ_π is optimized for:

$$J_{FS}(\theta_\pi) = J_\pi(\theta_\pi) - \alpha_A L_{A_\pi}(\theta_\pi) + \alpha_s S_\pi(\theta_\pi) \quad (3.10)$$

Here $L_{A_\pi}(\theta_\pi)$ is the advantage loss coming from the additional advantage head used to support the policy network³. On the other hand, the value network, parameterized by θ_v , optimizes the value loss defined as follows:

$$L_v(\theta_v) = \hat{\mathbb{E}}_t[(V_{\theta_v}(S_t) - \hat{V}_t^{targ})^2]$$

where $\hat{V}_t^{targ}$ is the target value function.

3.6.2 Experimental Setup

We conduct our experiments on all the 16 game environments available in the demanding Procgen benchmark² and the single environment from Crafter benchmark²⁴. These highly diverse environments provide the opportunity to analyze and draw conclusions regarding when to use a certain network architecture. We use the standard protocol for training and testing as introduced in² and²⁴ for Procgen and Crafter respectively. Our code is publicly available.² For Procgen, we employ the easy distribution mode and train all the agents on only 200 levels of the games for 25M timesteps while testing on the full distribution of levels. The term full distribution refers to the configuration that each episode in the testing phase can belong to any level selected from an infinite set of procedurally generated levels. Thus, a learned policy needs to perform well in the scenarios not encountered during training. Crafter offers environment variation and multiple achievement targets that require a generalizable policy to perform better. For Crafter, we train the agent for 1M timesteps.

²<https://github.com/nasiknafi/sensitivity-to-decoupling>

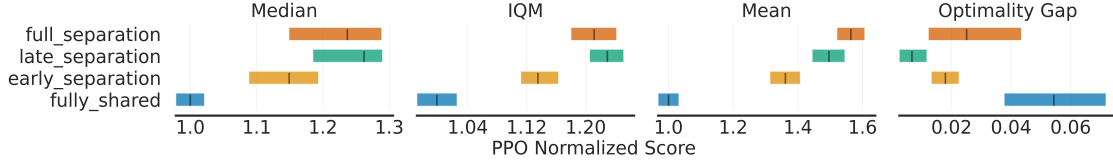


Figure 3.8: PPO normalized scores for the four architecture variants across all the 16 environments in the Procgen benchmark.

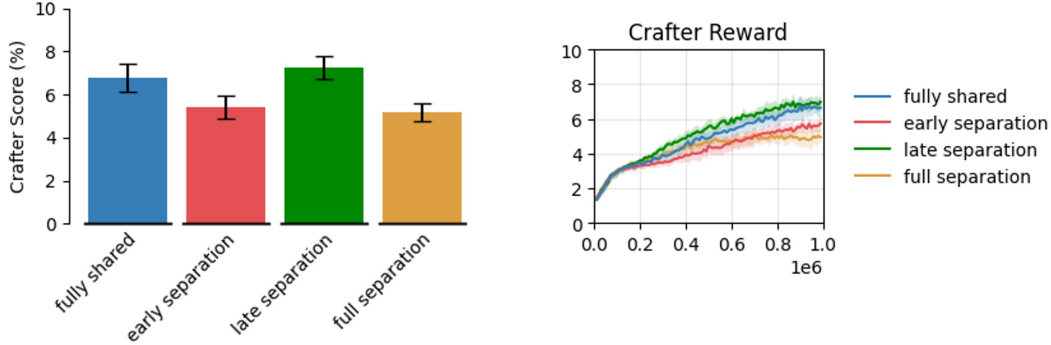


Figure 3.9: (Left) Crafter score for the four architectures with different degrees of separation for policy and value; (right) Comparison of the reward achieved by each agent during 1M timestep. In terms of both metrics, *late separation* performs better.

3.6.3 Generalization Across Benchmarks

To analyze the overall performance, we first report the results combined across all 16 Procgen environments. As a performance measure, we use the average return achieved by the agent in the test levels or episodes which is often referred to as the *test score* or *test reward*. In addition to the traditional mean and median, to address statistical uncertainty across all trials, we consider more critical metrics such as Interquartile Mean (IQM) and optimality gap (OG) as introduced by⁶⁷.

Figure 3.8 shows the evaluation metrics in terms of the PPO-normalized scores across the benchmark. As described previously in Section 3.6.1, *fully shared* approach refers to the PPO algorithm. Thus, Figure 3.8 can be considered as a performance representation of other network architectures relative to the fully shared version wherein the score of the *fully shared* version is 1 in the scale. It is evident from Figure 3.8 that *late separation* outperforms all

other approaches in terms of the IQM value across all environments while sharing a significant portion of the network. While the median value of *late separation* is also higher than all other approaches, the mean value is slightly less compared to the *full separation* but still better than *early separation*. Further, the optimality gap of *late separation* is the lowest among all other approaches. The study by Agarwal et al.⁶⁷ demonstrates that the Interquartile Mean (IQM) proves to be a more comprehensive indicator of overall performance compared to the median. This is attributed to IQM’s consideration of 50% of the combined runs, providing a nuanced perspective beyond a simple performance order. Additionally, IQM exhibits greater resilience to outliers when compared to the mean. The Optimality Gap (OG) signifies the extent to which the algorithm falls short of achieving a minimum score, beyond which further enhancements hold diminishing importance. Thus it is important to consider IQM and OG as the performance measure while analyzing the outcome.

From the results, we observe that *early separation* while having a large portion of the network separated for the policy and value still performs worse than the *late separation*. A naive assumption could have led one to believe that *early separation* might work best as this should capture the policy-value representation asymmetry better through its larger separated part. However, experimental studies show a deviation from the general expectation. We argue that such deviation is due to the combined adversarial effect of the higher isolation of the policy network and the lack of separate policy and value optimization. The rationale behind this argument refers to the addition of an advantage head (or value head) to the policy network and separate optimization of the policy and value networks to alleviate performance degradation that occurs with naive separation^{3;16}. We present more details related to this issue in section 3.6.5 through additional experiments.

On the other hand, the better performance of *late separation* in terms of IQM and OG can be attributed to the fact that models with late separation can still capture the necessary policy-value representation asymmetry through partial decoupling. Due to the hierarchical representation of image features, sharing the early layers that learn low-level features (e.g., edges and dots) or mid-level features (e.g., object parts) does not restrict achieving policy-value representation asymmetry (through high-level features). At the same time, *late*

Table 3.3: Comparison of computational time and the number of convolutional layers

Methods	No. of Conv. Layer	GPU Hours	GPU Memory
fully shared	15	3-4	2.5-3 GB
late separation	20	3-5	2.5-3 GB
early separation	25	3-5	2.5-3 GB
full separation	30	5-10	4-5 GB

separation does not suffer from the value gradient as the degree of separation is not high. *Late separation* has most of the layers shared, hence the intermediate representation receives enough signal from the value function that eliminates the instability due to the dependency of the policy function on the value gradient. Table 3.3 shows a comparison of the computational overhead for all four variants used in our experiments. As partially separated models require significantly less time than the fully separated approach and *late separation* can achieve competitive or better performance, we recommend using *late separation* in general, especially when computational complexity is crucial.

Figure 3.9 shows the Crafter results and it is evident that *late separation* performs better compared to the others with respect to the Crafter score and achieved rewards. Crafter score is defined as the geometric mean of success rates. Thus it carries greater weight for unlocking new achievements²⁴.

3.6.4 Performance on Individual Procgen Environments

In addition to the overall performance, we look at the empirical results for all individual environments to evaluate how different degrees of separation affect them. Figure 3.10 presents results across the entire Procgen benchmark over the training time of 25M timesteps. The solid line refers to the running mean of rewards while the shaded regions refer to the variance across trials. We observe that in all the environments except *Plunder*, *late separation* achieves competitive scores, even better in some cases, compared to the *full separation*. *Early separation* performs competitively with *late separation* in a few environments, however, this

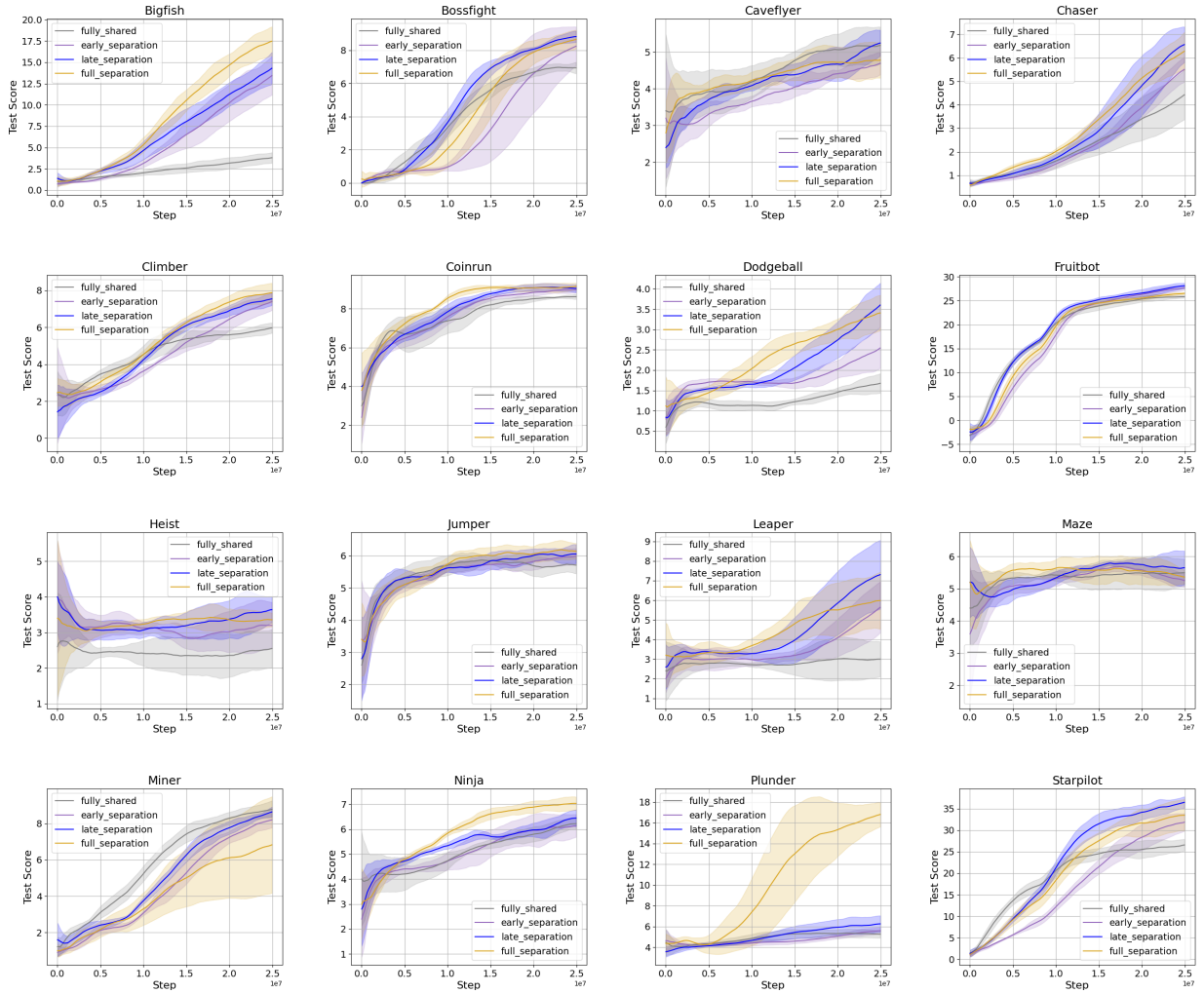


Figure 3.10: Test performance for fully shared, early separated, lately separated, and fully separated architecture across all 16 environments from the Procgen benchmark. Mean and standard deviation are calculated over 5 trials with different seeds.

performance is not consistent. The *fully shared* architecture one only performs competitively in the *caveflyer* and *miner* environments. A detailed analysis of each environment using the IQM is provided in Figure 3.11.

We now examine the issue of complete failure of all architectures except *full separation* in the *Plunder* environment to assess the necessity of *full separation*. Our investigation reveals that, unlike other environments, *Plunder* includes an on-screen countdown timer (depicted by the green bars on the top in each figure in Figure 3.12). When the timer runs out, the episode ends at that instant. Thus, this on-screen timer acts as a life bar for the agents.

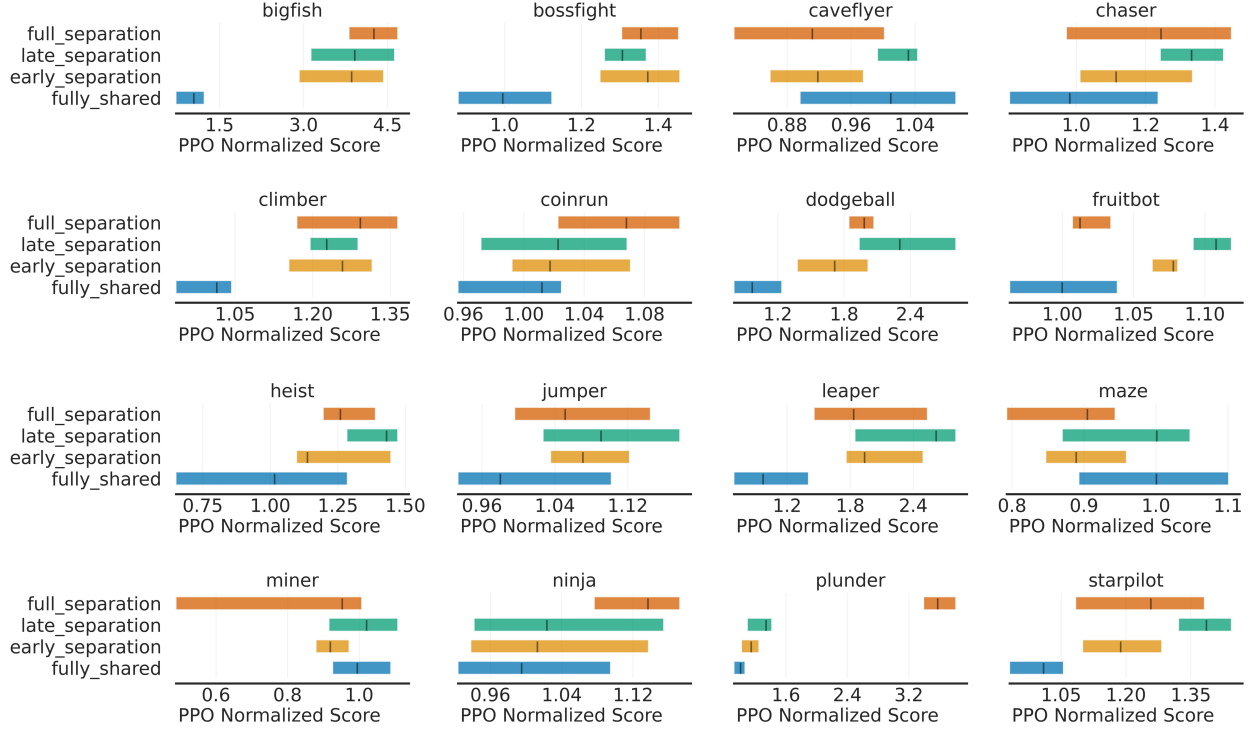


Figure 3.11: Interquartile Mean (IQM) with 95% confidence interval for PPO (fully shared) normalized test rewards for all the 16 environments from the Procgen benchmark.

The policy needs to learn to avoid hitting friendly ships and destroying enemy pirate ships by firing cannonballs. A target in the bottom left corner of the screen shows the color of the targeted enemy ships. However, the life bar is an important source of value estimation of the state. Figure 3.12 presents examples of the learned policy and value representation highlighted through Grad-CAM⁶⁶ for both *late separation* and *full separation*. It is evident that the value representation in the fully separated one pays attention to the top-left corner where the life bar ends while the value network in case of late (partial) separation fails to do so. On the other hand, the policy representations for a fully separate network keep track of the end of the life bar while also paying attention to the enemy ships. However, the policy representations in case of late (partial) separation focus on the complete life bar. We hypothesize that this happens due to the overarching effect of the value function. In partial separation, while sharing some parts, the policy network puts significant importance on the value source and fails to distinguish between the policy and value representation.

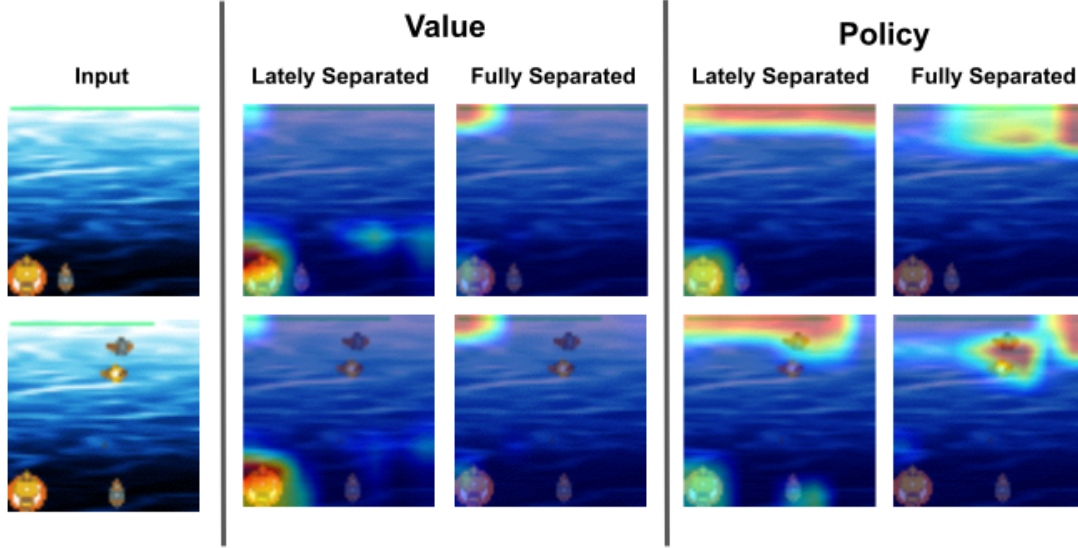


Figure 3.12: Learned representation of the policy and value using lately separated and fully separated architectures for the *Plunder* environment (at two different timesteps). The red-marked regions correspond to the higher value of gradients while blue regions correspond to the lower value of gradients. The learned policy representations using the partial late separation approach get biased with the value function and primarily look at the whole life bar as opposed to the end mark focused by the policy representation learned by the fully separated approach.

Thus, we conclude that when explicit sources of value estimation are present in the input observation, then consideration should be given to learning representation for policy and value through two fully separate networks, irrespective of the computational time overhead to gain generalization improvements.

3.6.5 Extended Study for Early Separation

To investigate why early separation does not perform well while separating the network at a considerable scale, we conduct additional experiments where we integrate a few mechanisms that have been proven helpful in the case of full separation. Figure 3.13 presents the experimental results. *Early separation joint* denotes the same implementation as of early separation. The term "joint" refers to the fact that the optimization of the policy and value network has been done together. As the model with *full separation* consists of two different networks, two optimizers are used to update the two different networks based on

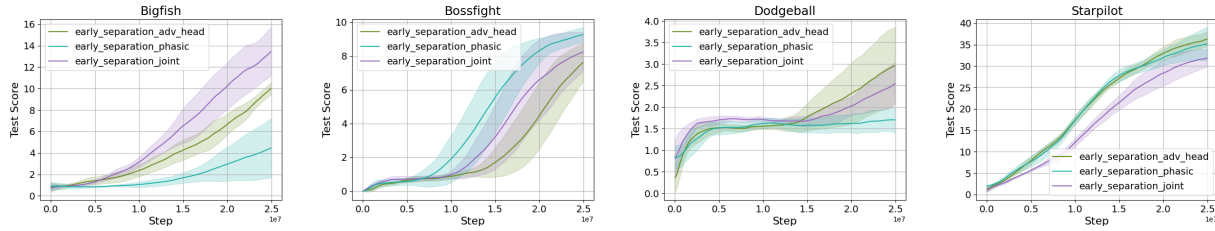


Figure 3.13: Test performance of the *early separation* architecture and its two variants on four environments from the Procgen benchmark. *Early separation joint*, the baseline, uses the loss function as defined in eq. 3.8, *early separation phasic* uses separate optimization phases for policy and value loss, *early separation adv head* incorporates an extra advantage head in the policy subnetwork. Results show that neither of these two extensions alone can improve the performance of the *early separation* architecture.

their corresponding loss functions. This update generally happens in two phases: the policy phase and the value phase. Each phase performs multiple gradient updates defined by the hyperparameters: number of policy epochs E_π and number of value epochs E_v . The training phase of the value network occurs after every N_π policy update. We incorporate the same training style with the early separation model. We use two optimizers - one dedicated to policy parameters and another to value parameters. Thus, shared parameters are updated twice. We refer to this variant as *early separation phasic*.

As the network size of the separated part increases in the *early separation*, the downstream policy network may suffer more from value gradient and as suggested by ¹⁶, the policy network without enough value gradient may collapse. This may be partially overcome by introducing an extra value or advantage head. Thus we create another variant of early separation that includes an extra advantage head in the policy subnetwork. This additional advantage head can be trained using the following advantage loss:

$$L_A(\theta, \phi_\pi) = \mathbb{E}_t[(A_{\theta, \phi_\pi}(s_t, a_t) - \hat{A}_t)^2]$$

where $\hat{A}_t$ is the generalized advantage estimate (GAE)⁶⁸ at the time step t calculated from the value predicted by value subnetwork. This extra advantage head provides supplementary value gradients to the policy subnetwork. We denote this variant as *early separation adv head*.

Results in Figure 3.13 show that while none of the modified variants performs better

across all cases, in most cases, disjoint phasic optimization aids the network. For some environments like *dodgeball* and *starpilot*, the advantage head helps in performance. However, disjoint optimization introduces a high computational burden as two different optimizers need to operate. Also, it requires additional hyperparameter (E_π, E_v, N_π) tuning. For many environments, the value update frequency E_v is high, thus requiring more updates. We do not combine both the extra advantage head and disjoint phasic optimization, because by doing so the model will be very similar to the *full separation* one. Therefore, this will again introduce all the drawbacks of the *full separation*, reducing the benefits of *partial separation*.

3.7 Conclusion

In this work, we have simultaneously addressed the issue of limited generalization due to overfitting in a fully shared actor-critic network and the added complexity in the case of fully decoupled ones. Our solution, APDAC, differs from existing methods in the partial separation of its network and adding attention mechanisms to each split sub-network. Our empirical results demonstrate highly competitive performance compared to a fully decoupled state-of-the-art approach while reducing the number of required convolutional layers and the computational cost. APDAC learns compact policy representations that are robust to the variation of the unseen task episodes. Although attention greatly improves representation learning, the limited performance gap between APDAC and the ablation can be considered a limitation of the current work. Further, this work empirically studies the four possible general categories of policy-value networks that use different levels of decoupling/separation for the policy and value function, seeking to infer meaningful insights regarding the impact of the extent of decoupling on generalization performance through systematic comparison. The findings suggest when to leverage two fully separate networks even though it might entail increased computational complexity. It identifies the counterintuitive phenomenon that early separation fails even though it separates a large portion of the network. Thus, partial late separation (APDAC) appears as a promising way forward in the pursuit of generalization in deep RL on the grounds of performance, efficiency, and interpretability.

Chapter 4

Minimalist Approach to Augmentation Invariant Representation Learning in RL

4.1 Introduction

Despite the success of modern deep Reinforcement Learning (RL), learning a generalizable policy directly from visual observations remains a foundational concern^{21 2}. RL algorithms may completely fail to generalize when slight environment changes occur as opposed to the biological agents who can quickly adapt or generalize previous learnings to unseen environmental conditions³. While the use of diverse samples can effectively improve the generalization performance of RL agents, data collection for training real-world control tasks remains costly and notoriously difficult. The online nature of many state-of-the-art RL algorithms exacerbates the issues further. To address this problem, recent approaches have proposed to leverage techniques like data augmentation⁶⁹, regularization⁶, or representation learning⁴. In this work, we provide a solution from a combined perspective of data augmentation and representation learning that is simple to implement and works well without requiring the computational overhead of the previous methods.

We observe that current RL algorithms leverage data augmentation in different ways: (i) simply apply augmentation to data and use those augmented data directly for RL objective estimation⁶⁹, (ii) add additional auxiliary losses based on augmented data to regularize the policy and value function^{6 70}, and (iii) use unsupervised contrastive and non-contrastive learning to learn representation from similar and dissimilar pairs^{4 5}. For contrastive losses similar (positive) data denotes the augmented ones and dissimilar (negative) ones refer to the samples from other trajectories⁴. While contrastive loss-based approaches attempt to learn better representation, without extensive hyperparameter tuning, naive application of data augmentation often outperforms them in challenging tasks where the agent needs to generalize to unseen contexts⁶⁹. Existing non-contrastive approaches use only the positive pairs, however, leverage carefully designed architecture with projection and predictor heads⁵, making them difficult to incorporate with any RL algorithms. RL algorithms that use augmentation-based regularization aim to directly learn transformation invariant policy and value function, thus sacrificing the explicit representation learning step⁶. In general, learning from state-based features is known to be more sample-efficient than learning from raw pixels⁴. Thus learning latent representations for high-dimensional observations that preserves the semantic meaning can greatly help the control algorithm to achieve sample-efficiency and generalization. Non-contrastive similarity-based approaches have recently achieved more attention in computer vision^{71 72 73}, however, the full potential of such an approach is yet to be explored in-depth for reinforcement learning.

In this paper, we propose Reinforcement learning with Augmentation Invariant Representation (RAIR) that distinctly considers the representation learning task and the downstream RL task. More specifically, we employ non-contrastive self-supervised embedding learning and RL objective optimization in a phasic way within the same batch. During the unsupervised representation learning phase, RAIR learns representations that are invariant to the augmented version of the observed state using a simple encoder and then goes to the next phase where it optimizes the standard RL objective. By augmentation invariant, we refer to the property that for both the original observation and corresponding transformed observation the encoder will produce a similar latent representation so that the agent can perceive

that both are the same. While we validate our approach using Proximal Policy Optimization (PPO)⁷ that uses an actor-critic architecture, our method can be used with any RL algorithm (on-policy and off-policy) that uses neural network-based function approximator to represent the policy and value function, and the observation space allows some kind of meaningful transformations of the states. We evaluate our proposed approach on the Procgen benchmark². We empirically show that our proposed approach outperforms standard PPO without any kind of augmentation and is comparable to or better than generalization-specific previous approaches that leverage data augmentation.

4.2 Related Works

Generalization in Reinforcement Learning (RL) has been extensively addressed in recent years^{74 75 21}. Several successful methods have emerged to enhance generalization in deep RL. These methods include the application of regularization techniques such as dropout⁵³, batch normalization^{53 21 52}, and data augmentation^{21 6}. To mitigate observational overfitting,⁷⁶ have introduced a feature-swapping regularization technique. Zhang et al.⁷⁷ and Agarwal et al.⁷⁸ have employed bisimulation metrics to analyze the similarity between states, enabling the learning of task-relevant representations. Decoupled policy and value networks have been proposed to avoid overfitting and bolster generalization^{3 26 28}. Additionally, policy distillation techniques have been employed to enhance generalization, as demonstrated by⁷⁹ and⁸⁰. More recent works have introduced interesting approaches such as the generalist-specialist training framework⁸¹ and pre-trained encoder⁸².

Data Augmentation is an effective way to learn from limited data and has widely been used in computer vision^{83 84 85}. In supervised learning, an augmented version of a sample is assigned the same label and directly used as an independent sample during training^{86 85}. Self-supervised methods exploit the augmented images to generate pseudo labels for pre-text tasks^{87 88 89 71 90}. Recently, data augmentation has gained significant attention in RL to improve the sample efficiency and generalization⁹¹. The naive approach is to simply use the augmented data as an additional sample for RL optimization^{21 92}. Contrastive loss has

been adopted for RL in ⁴ that uses crop as the data augmentation technique. Laskin et al. ⁶⁹ presents a comprehensive study on data augmentation for RL and shows that data augmentation can outperform even approaches dedicated to learning better representation. Hansen and Wang ⁵ introduces non-contrastive loss for RL, which is the most similar work to our approach, and deploys an almost identical non-contrastive mechanism as used in general computer vision. Yarats et al. ⁷⁰ uses augmented data to regularize the Q-function in value-based RL algorithms. Following their work, Raileanu et al. ⁶ proposes to regularize both the policy and value functions in an actor-critic setting. Raileanu et al. ⁶ also presents a way to automatically select the best augmentation technique that achieves better generalization. In our work, we aim to learn augmentation invariant representation using non-contrastive loss.

4.3 Background

4.3.1 Contextual Markov Decision Process

Contextual Markov Decision Process (CMDP) extends the general MDP formulation by introducing the dependence on the context. CMDP allows a set of contexts and every context induces a slight variation in the base MDP thus resulting in a number of MDPs that share similar characteristics but vary in terms of initial state distribution and the transition function. Here, we assume a CMDP is defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, r, \mu_C, \mu_S)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{C}$ is the context space, $\mathcal{T}(s'|s, a)$ is the transition function, r is the reward function, μ_C is the context distribution, and μ_S is the context-dependent initial state distribution. Each episode corresponds to a context sampled according to $c \sim \mu_C$. An initial state is sampled according to $s_0 \sim \mu(\cdot|c)$ and the subsequent states within that episode are sampled based on $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t, c)$. Consider d_π^c as the state distribution that is generated through the execution of the acting policy π in context c . During training, the agent has access to a limited number of contexts. The objective is to learn a generalizable policy π such that the expected return over all possible contexts $\mathcal{G} = \mathbb{E}_{c \sim \mu_C, s \sim d_\pi^c, a \sim \pi(s)}[r(s, a)]$ is maximized, where each context offers highly diverse observations.

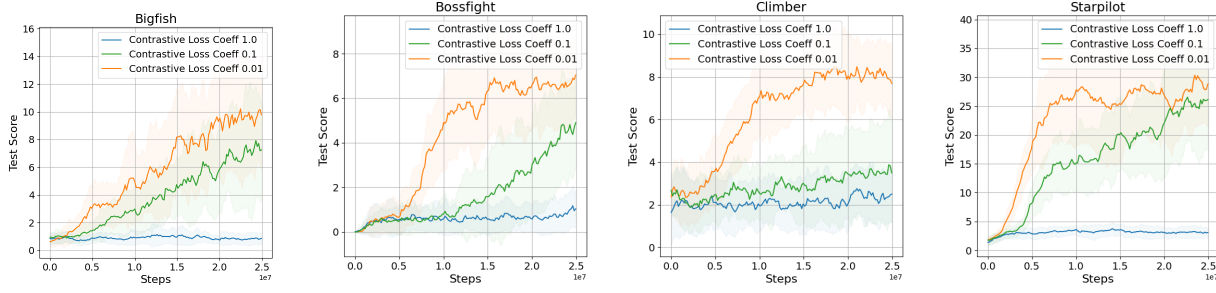


Figure 4.1: Instability in the CURL⁴ performance with different coefficients for the auxiliary contrastive loss. The experiments are for "crop" data augmentation in Procgen. It shows how much the contrastive loss coefficient hyperparameter impacts the performance.

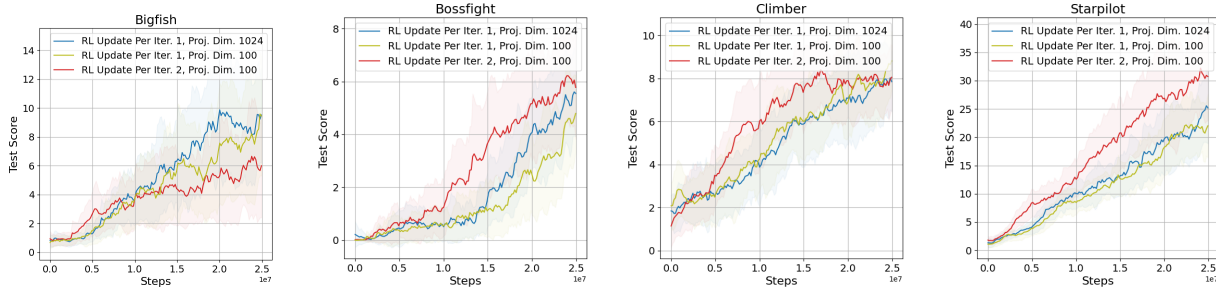


Figure 4.2: Test performance of non-contrastive approach SODA⁵ with different combinations of the hyperparameter RL updates per iteration and projection dimension. The experiments are for "crop" data augmentation. This reveals that the performance highly is sensitive to the hyperparameters.

4.3.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO)⁷ is the most popular actor-critic method. To handle complex visual inputs, deep RL algorithms use deep neural networks for the policy and value function approximation. Traditionally, PPO leverages a shared neural network for policy and value functions. Given that the network is parameterized by θ , PPO optimizes the following objective:

$$J_{PPO}(\theta) = J_{\pi}(\theta) - \alpha_v L_V(\theta) + \alpha_s S_{\pi}(\theta) \quad (4.1)$$

where $J_{\pi}(\theta)$ is the policy gradient objective, $L_V(\theta)$ is the value loss, $S_{\pi}(\theta, \phi_{\pi})$ is the entropy bonus for exploration, and α_v and α_s are the coefficients for the respective terms. For this work, we employ PPO as a foundation and present our representation learning framework

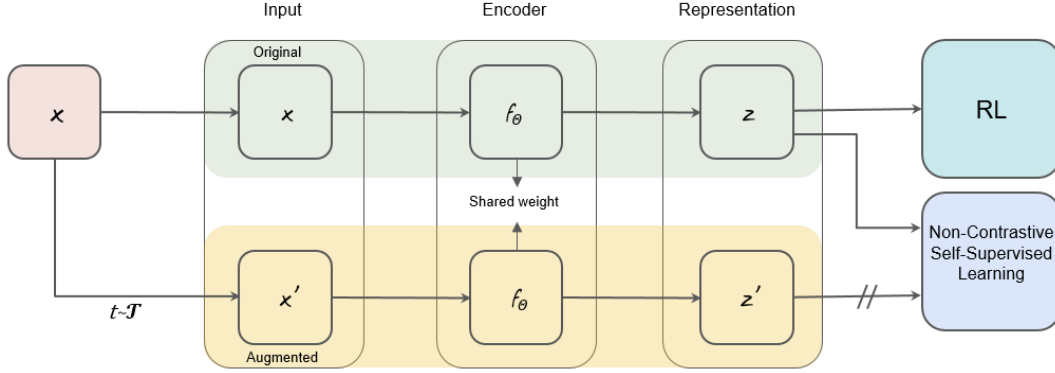


Figure 4.3: Overview of the proposed non-contrastive RL approach. We directly use the raw sample x for the RL objective and use the augmented one x' only for self-supervised representation learning. RAIR does not use any projector or predictor heads.

using standard PPO. Please refer to Section 2.3.1 for more details about PPO.

4.4 Augmentation Invariant Representation for RL

In this section, we describe our proposed approach *RAIR* that enables better representation learning for pixel-based observations and consequently improves the generalization ability of the RL agent in zero-shot settings. We first present that existing contrastive and non-contrastive approaches are highly sensitive to the hyperparameters. Figure 4.1 shows that the CURL performance drastically varies with respect to the contrastive loss coefficient. For Procgen, it works best for a coefficient value 0.01 while the original work⁴ reported that for ALE environments the best results were obtained with the value 1.0. Figure 4.2 shows that the performance of the non-contrastive SODA approach varies significantly with respect to the RL update per iteration value and the projection dimension used for expressive representation learning. Further, we present a rationale behind the poor generalization performance of contrastive learning methods. Such investigation was missing for a long in the previous works that demonstrated the poor performance of the contrastive approach without any arguments^{69 5}. Our critical insight is that even if two observations visually appear different, if they share the same semantic meaning then their latent representation should

be semantically similar. For example, an RGB observation and its grayscale counterpart or a random cropped counterpart denote the same state. We argue that this is sufficient and risk-free compared to more complicated contrastive and non-contrastive approaches. Thus, we propose a *minimalist* framework²⁵, seeking to isolate and attribute performance to algorithmic contributions rather than entangle it with the specifics of implementation such as encoders, projection head architectures, augmentation mechanisms, and similarity measures between observations. In this use of a minimalist approach, we seek to separately and directly optimize the state representation encoder so that it can generate similar latent representations for augmented data while learning the RL objective. Figure 4.3 presents an overview of the approach.

4.4.1 Augmentation Invariant Representation

We propose to use non-contrastive self-supervised learning for RL to ensure that the representations of behaviorally similar states are not being pushed far apart. Contrastive learning is designed to minimize the distance between anchor-positive pairs (similar) while maximizing the distance between anchor-negative pairs (dissimilar). Thus, similar data points are clustered together, and dissimilar points are separated further. However, the notion of dissimilarity in RL is very different. An observation even being so different in visual appearance may have similar consequences if they are behaviorally similar. Thus, it is

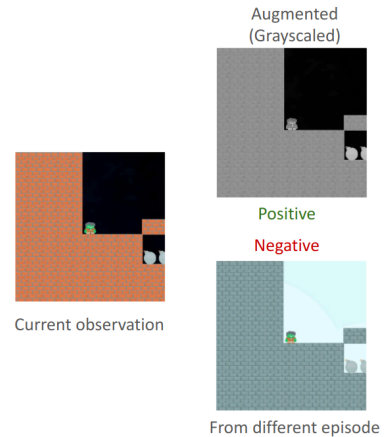


Figure 4.4: Similar positive and negative pair with the anchor (current observation)

hard to distinguish true negative samples without secondary measures. Contrastive learning runs the risk of pushing the behaviorally similar states further in the latent space and disturbs the representation learning task. Figure 4.4 shows an example of such a scenario where two semantically similar states appeared in different trajectories. The optimization under

Table 4.1: Components used in proposed RAIR compared to CURL and SODA.

Method	Target Encoder	Projection Head	Predictor Head	Negative Sample	Augmentation Network	Similarity Measure
CURL	✓	×	×	✓	Online + Target	Bi-linear Inner Prod.
SODA	✓	✓	✓	×	Online	L2 Norm
RAIR	×	×	×	×	Online	Cosine Similarity

contrastive loss will have a contradicting objective where the anchor and augmented image will be matched, however, the "negative" from another trajectory will be pushed apart. But indeed all three are the same. Therefore, we develop a non-contrastive learning approach to RL that uses only the positive pairs generated through data augmentation. For augmented states, we know for sure that our agent should consider them similar. However, to enable non-contrastive learning, as opposed to other existing non-contrastive approaches, we keep our architecture simple and minimal with the aim of reducing the number of hyperparameters and the interference with the RL objective.

To this end, we propose to disentangle the representation learning task from the RL task. Our aim is to learn encoders $f_\phi : \mathcal{S} \rightarrow \mathcal{Z}$ that captures representations of states that are augmentation invariant, at the same time suitable for the control task. We train the encoder to minimize the cosine similarity loss which is defined as follows:

$$J(\phi) = 1 - \cos_s(z, \hat{z}) \quad (4.2)$$

where $z = f_\phi(s)$, $\hat{z} = f_\phi(t_r(s))$; and the state transformation mapping $t_r : \mathcal{S} \times \mathcal{H} \rightarrow \mathcal{S}$ denotes the augmentation function with $\mathcal{H}$ being the set of all possible parameters for $t_r(\cdot)$. The $\cos_s$ refers to the cosine similarity metric,

$$\cos_s(z_i, z_j) = \frac{z_i \cdot z_j}{\max(\|z_i\|_2 \cdot \|z_j\|_2, \epsilon)}, \quad (4.3)$$

Algorithm 1 RAIR: Reinforcement learning with Augmentation Invariant Representation

```
1: Hyperparameters: total number of updates  $N$ , replay buffer size  $T$ , minibatch size  
    $M$ , encoder network params  $\phi$ , params for actor-critic heads  $\theta$ , image transformation  $t_r$   
   with parameters  $v_i$ .  
2: for  $n = 1, \dots, N$  do  
3:   Collect  $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^T$  using  $\pi(\theta)$   
4:   for  $j = 1, \dots, \frac{T}{M}$  do  
5:      $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^M \sim \mathcal{D}$   
6:     for  $i = 1, \dots, M$  do  
7:        $v_i \sim \mathcal{H}$   
8:        $z_i \leftarrow f_\phi(s_i)$   
9:        $\hat{z}_i \leftarrow f_\phi(t_r(s_i; v_i))$   
10:    end for  
11:     $J(\phi) = \frac{1}{M} \sum_{i=1}^M 1 - \cos_s(z_i, \hat{z}_i)$   
12:    Train encoder with  $J(\phi)$   
13:     $J_{PPO}(\theta, \phi) \leftarrow J_\pi + \alpha_v L_V - \alpha_\pi S_\pi$   
14:    Train full actor-critic network with  $J_{PPO}(\theta, \phi)$   
15:  end for  
16: end for
```

where ϵ is a very small number used to avoid division by zero. Other standard similarity metrics such as l_1 distance can also be used in place of cosine similarity. Unlike other algorithms, we directly and solely maximize the agreement between the learned latent representations of similar observations. We do not incorporate any projection or predictor head that introduces more hyperparameters. Our approach also reduces the complexity of learning instance discrimination, which has the risk of destabilizing the RL objective. We use only one augmented observation to generate the target for the self-supervised loss calculation. Our RL update is more principled in the sense that we update the RL objective only using the direct observation received from the simulator. Table 4.1 presents a comprehensive comparison of the components used in RAIR versus other contrastive and non-contrastive approaches.

4.4.2 Combining Representation Learning with RL

In order to lay out a fully functional reinforcement learning method, we incorporate our augmentation invariant representation learning framework with Proximal Policy Optimization (PPO) which is an actor-critic algorithm. We modify the standard actor-critic algorithm by

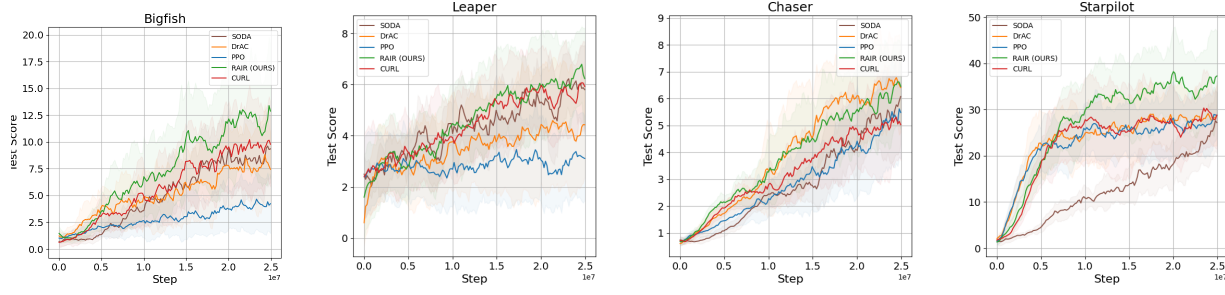


Figure 4.5: Test performance of RAIR (Ours), DrAC⁶, CURL⁴, SODA⁵, and PPO⁷ for "crop" data augmentation in four Procgen environments during training. Mean and std are calculated over 5 trials.

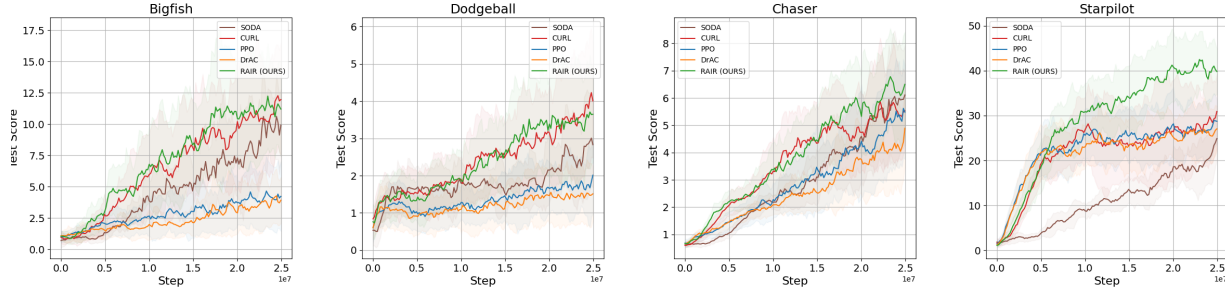


Figure 4.6: Test performance of RAIR (Ours), DrAC, and PPO for "color jitter" data augmentation in four Procgen environments.

adding an additional component that calculates the cosine similarity loss and updates the encoder based on that. Algorithm 1 shows all the steps. After the encoder update, RAIR estimates the PPO objective and jointly updates the encoder and the actor-critic parameters. Thus, RAIR iteratively updates the encoder (only) and the combined policy and value function (including the encoder) in turn. We observe training with a single combined loss function is often susceptible to instability and high variance. After the training phase, the policy π is used to collect data from the environments. Here, the update sequence is crucial - the encoder update should precede the update for the RL objective. Otherwise, there may be a misalignment between the encoder output and the learned policy. This is because if the encoder update occurs after the policy update, then the mapping learned by the policy head may get disrupted.

Table 4.2: Average PPO-Normalized Return (%) of different approaches on test levels across all 16 environments. Evaluated with 5 trials for each environment, each with a different seed.

Augmentation	PPO	DrAC	CURL	SODA	RAIR (Ours)
Crop	100.0	136.0	125.1	131.8	134.0
Color Jitter	100.0	119.3	138.3	129.8	140.4
Grayscale	100.0	95.0	141.1	136.8	130.9
Cutout	100.0	109.2	119.0	113.2	127.4

4.5 Experiments and Discussions

We conduct our experiments on the Procgen benchmark² that offers sixteen procedurally generated environments. Each environment generates levels with diverse dynamics and backgrounds. Each level or episode can be considered as a context as the levels are generated using some seed. Following commonly used standards, we train the agent on 200 contexts or levels and test the agent on the full distribution of contexts. Training on a limited set of levels and testing on the full distribution of possibilities provides the opportunity to evaluate the model’s generalization capability. The objective is to achieve better performance in unseen levels or contexts. We conduct experiments on GTX-2080 GPU and all experiments need nearly 6GB of memory. Figure 4.5 and Figure 4.6 present the experimental results for four Procgen environments. RAIR outperforms standard PPO⁷ and achieves competitive performance with respect to contrastive approach CURL⁴ and previous non-contrastive approach SODA⁵. It should be noted that CURL and SODA need extensive hyperparameter tuning and we also do so to report the best baseline performance. The generalization-specific approach UCB-DrAC⁶ performs well while using crop as a data augmentation method (Figure 4.5), however, does not perform well enough with other augmentations such as color jitter (4.6). Our findings suggest that self-supervised learning-based approaches both contrastive and non-contrastive can take the benefit of any type of data augmentation, while our proposed minimalist non-contrastive approach performs comparatively better in many cases or matches the performance with minimum overhead. Table 4.2 presents the PPO-normalized return (%) aggregated over all 16 environments from the Procgen benchmark across dif-

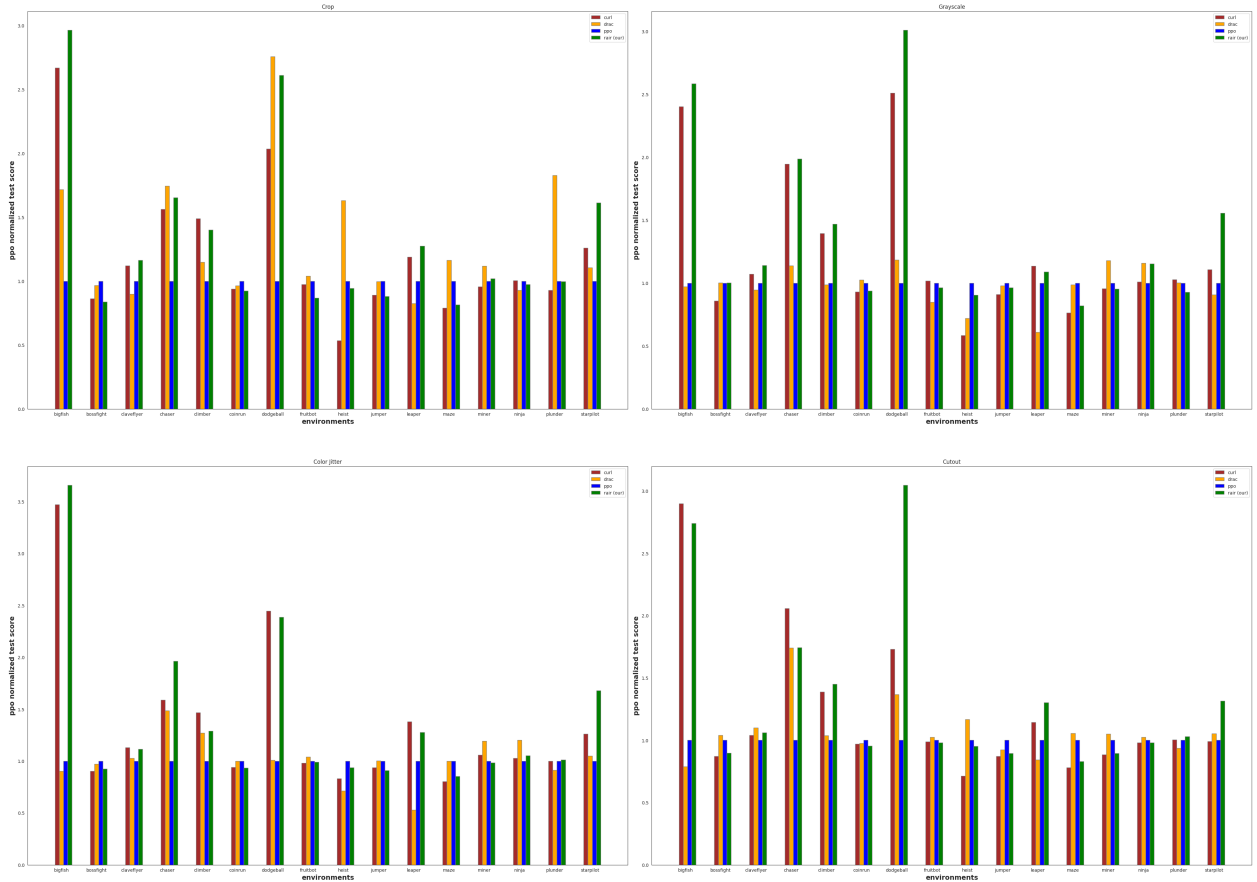


Figure 4.7: PPO normalized test score of RAIR (Ours), DrAC, CURL, and PPO for "crop", "grayscale", "color jitter", and "cutout" data augmentation for all 16 Procgen environments. Each agent is trained on 200 training levels for 25M timestep and then evaluated on 100 episodes. Means are calculated over 5 trials for every environment.

ferent augmentation types. In Figure 4.7, we present results for four data augmentation techniques: *crop*, *grayscale*, *cutout* and *color jitter* across all sixteen environments from the Procgen benchmark. We removed the SODA experiments as they did not outperform others in the previous experiments. From both evaluations, it is evident that RAIR remains consistent across environments and augmentation types. We also observe similar results using other augmentation techniques like flip, rotate, cutout color, and random convolution.

Figure 4.8 presents the ppo normalized score for different data augmentation techniques across all 16 Procgen environments over the course of training. This helps to assess the sample efficiency. The results show that our proposed non-contrastive approach dominates in all of them, except crop, while the data regularization-based approach DrAC drastically

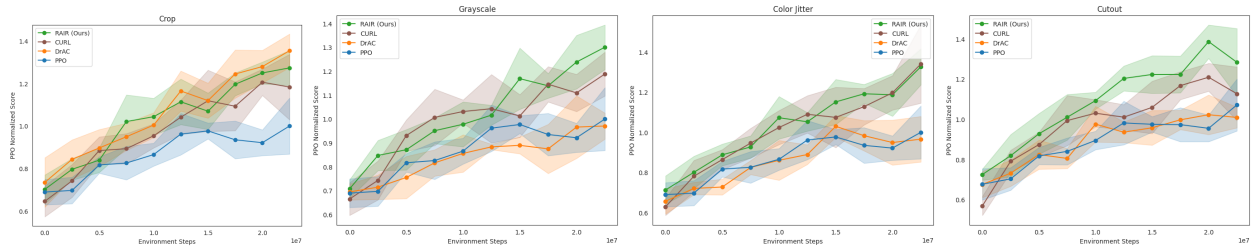


Figure 4.8: PPO normalized test performance of RAIR (Ours), DrAC, CURL, and PPO for ”crop”, ”grayscale”, ”color jitter”, and ”cutout” data augmentation across all 16 Procgen environments. Mean and std are calculated over 5 trials with different seeds.

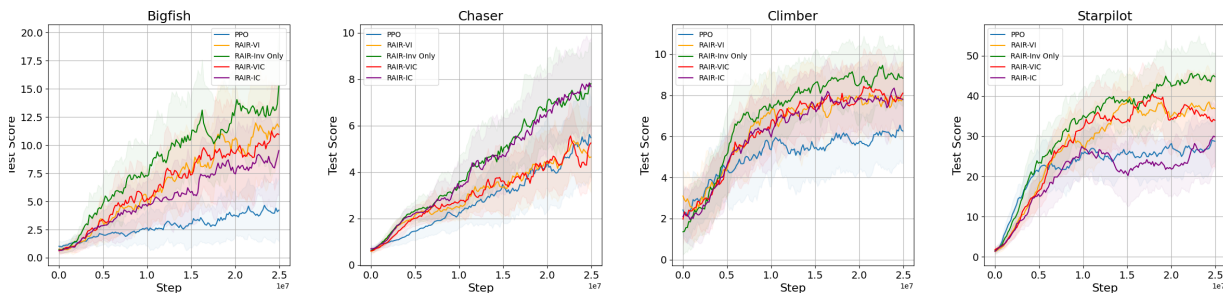


Figure 4.9: Test performance of our proposed cosine similarity-based invariance learning vs. variance-invariance-covariance based regularization method (VICReg) and its different ablations (removing one term at a time). Means and standard deviations are calculated over 5 trials.

fails. The contrastive counterpart performs competitively.

4.5.1 Impact of Variance and Covariance with Invariance

Zbontar et al.⁹³ shows the importance of the covariance-based redundancy reduction term in the non-contrastive loss function. Going forward Bardes et al.⁷¹ proposes to add a variance term that ensures the standard deviation (over a batch) of each variable of the latent embedding vectors. Such a term makes the embedding vectors different for different samples within a batch. In this section, we analyze the impact of adding such variance and covariance terms in addition to the invariance in the context of RL. We observe that while such additional regularization works well for general computer vision tasks, they do not help much in RL when compared with simple non-contrastive invariance learning. Figure 4.9 shows that our

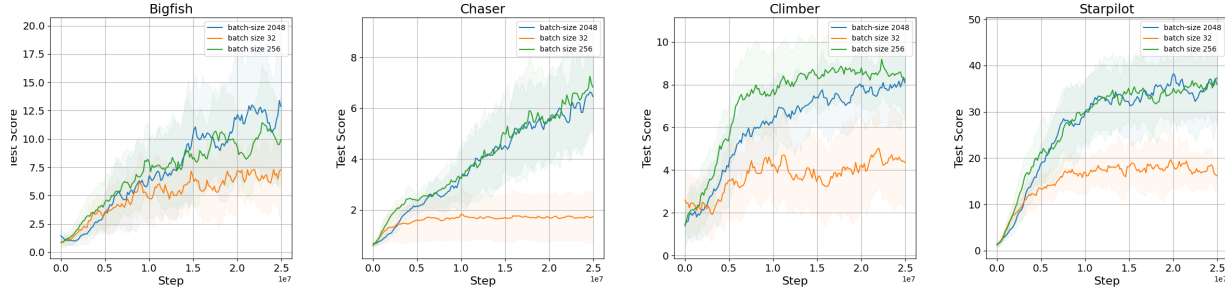


Figure 4.10: Analyzing the impact of batch size.

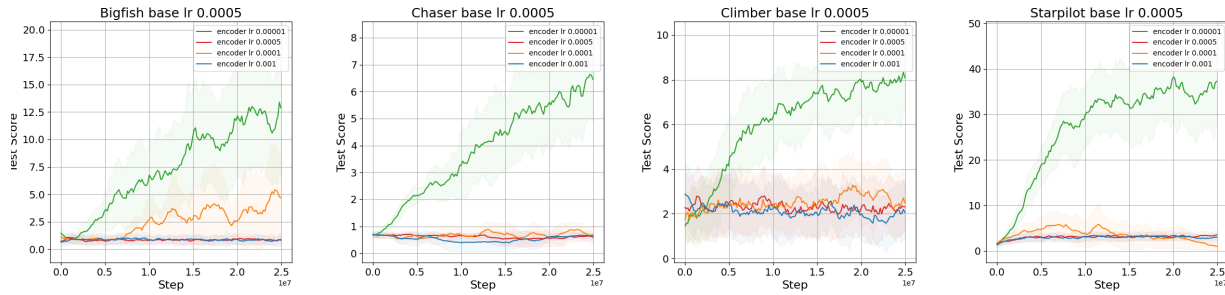


Figure 4.11: Analyzing the impact of encoder learning rate while keeping the learning rate of the optimizer that optimizes RL objective (which is set to 0.0005).

cosine similarity-based invariance learning outperforms in most of the environments.

4.5.2 Relation to the Batch Size

Contrastive loss is known to perform well only with large batch sizes, however, large batch size introduces high computational overhead. Thus, we investigate how the batch size impacts in case of non-contrastive loss within the realm of RL. We experiment with three different batch sizes $\{32, 256, 2048\}$ and find that our proposed non-contrastive method is robust against the variation of batch size, however, too low batch size can still negatively impact the performance.

4.5.3 Sensitivity to Hyperparameters

The main hyperparameter for our approach is the learning rate of the encoder optimizer as we use two different optimizers. One for the encoder only and one for the whole network including the encoder and the policy/value heads. We observe that the learning rate for the encoder network should be lower than the master network to an extent to avoid any over-shooting. To test the performance variation, we fix the learning rate of the main optimizer and vary the encoder learning rate. The learning rate for the main optimizer was set to 0.0005 and the encoder learning rate was swept over the values from the set $\{0.001, 0.0005, 0.0001, 0.00001\}$. Figure 4.11 shows that when the encoder learning rate is 0.00001 and the main optimizer’s learning rate is 0.0005 which is higher than the encoder learning rate, the performance is significantly higher compared to other combinations. On the other hand, the baseline contrastive and non-contrastive learning methods are highly sensitive to the hyperparameters. Figures 4.1 and 4.2 present the results of some of our hyperparameter searches for the baselines. Our code is publicly available. ¹

4.5.4 Importance of Update Sequence

RAIR employs a phasic technique where first the encoder is updated based on the non-contrastive loss and then the whole actor-critic parameters are updated based on the RL objective. We analyze how the update sequence affects the training. Thus, we create an alternate version where the actor-critic update precedes the non-contrastive encoder update. We observe that the best performance is achieved when the encoder update happens prior to the RL update. This is because if the RL update occurs earlier, then the weight update in the encoder through self-supervised loss will create incompatibility with the fully connected layers used for policy and value head. Thus, collecting new data with that inconsistent network may lead to sub-optimal samples.

¹<https://github.com/nasiknafi/rair>

4.6 Conclusion

In conclusion, this chapter introduces RAIR (Reinforcement learning with Augmentation Invariant Representation), a new approach aimed at enhancing the generalization performance of reinforcement learning (RL) agents through data augmentation. Unlike existing methods that directly use augmented data or complex contrastive/non-contrastive learning mechanisms, RAIR takes a distinct approach by simply adding cosine similarity-based non-contrastive loss for RL with a straightforward encoder. RAIR eliminates the drawbacks of contrastive learning and at the same time, achieves similar or better performance compared to similar non-contrastive approaches with minimum overhead in terms of architecture design and hyperparameter tuning. However, additional experiments with more benchmarks may strengthen the results. RAIR aims to learn latent representations of an observation that is invariant to the representation of its corresponding augmented versions. Thus, the proposed approach leads to agents that are more robust in adapting to changes in the environment, leading to improved generalization. Thus, we would like to conclude that simplicity in design and auxiliary loss wins in RL because this minimally interferes with the RL objective.

Chapter 5

Value Target Shaping to Address Uncertainty in Episode Length

5.1 Introduction

Generalization to unseen variations of an environment is an indispensable aspect of current reinforcement learning research. Similar to any deep neural network-based function approximator, deep RL agents are prone to overfitting. They tend to memorize the training data distribution, particularly the training episodes. Previously, the diversity between training and testing scenarios was not well formulated due to the limitation of the simulated environment design, hence the issue of overfitting was somewhat overlooked. Recent benchmarks whose designs are based on the Contextual Markov Decision Process (CMDP) framework are capable of generating different episodes corresponding to different variations of the environment, while keeping the task objective fixed⁹⁴. Research that leverages these CMDP-based environments (often referred to as multi-environment) has shown that even in the case of minor changes or perturbations to the environment, RL agents fail to generalize^{3;21}.

The poor generalization performance of an RL agent stems from the overfitting of the value function. To minimize the prediction error, the value function often establishes some spurious correlation with the observed state by prioritizing features that are specific to some

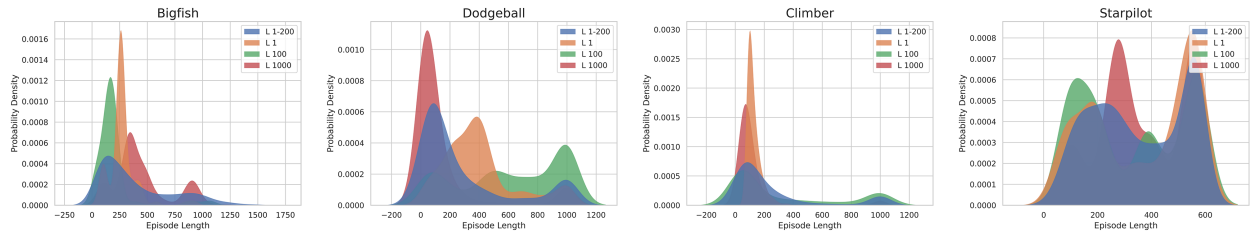


Figure 5.1: Distribution of task completion time or episode lengths of four Progen environments for a fully trained PPO agent estimated based on 1000 episodes randomly sampled from the 1 to 200 training levels vs. only level 1, level 100, and level 1000. The difference in the distribution over multiple levels vs. a single level is clearly visible.

episode or trajectory. Thus, regularization has been shown to be an effective overfitting prevention technique not only for singleton environments but also for CMDP-based multi-environment settings^{52;76}. However, regularization suffers from premature convergence, leading to suboptimal solutions. Moon et al.⁴⁶ shows that learning a good estimate of the value function for multi-environment settings is harder than for singleton environments. Due to the variations in such settings, the agent generally observes fewer samples from a particular environment. Thus, the agent relies more on memorization to better estimate the value function and collapses in unseen states.

In this work, we take a closer look at the value function estimation in CMDPs, where episodes vary based on contexts and an agent must learn a value function (and consequently a policy) that is optimal over all contexts including the unseen ones beyond the training contexts. We first show that the task completion times or episode lengths vary remarkably when multiple contexts are considered compared to a single one. Figure 5.1 shows the distribution of episode lengths for four Progen environments for a trained PPO agent. The distribution is estimated based on 1000 episodes completed by the agent which are sampled from the training contexts or levels 1 to 200. We contrast this distribution with the ones obtained using a single level such as the level with seeds 1, 100, and 1000. Because of the variability of the episode length, the value estimates of a semantically similar state will differ in different episodes. This variability significantly impacts the accurate value estimation for those similar states.

To address the issue of value estimation in unseen episodes or contexts of the environment, we propose a new approach, Augmented Value Targets (AVaTar), which augments multiple value function targets based on the observed reward and reshapes the traditional target. Our novel contribution is that these targets generated using distinct discount factors γ_i account for episode length variation and the value function is optimized based on the average *augmented value targets*. We demonstrate that this average is bounded above by the value loss as if those value targets were used as an independent sample during batch optimization. Thus, using the average of the augmented value targets acts as an efficient approximation. We evaluate our proposed target augmentation scheme on all sixteen environments in the Procgen benchmark and the challenging Crafter benchmark. Experimental results show that AVaTar significantly outperforms the standard policy gradient approach Proximal Policy Optimization (PPO)⁷. Moreover, we show that our proposed approach achieves better performance compared to the recent generalization-specific method UCB-DrAC⁶ on Procgen, and when integrated with the same method it further improves the performance.

5.2 Preliminaries

5.2.1 Contextual Markov Decision Process

A Contextual Markov Decision Process (CMDP) extends the general MDP formulation by introducing the dependence on the context¹⁴. CMDP allows a set of contexts and every context induces a slight variation in the base MDP thus resulting in a number of MDPs that share similar characteristics but vary in terms of initial state distribution and the transition function. Here, we assume a CMDP is defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, r, \mu_C, \mu_S)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{C}$ is the context space, $\mathcal{T}(s'|s, a)$ is the transition function, r is the reward function, μ_C is the context distribution, and μ_S is the context-dependent initial state distribution. Each episode corresponds to a context sampled according to $c \sim \mu_C$. An initial state is sampled according to $s_0 \sim \mu(\cdot|c)$ and the subsequent states within that episode are sampled based on $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t, c)$. Consider d_π^c as the state distribution that

is generated through the execution of the acting policy π in context c . During training, the agent has access to a limited number of contexts. The objective is to learn a generalizable policy π such that the expected return over all possible contexts $\mathcal{G} = \mathbb{E}_{c \sim \mu_C, s \sim d_\pi^c, a \sim \pi(s)}[r(s, a)]$ is maximized.

In this work, we experiment with the Procgen² benchmark that offers sixteen procedurally generated environments. Each environment refers to a CMDP and each level or episode that is being generated using a random seed refers to a context. Thus, we can consider each level of the game as a sampled context c . We train the agent on 200 contexts or levels and test on the full distribution of contexts where full distribution refers to a nearly infinite set of procedurally generated levels. Thus, the size of the training set is smaller than the test set enabling the evaluation of the model’s generalization capability beyond those 200 training contexts. We also use the Crafter²⁴ benchmark that offers diverse episodes through different topological maps and distribution of entities as well as tasks. The aim is to perform better in unseen contexts beyond the training ones, where the contexts define the variation in backgrounds, entities, and dynamics.

5.2.2 Proximal Policy Optimization

We build upon the Proximal Policy Optimization (PPO) algorithm for this work. PPO is the current standard baseline for policy gradient approaches⁷. PPO is a practical and efficient adaptation of the Trust Region Policy Optimization (TRPO)⁴⁰ method. PPO generally uses a shared neural network as policy and value function approximators. If the network is parameterized by θ , then PPO optimizes the following objective function :

$$J_{PPO}(\theta) = J_\pi(\theta) - \alpha_v L_V(\theta) + \alpha_s S_\pi(\theta) \quad (5.1)$$

where $J_\pi(\theta)$ is the policy gradient objective, $L_V(\theta)$ is the value loss, $S_\pi(\theta, \phi_\pi)$ is the entropy bonus for exploration, and α_v and α_s are the corresponding coefficients. Please refer to Section 2.3.1 for a detailed overview of PPO.

5.3 Methodology

In this section, we describe Augmented Value Targets (AVaTar), our approach to achieving better generalization in contextual CMDP. It is designed to address the limitations of accurate value estimation. Particularly, the motivation comes from the necessity of a value estimator that can accurately estimate the value of a similar state encountered in a test (unseen) context or episode. As the value estimate may vary between the episodes or contexts, we need a flexible estimator that does not overfit just to the observed value target during training. Thus we would like to have a value estimator that will respect the variability in the value estimate and learn value representation such that it can make a balance between the context-specific and context-agnostic features.

5.3.1 Intuition

The key idea behind our approach is to generate some pseudo-value target that will mimic the value of a state in an unseen context. We observe that CMDP-based environments allow the generation of diverse episodes or levels (contexts). The diversity comes in the form of background color, level formation (e.g. varying number of obstacles), enemy distribution, etc. These contextual variations inherently induce a variation in the possible episode length (See Figure 5.1). For say, an agent may need to cross five waterbodies at an unseen level as opposed to two or three frequently encountered in the training scenarios. Crossing five instead of two or three will incur lengthy episodes even if the agent follows an optimal policy. For some environments, the variation in the episode length can be due to the variation in spawned enemies in the environment. Depending on the number of enemies that the agent needs to tackle, it may require different amounts of timestep to finish the episodes.

Starting from the same state, if the lengths of episodes vary significantly, then this will contribute to changing values for earlier states in a trajectory³. As such, the same final reward may be heavily discounted, leading to a much smaller value estimate, if the episode is too long. On the other hand, a shorter episode would impart a higher value estimate to the same reward. Thus, an agent fails to accurately predict the value of future rewards

Algorithm 2 AVaTar: Augmented Value Targets

```
1: Hyperparameters: Total number of updates  $N$ , replay buffer size  $T$ , number of epochs  
   per update  $E$ , number of discount factors  $n_\gamma$ , initial network parameters  $\theta$ , Gaussian  
   mean  $\mu$ , std  $\sigma$ , truncation range  $(r_l, r_u)$   
2: for  $n = 1, \dots, N$  do  
3:   Collect  $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^T$  using  $\pi(\theta)$   
4:   Sample  $n_\gamma$  number of discount factors from a truncated Gaussian Normal with given  
   parameters  
5:   Compute augmented value targets  $\hat{V}_t^{\gamma_i}$  for all sampled discount factors  $\gamma_i$  where  $i =$   
    $1, \dots, n_\gamma$  and for each state  $s_t$   
6:   Calculate the average of the value targets  $\hat{V}_t^{avg}$  using all  $\hat{V}_t^{\gamma_i}$  and the advantage  
   estimation  $\hat{A}_t^{avg}$   
7:   for  $i = 1, \dots, E$  do  
8:      $L_V(\phi) = \hat{\mathbb{E}}_t \left[ \left( V_\theta(s_t, a_t) - \hat{V}_t^{avg} \right)^2 \right]$   
9:      $J_{AVaTar}(\theta) = J_\pi(\theta) + \alpha_v L_V(\theta) - \alpha_\pi S_\pi(\theta)$   
10:     $\theta \leftarrow \arg \max_\theta J_{AVaTar}$   
11:   end for  
12: end for
```

in unseen episodes or contexts, particularly in regard to widely varying episode lengths. Therefore, variation in episode lengths implies variation in the value target.

To avoid the possibility of overfitting to a value estimate that corresponds to a particular constant episode length, we propose to generate some value targets that would have resulted from the variation in episode length. However, there is no way to anticipate a possible unseen episode length. In this work, we identify the connection between episode length and discount factor. In the next section, we discuss how discount factors can be leveraged to augment value targets corresponding to different episode lengths.

5.3.2 Augmented Value Targets (AVaTar)

Consider a particular state s appears n times (in a semantically similar way but may be in different visual appearances due to the context) in the full set of context or trajectory (level) D_L , and takes n different values corresponding to each episode length. According to the problem setup, the set of the training context D_S will be a smaller subset of the full set of contexts. Thus, $|D_S| \ll |D_L|$. So, it may happen that only a few contexts say m where s

is present will appear in the training context and $m \ll n$. Because of this, in an actor-critic architecture, the critic network will be optimized using the discrepancy in the prediction of the return observed only in m observation and will try to generalize to the rest of the unseen $m - n$ scenarios.

It is obvious that if we would have all n observations of state s in the training set, then the model would learn based on more contexts and achieve better performance across all contexts. However, that is not possible in the limited training context setting. But if we can somehow augment the training set $|D_S|$ with additional such targets, then the model will not overfit to the m observation but rather learn to predict the value across all contexts.

We identify that by changing the discount factor γ we can effectively create many pseudo-targets for imaginary episodes with varying lengths based on the current episode or context.

Proposition 1. *For any two episodes with length L_1 and $L_2 = a \times L_1$, where a is a real number s.t. $a > 0$ and $a \neq 1$, there exist two distinct discount factors γ_1 and γ_2 respectively such that they yield the same value of discounting for the initial state.*

Proof. $\gamma_1^L = \gamma_2^{aL} \implies \gamma_2 = \gamma_1^{1/a}$.

Since $a \neq 1$, γ_1 and γ_2 are distinct.

For example, if we have an episode with a length of 100 and the discount factor in a fixed discount setting is 0.9, then the value estimate of the initial state will be $0.9^{100} = 0.000026$ given that there is a reward +1 at the end. Similarly, if the episode length is 50, then the value estimation will be $0.9^{50} = 0.005154$. Thus, based on our proposition there exists another gamma such that the discounted value estimate for length 50 will correspond to an estimate with length 100,

$$0.9^{50} = 0.005154 = 0.9487^{100}. \quad (5.2)$$

This shows that we can mimic a value estimate for a state within an episode with a length 50 just from the episode with a length 100. Without loss of generality, this can be extended to rewards other than 1. In such cases, this will denote the value of effective discounting corresponding to that reward. Further, this can be extended to any intermediate state other than the initial state by considering the variation in the rest of the episodes starting from the

current state. The final concern is the unknown reward distribution in imaginary episodes. Thus, we restrict our episodes to reward-preserving episodes. This means we consider the amount of reward and the sequence they appear will be the same as we observed. This helps to avoid any overestimation or underestimation of the value function and at the same time only focuses on the variance that evolved from the variation in episode length.

Now, based on the explanation just presented, we propose to perturb the fixed discount factor by sampling a set of discount factors from a distribution to generate value estimates corresponding to some imaginary episode length. In our implementation, we sample n_γ distinct discount factors from a truncated Gaussian distribution with a mean equal to 0.99, the most commonly used discount factor, and a small standard deviation of 0.009. The rationale behind using a small standard deviation is the fact that we do not have any prior knowledge about how much the episode length can vary. Thus, introducing small variations is a suitable design choice.

In the next phase, the most important aspect is how to leverage those generated value estimates. In the case of data augmentation or domain randomization approaches, it is customary to add the augmented or generated samples in the experience replay buffer so that they can be sampled together with the original data collected directly through interaction with the environment. Following the same approach, here the newly created values can be added to the experience pool. However, as the state representation here is the same, rather we are just creating some pseudo-value target against the same state, it would be unnecessarily complex to manipulate the experience replay. Thus, in the next section, we provide an efficient alternative based on the principle of the mini-batch optimization process.

5.3.3 Efficient Computation using Average Augmented Target

Assume we have computed n_γ different value targets (cumulative discounted returns) for n_γ distinct discount factors, $\gamma_1, \gamma_2, \dots, \gamma_{n_\gamma}$. If we would execute an optimization using gradient descent considering all these augmented value targets as an independent sample of a batch,

then the value loss will be calculated as the average of the squared loss as follows:

$$L_v^{avg} = \frac{1}{n_\gamma} \sum_{i=0}^{n_\gamma} (V_i - V_s)^2 \quad (5.3)$$

Here, V_i corresponds to the value estimate for γ_i . V_s is common for all the samples because we are updating against a fixed sample but with different value targets. From 5.3, using basic algebra it follows that,

$$\frac{1}{n_\gamma} \sum_{i=0}^{n_\gamma} (V_i - V_s)^2 > \left(\frac{\sum_{i=0}^{n_\gamma} (V_i - V_s)}{n_\gamma} \right)^2 \quad (5.4)$$

Further, we can rewrite the equation 5.4 as

$$\frac{1}{n_\gamma} \sum_{i=0}^{n_\gamma} (V_i - V_s)^2 > \left(\frac{\sum_{i=0}^{n_\gamma} V_i}{n_\gamma} - V_s \right)^2 \quad (5.5)$$

The right-hand side of the equation. 5.5 shows that if we take the average of the generated value targets $\hat{V}_t^{avg} = \frac{\sum_{i=0}^{n_\gamma} V_i}{n_\gamma}$ and calculate the value loss with respect to that average then the loss will be upper bounded by the average value loss considering the targets individually. From a practical perspective, the right-hand side is easier to implement as we just need to calculate the average of the targets based on the reward observed with different γ s. As any other component of an existing policy optimization approach does not need to be altered, integration of this approximation will be straightforward. Being a lower approximation of the equation 5.3, it also reduces the risk of overestimation that may occur due to the augmented value targets. Our approach can be considered as a bootstrapped value target using randomly sampled discount factors. Algorithm 2 shows the details of the final algorithm. The overall process is almost identical to the generic PPO, however, we compute the extra value targets by sampling different discount factors and then the average of those each time before entering the update loop (marked in blue).

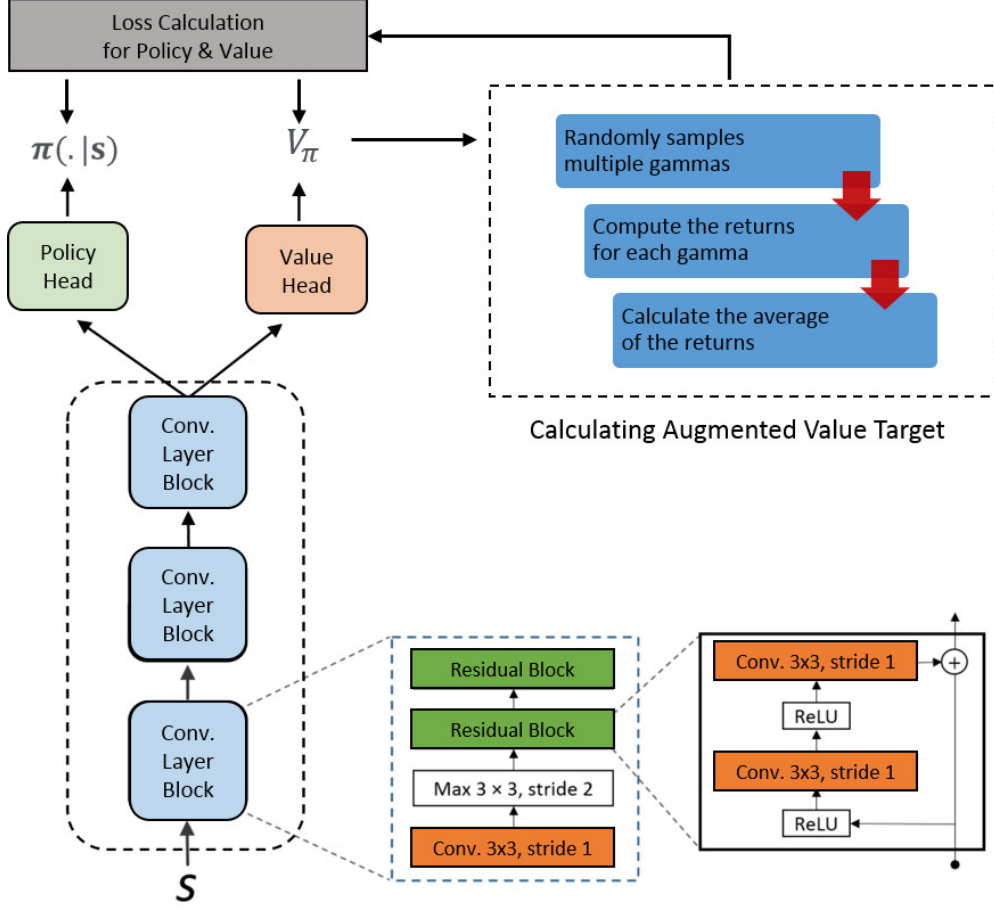


Figure 5.2: Details of the architecture used and the overall approach. Each block shown on the left side is almost identical to the IMPALA CNN architecture¹. Our calculation takes place based on the value prediction.

5.3.4 Network Architecture

We implement our model using large IMPALA-CNN architecture as our backbone motivated by the recent works on Progen^{2;3;16}. This relatively large model maintains a good balance between the reward achieved by learned policies in the highly diverse environment and required computational power². This deeper IMPALA CNN architecture incorporates 15 convolutional layers divided into three blocks¹ containing multiple convolutional layers. Each block includes a Conv - Pooling - Residual Block - Residual Block configuration, as detailed in the bottom-middle part of Figure 5.2. Each residual block has two Conv layers with a ReLU activation layer. Finally, the policy head and the value head consist of fully

Table 5.1: List of hyperparameters used for AVaTar

Hyperparameter	Values
# timesteps per rollout	256
# epochs per rollout	1 (Procgen), 5 (Crafter)
# minibatches per epoch	16
entropy bonus	0.01
clip range	0.2
λ for GAE	0.95
optimizer	ADAM
learning rate	5e-4
total timesteps	25M
Gaussian Mean for γ	0.99
Gaussian Std Dev for γ	0.009
Truncation Range	0.5 - 0.9999

connected layers. Our code is publicly available.¹ We conduct a hyperparameter search over the number of epochs per rollout $E \in [1, 3, 6]$ and found $E = 1$ works best on Procgen and $E = 5$ on Crafter. The other hyperparameters are listed in Table 5.1. For the baselines, we use the optimal hyperparameters from the corresponding literature.

5.4 Results and Discussions

We evaluate our approach using the two benchmarks Procgen² and Crafter²⁴. They offer an expansive variety of procedurally generated levels, making it an ideal choice to explore an agent’s generalization capabilities. Following the guideline from², for Procgen, we trained the model for 25 million time steps using the *easy* difficulty setting. The agent is trained on 200 levels and tested on the full distribution of levels. Every environment in the Procgen benchmark generates observations of size $64 \times 64 \times 3$ and the action space is comprised of 15 discrete actions. As an evaluation metric, we report the average return achieved by the agent in the test distribution of levels or episodes. For Crafter, following their evaluation protocol²⁴, we trained the agent for 1M timesteps and reported the Crafter score and the

¹<https://github.com/nasiknafi/AVaTar>

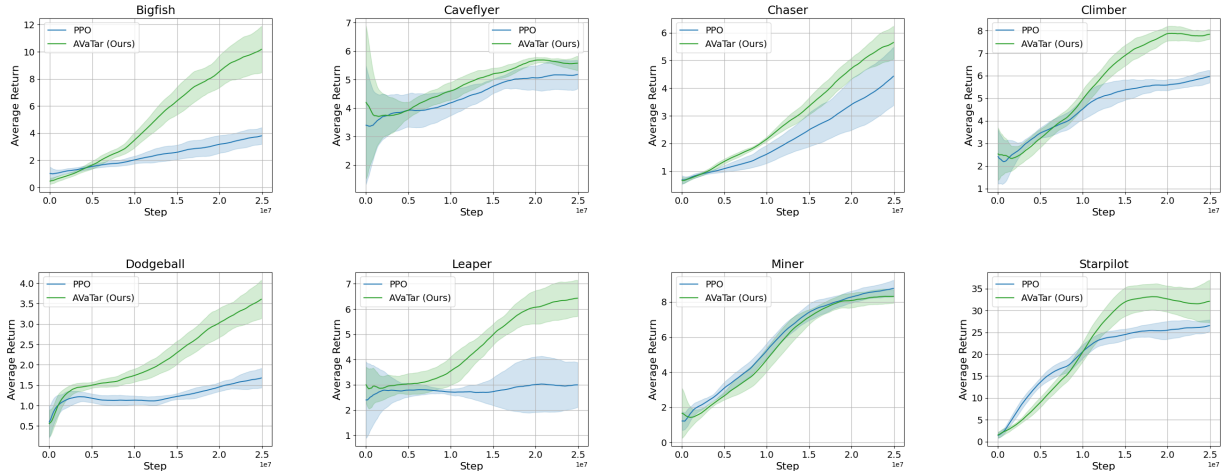


Figure 5.3: Test performance of our proposed Augmented Value Targets (AVaTar) and the standard PPO on Procgen environments.

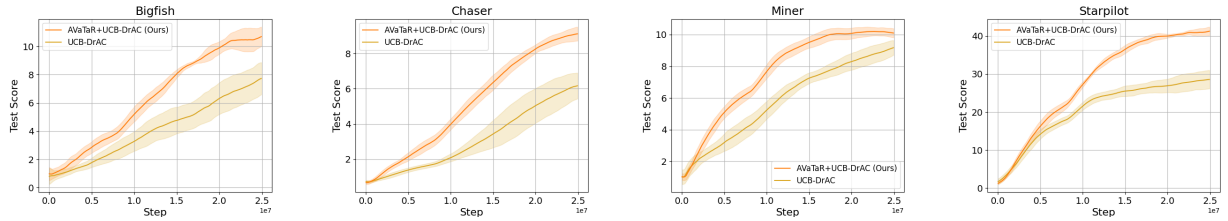


Figure 5.4: Comparison of the generalization-specific SOTA approach UCB-DrAC⁶ and proposed corresponding Augmented Value Targets (AVaTar) version $AVaTar + UCB-DrAC$ over four Procgen environments.

achieved reward.

5.4.1 Generalization Performance on Test Distribution

We first compare our proposed AVaTar with the standard and widely used policy optimization baseline PPO⁷ for all the 16 environments in Procgen. Figure 5.3 presents the results on the test distribution of the levels for 8 out of the 16 environments from Procgen. The presented rolling average and standard deviations are calculated over five trials. It is evident from Figure 5.3 that the proposed AVaTar significantly outperforms PPO on the unseen test levels. We also compare and combine AVaTar with one of the state-of-the-art approaches Data-regularized Actor-Critic (UCB-DrAC)⁶. UCB-DrAC dynamically determines the best

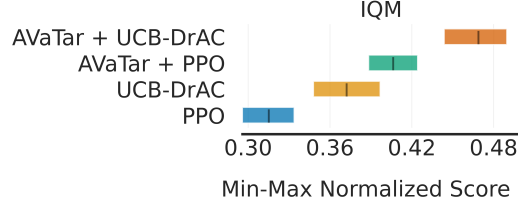


Figure 5.5: Comparison of Interquartile Mean (IQM) of the Min-Max normalized score across all 16 Procgen environments. Both AVaTar versions outperform PPO⁷ and UCB-DrAC⁶. The colored region shows the 95% confidence interval.

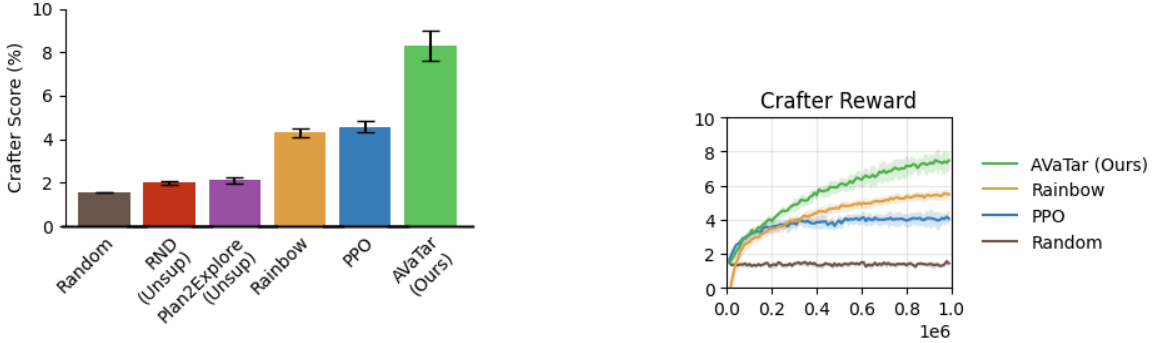


Figure 5.6: (Left) Crafter score of our AVaTar compared to standard PPO⁷ and popular Rainbow⁸ along with other approaches; (right) Comparison of the reward achieved by each agent during 1M timestep. Proposed AVaTar outperforms other approaches considering both metrics. We use 10 different seeds for our experiments.

form of data augmentation that will benefit achieving generalization given an environment. We select UCB-DrAC as this approach uses regularization for both policy (actor) and value (critic) functions through data augmentation. Further, UCB-DrAC outperforms general regularization techniques such as L2-regularization²¹. Figure 5.4 shows results for four of the Procgen environments and the results indicate that incorporating proposed augmented value targets with UCB-DrAC (AVaTar + UCB-DrAC) outperforms the base UCB-DrAC. This shows that AVaTar can be integrated with any existing RL algorithms. In addition, AVaTar does not introduce any additional computing overhead.

We further show a comparison based on the Interquartile Mean (IQM) of Min-Max normalized scores (average returns) for all the approaches in Figure 5.5 as proposed by⁶⁷. This score has been calculated considering all the 16 Procgen environments, thus showing the

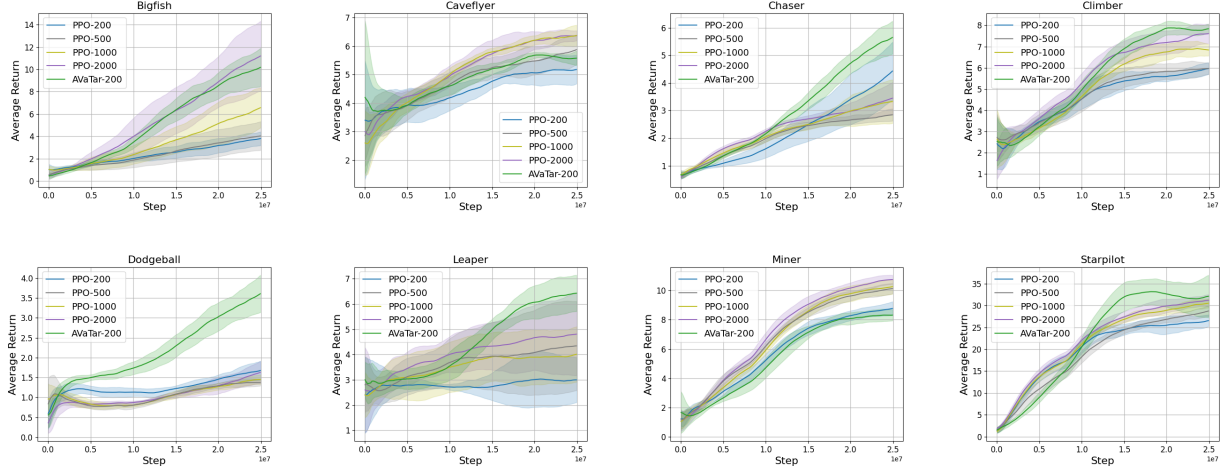


Figure 5.7: Test performance of standard PPO with the increasing number of training contexts (levels) and our proposed Augmented Value Targets (AVaTar) with only 200 contexts over eight Procgen environments. It is evident that while training on 200 levels AVaTar can achieve performance gain that is even better than the PPO with 2000 training levels.

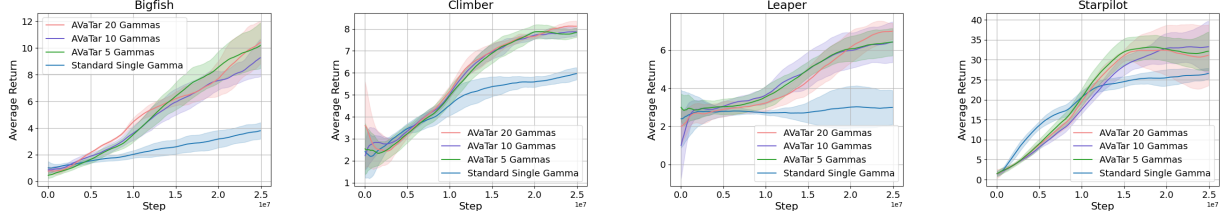


Figure 5.8: Test performance of our proposed Augmented Value Targets (AVaTar) with the varying number of gammas $n_\gamma = \{5, 10, 20\}$. We observe that the performance differences with varying gamma are marginal compared to the performance gain with respect to the fixed gamma (the standard PPO).

aggregate performance across the whole Procgen benchmark. The results indicate that both AVaTar versions achieve higher IQM than UCB-DrAC and PPO. Agarwal et al.⁶⁷ shows that IQM is more reflective of overall performance than the median, as it takes into account 50% of the combined runs rather than just considering the performance order. Also, IQM is known to be more robust to outliers than the mean.

Figure 5.6 shows that AVaTar outperforms both PPO⁷ and Rainbow⁸ algorithms in terms of crafter scores and achieved rewards. Crafter defines 22 achievements for the agent covering a wide range of difficulties. The crafter score represents the agent’s ability to accomplish multiple achievements in the environment instead of repetitively unlocking the

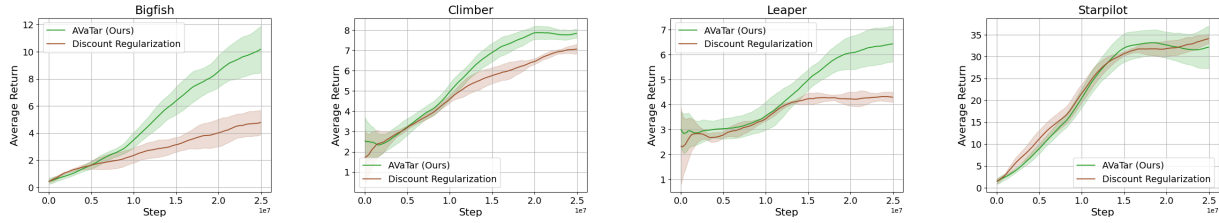


Figure 5.9: Comparison of proposed Augmented Value Targets (AVaTar) with discount regularization. Discount regularization uses a fixed but slightly lower discount factor as a regularizer⁹.

same achievements.

5.4.2 Assessing the Generalization Performance

To better assess the generalization performance, we compare AVaTar with different PPO models that leverage an increased number of training levels. A model generally performs better in terms of generalization if it encounters a large number of contexts during training. Thus, we train different PPO models with 200, 500, 1000, and 2000 contexts or levels from the training set. As expected the performance increased in almost all cases with the increase in training contexts. On the other hand, we train the proposed AVaTar on only 200 contexts from the environment. Figure 5.7 shows that AVaTar (green line) while learning from only 200 contexts, can still achieve generalization better than a PPO agent that is trained on 2000 contexts.

5.4.3 Additional Analysis

Figure 5.8 presents the results of varying numbers of discount factors. We experiment with three different values for the number of γ s ($n_\gamma = \{5, 10, 20\}$). The experimental results show that the performance does not differ that much due to the number of discount factors. However, the performance gain compared to the standard fixed discount factor is clearly evident.

Figure 5.9 shows the results of comparison with a discount regularization-based ap-

proach⁹. Proposed AVaTar outperforms such fixed lower discount factor (0.995 instead of standard 0.999) based regularization⁹.

5.5 Related Works

Recent work has suggested that lack of generalization is an endemic issue in deep reinforcement learning, with state-of-the-art algorithms tending to overfit to the environment, thus leading to models that simply memorize surface-level aspects of the environment^{21;74;75}. However, the ideal and intelligent RL agents should strive to avoid overfitting and be able to generalize effectively to previously unseen data^{17;18;50;80}.

Regularization techniques such as dropout⁵³, batch normalization^{21;52;53}, data augmentation^{6;21;56;94;95}, feature-swapping regularization⁷⁶, and policy distillation^{79;80} are the most intuitive approaches to improve generalization. By utilizing bisimulation metrics to investigate similarities between states, researchers have been able to learn task-relevant representations⁸¹. Additionally, the generalist-specialist training framework⁸¹ and the use of language models with history compression⁹⁶ have been proposed to enable the memory to store abstractions of the observations and to facilitate generalization.

The Delayed-Critic Policy Gradient (DCPG) method was proposed to tackle the problem of the value network being more vulnerable to overfitting. This approach involves training the value function less often, but with a larger set of training data⁴⁶. As discussed in³, sharing features between the policy and value functions can lead to overfitting, thus impairing the model’s capacity to generalize. To address this issue,³ make use of fully disconnected policy and value functions, resulting in improved generalization and sample efficiency. Recent works further show that partial separation of the policy and value network can achieve competitive generalization performance while incurring less computational overhead on the majority of Procgen tasks^{26;28}. Amit et al.⁹ shows that a lower discount factor can act as a regularizer while Nafi et al.³³ shows that hyperbolically discounted advantage can help to improve generalization. In this work, we leverage discounting, however, to generate pseudo value targets.

5.6 Conclusion

In conclusion, our proposed approach AVaTar effectively generalizes the value estimation of the RL agent over unseen contexts. By utilizing augmented value function targets to incorporate the possibility of episode length variations, the AVaTar outperformed the existing standard policy gradient algorithm PPO and the state-of-the-art generalization-specific algorithm UCB-DrAC. AVaTar also has the advantage of being relatively simple and efficient, requiring no additional computing resources compared to the standard PPO. This makes AVaTar an excellent choice for real-world applications where generalization performance is critical.

Chapter 6

Policy Optimization with Horizon Regularized Advantage

6.1 Introduction

Deep neural networks have paved the way for the recent advances in machine learning and enabled powerful RL agents that can master games and real-world applications alike⁹⁷. However, neural networks are sensitive to the underlying training data distribution and hence memorize the data on which they are trained (overfitting), as their objective is to minimize the empirical prediction error⁹⁸. In RL, this manifests into agents learning training trajectories and being unable to *generalize* well, not only to unseen states but also across environments^{22;74}. Generalization in RL refers to the capability of an agent to perform well in similar but unseen environments and is currently seen as an active research challenge. Training deep RL algorithms is known to be data-intensive and given a sufficiently large set of samples, they can learn a specific skill^{11–13} but tend to overfit even with large training samples^{2;16–18}. To facilitate research on this issue, newer benchmarks have been developed under the Contextual Markov Decision Process (CMDP) framework. In CMDP, different episodes correspond to different variations of environments where the variation can be identified by a context, however, the episodes share some basic properties and high-level goals.

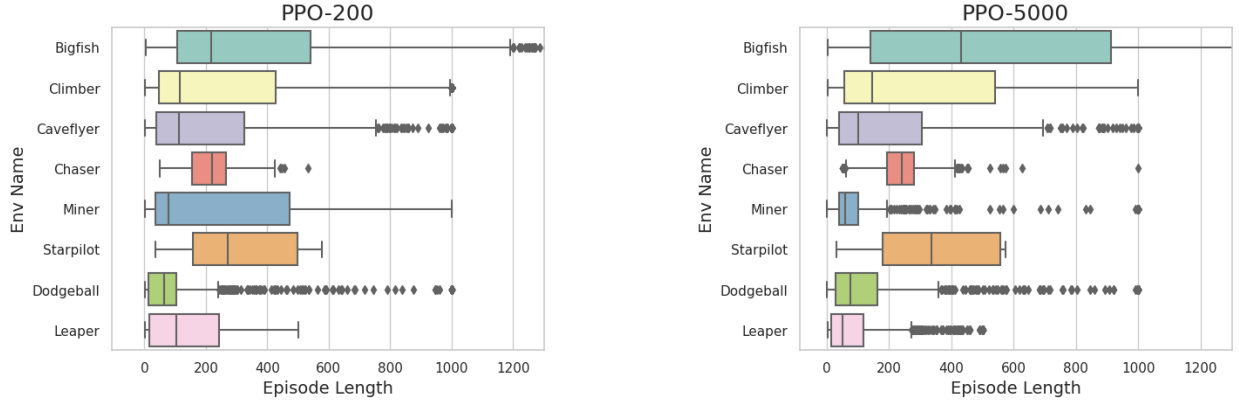


Figure 6.1: Distribution of episode lengths for a fully trained PPO agent estimated based on 1000 episodes randomly sampled from the test levels. **Left:** the agent is trained on 200 levels. **Right:** the agent is trained on 5000 levels; this denotes the inherent uncertainty and variation in completion time across levels (contexts) irrespective of the agent’s expertise.

Procedurally generated (PCG) environments such as Procgen², Minigrid⁴⁸, etc, and a set of similar robotics tasks⁹⁹ are examples of such benchmarks. CMDP enables the evaluation of generalization through held-out contexts used only in testing or generating an infinite number of contexts.

In this work, we look at generalization in CMDP from a new perspective of reward discounting that aims to avoid overfitting of the advantage estimate. RL algorithms generally specify a discount factor γ s.t. $0 \leq \gamma < 1$, that exponentially discounts the future reward r_t at step t as $\gamma^t r_t$ ⁴¹. Such exponential discounting guarantees the theoretical convergence of the value function and stabilizes optimization. The form and magnitude of the discount function itself establish strong priors over the solutions learned, and the magnitude of the discount factor sets a fixed *effective horizon* for the agent such that all rewards beyond that point are considered insignificant¹⁰⁰. Exponential discounting of future rewards is consistent with the prior belief that there exists a known, *fixed* risk or hazard rate for the agent in the environment (hazard rate is defined as the per-time-step risk the agent incurs as it acts in the environment)^{42;101}. We argue that this assumption of a fixed hazard rate (consequently, fixed episode length) in an environment does not hold in a CMDP setting.

In support of our claim, we identify that in a procedurally generated environment (an example of CMDP), where an environment’s dynamics and attribute distribution change

across levels, the generated levels are highly diverse. Different environment context indicates different degrees of uncertainty, yielding high variance in the observed episode length. To illustrate this uncertainty in the environments, in Figure 6.1 we plot the distribution of episode length measured over 1000 episodes sampled randomly from the test levels for 8 Procgen environments. The episodes were generated using a learned PPO⁷ policy trained on 25M time steps. Even for a reasonably trained agent, we see that the episode length varies significantly for each environment. Raileanu and Fergus³ show that even starting from semantically identical states, episode lengths can vary a lot due to variations in the level generation (contexts). The value estimate of a state highly depends on the length of the episode. For say, if the agent receives a reward of 10 at the end of the episode, then the short episode will have a higher expected value than the long episode due to the variable γ . Thus, a fixed effective horizon used as in the single exponential discounting may fail to better assess expected future rewards. Mandal et al.¹⁰² also describes the role of an optimal effective horizon in the presence of episode length uncertainty. Additionally, we identify that the auxiliary task of learning over multiple horizons as presented in⁴² collapses in the case of the actor-critic algorithm. This is due to the fact that in value-based algorithms the Q function directly defines the policy ($argmax$), while in actor-critic there are distinct value functions and policy functions. To address the uncertainty in the unseen environment context and achieve better generalization using an actor-critic algorithm, we propose to use a horizon-regularized advantage estimate that considers multiple discount factors (horizons). In summary, our contributions include:

- We show that CMDP indeed implies a high degree of uncertainty in the episode length or task completion time.
- We argue that a single discount factor introduces a very restricted inductive bias. To address this, we propose to mix advantage estimates from different discount factors to smooth the estimate so that it can better approximate the advantage in unseen scenarios with unknown episode lengths.

- We evaluate our approach on 3 different generalization benchmarks - Procgen, Crafter, and Minigrid; and our approach significantly outperforms PPO⁶⁸ and other baselines that use single exponential discounting.

6.2 Related Work

Generalization in Deep RL. Recent studies have highlighted the inability of RL agents to generalize to new scenarios^{54;74;75} which has led to an increasing effort on developing intelligent agents that avoid overfitting and generalize well to unseen data^{17;18;50;80}. Methods that have been used with some success include regularization techniques like dropout⁵³, batch normalization^{52–54}, and data augmentation^{6;21;56;94;95}. Raileanu and Fergus³ use decoupled policy and value networks to improve generalization while Nafi et al.^{26, 28} present the potential of partial decoupling to improve generalization. Cobbe et al.¹⁶ introduce phase-wise training of decoupled actor-critic architecture that ensures better sample efficiency and generalization. Paischer et al.⁹⁶ show that storing compact abstraction of the observation history using language model allows generalization. Zhang et al.⁷⁷ and Agarwal et al.⁷⁸ use bisimulation metrics to measure similarity between states with the aim of learning task-relevant representations. Mazoure et al.¹⁰³ propose to predict the future states by maximizing the mutual information between its internal representation of successive time steps. Bengio et al.¹⁰⁴ investigate the link between interference and generalization in temporal difference (TD) learning and suggest that TD causes low interference that leads to under-generalizing parameters. A feature-swapping regularization technique to avoid observational overfitting is proposed in⁷⁶. Generalist-specialist training framework, as introduced in⁸¹, alternates between two model training phases - one that encourages the development of general skills and another that promotes specialization in specific tasks or sub-domains. Igl et al.⁷⁹ and Lyle et al.⁸⁰ leverage policy distillation to improve generalization. Recently, the importance of balanced exploration to find a generalizable policy has been demonstrated¹⁰⁵.

Discounting. A lower discount rate has been shown to have the effect of a regularizer that can improve generalization⁹. However, determining the appropriate exponential discount factor for a particular environment remains challenging¹⁰². Nafi et al.³¹ presents a method that utilizes randomly generated discount factors to simulate augmented value targets and use them to reduce value function overfitting. Beyond the classical exponential discounting scheme, hyperbolic discounting has been studied in the fields of behavioral psychology, economics, neuroscience, and lately, to a limited extent, in reinforcement learning. Sozou¹⁰¹ proposed a per-time-step death via the hazard rate, whereas Dasgupta and Maskin¹⁰⁶ proposed that uncertainty over the timing of rewards leads to preference reversals as exhibited in hyperbolic discounting. Alexander and Brown¹⁰⁷ proposed a hyperbolic discounting-based temporal difference (TD) learning method. Although TD learning relies on exponential discounting in its calculation, naive modification to discount hyperbolically has been shown to be inconsistent. Kurth-Nelson and Redish¹⁰⁸ proposed the modeling of hyperbolic discounting via distributed exponential discounting. Fedus et al.⁴² extended this formulation to deep reinforcement learning by approximating hyperbolic discounting from exponential discounting and evaluated their approach using a value-based method, Rainbow⁸, on the Arcade Learning Environment (ALE)⁴³ benchmark. However, in this work, we propose a horizon-regularized advantage estimate that leverages multiple discount factors.

6.3 Backgrounds

6.3.1 Contextual Markov Decision Process

We consider a Contextual Markov Decision Process (CMDP) represented by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}, r, \mu_C, \mu_S)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{C}$ is the context space, $\mathcal{T}(s'|s, a)$ is the transition function, r is the reward function, μ_C is the context distribution, and μ_S is the initial state distribution that depends on the selected context. The context for an episode is selected based on the distribution $c \sim \mu_C$. Following $s_0 \sim \mu(\cdot|c)$, an initial state is sampled for the selected context (episode). After that, the successive

states for that episode given the selected context are sampled according to the distribution $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t, c)$. Let d_π^c represent the state distribution when the agent acts according to the policy π under the context c . The agent has access to a small subset of contexts during training. However, the ultimate objective is to learn a policy π that maximizes, $\mathcal{G} = \mathbb{E}_{c \sim \mu_C, s \sim d_\pi^c, a \sim \pi(s)}[r(s, a)]$, the expected return over *all possible contexts*.

6.3.2 Proximal Policy Optimization

Similar to the previous chapters, the work presented in this Chapter also builds upon the Proximal Policy Optimization (PPO) algorithm⁷, a widely adopted policy gradient method in reinforcement learning. PPO is an efficient and implementation-friendly variant of the Trust Region Policy Optimization (TRPO)⁴⁰. PPO optimizes the following objective function where the policy and value network is parameterized by θ :

$$J_{PPO}(\theta) = J_\pi(\theta) - \alpha_v L_V(\theta) + \alpha_s S_\pi(\theta) \quad (6.1)$$

where $J_\pi(\theta)$ is the policy gradient objective, $L_V(\theta)$ is the value loss, $S_\pi(\theta, \phi_\pi)$ is the entropy bonus for exploration, and α_v and α_s are the corresponding coefficients. For details, please refer to Section 2.3.1.

6.4 Methodology

In order to achieve a generalizable policy, we aim to learn a good advantage estimate to guide policy optimization. We identify that fixed exponential reward discounting results in an overfitted advantage estimate that restricts generalization. Thus, we propose to optimize the policy using a horizon regularized advantage estimate to achieve generalization.

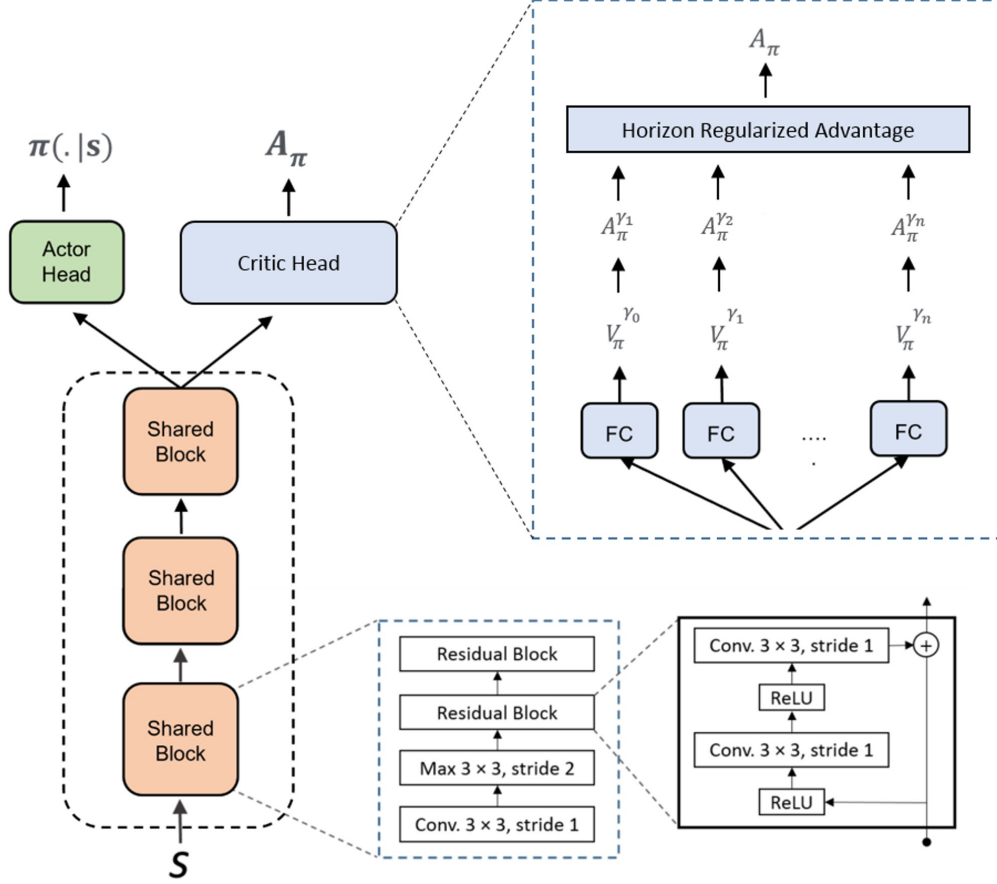


Figure 6.2: Details of our proposed architecture. Each *Shared Block* is identical to the IMPALA CNN architecture¹. The *Critic Head* predicts state-values $V_\pi^{\gamma_i}$ for n_γ different discount factors. These exponentially discounted state-value predictions are then used to calculate the corresponding advantage estimates and the horizon regularized advantage function.

6.4.1 Horizon Regularized Advantage

Generalized Advantage Estimate (GAE)⁶⁸ or simply advantage denotes the additional expected return that can be achieved by following a particular action compared to the state’s absolute value. The advantage is more resistant to overfitting to environment idiosyncrasies than value estimates and less dependent on the number of remaining steps in the episode Raileanu and Fergus³. Advantage estimates guide the policy gradient in an actor-critic setup. Thus, we focus on advantage instead of value function.

A single, fixed discount factor in the case of exponential discounting imposes an effective horizon for the agent¹⁰⁰. As a result, the agent’s value function estimate relies on a prior

belief about the length of the episode. The episode length can significantly change the value or advantage estimate of the earlier states in a trajectory based on the later reward. For example, the final reward of an episode will be highly discounted and perceived as small if the episode length is too long. However, if the episode length is small, then the same final reward will contribute much more to the advantage estimate. Thus, due to the fixed effective horizon, an agent may fail to correctly anticipate the worth of future rewards in case of highly-varied episode length (as depicted in Figure 6.1). As we can not restrict the length of an episode, we propose to relax this fixed effective horizon by considering an estimate that arises from multiple horizons. Thus, we need a value estimate that considers multiple discount factors γ_n while calculating the advantage estimate.

We introduce the notion of horizon regularized advantage (HRA) which can be considered a smoothed version of the actual advantage estimate. Formally, we define HRA as a function of multiple advantage estimates over multiple horizons. HRA prevents large policy changes corresponding to a single observed episode length that might destabilize the learning process. As the most simple versions of such HRA, we propose the average of the advantages estimated for different discount factors. If the advantage is a GAE,

$$\hat{A}_{HRA} = \frac{1}{n_\gamma} \sum_{i=0}^{n_\gamma} \hat{A}_{GAE(\gamma_i, \lambda)} \quad (6.2)$$

where $\hat{A}_{GAE(\gamma_i, \lambda)} = \sum_{l=0}^{\infty} (\gamma_i \lambda)^l \delta_{t+l}^V$ and δ_t^V is the single step TD error. We use this HRA in the policy gradient objective of PPO⁷ in Equation 2.14. Using a multi-head architecture (as shown in Figure 6.2), where each head corresponds to the value function for each γ_i , we minimize the average of the losses calculated for this multiple γ_i . The loss function corresponding to a γ_i is defined as:

$$L_v^{\gamma_i}(\theta) = \hat{\mathbb{E}}_t \left[\left(V_{\theta}^{\gamma_i}(s_t) - \hat{V}_{targ}^{\gamma_i} \right)^2 \right] \quad (6.3)$$

6.4.2 Regularization through Hyperbolic Approximation

Unseen levels in a procedurally generated environment imply an unknown hazard rate. Consider an episode sampled from a new level m has an associated hazard rate λ_m , where $\gamma_m = e^{-\lambda_m}$ ⁴². This hazard doesn't necessarily mean only the chance of dying of an agent, but the variance or uncertainty in the completion time, which is analogous to survival time or how long the agent interacts with the environment, can also be modeled through hazard rate or the corresponding discount factor. Since an agent cannot accurately estimate the hazard or put simply the value of the uncertainty parameter γ_m for each new level in a model-free setup and hyperbolic discounting is better able to capture the uncertainty⁴², thus to aid policy optimization we propose to regularize the advantage estimate through hyperbolic-discounting which is a weighted integral of the estimates obtained from multiple discount factors. Advantage is defined as $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$. Leveraging the hyperbolic function evaluation $\Gamma_k(t) = \frac{1}{1+kt} = \int_0^1 \gamma^{kt} d\gamma$ and motivated by the Q function estimation of⁴², we propose to estimate hyperbolically-discounted advantage as follows:

$$A_{HRA}^{\Gamma_k}(s, a) = \int_0^1 A_{GAE}^{(\gamma^k)}(s, a) d\gamma \quad (6.4)$$

Based on the value function calculated over all the γ^k where $0 \leq \gamma < 1$, we estimate the hyperbolically-discounted advantage. Note that the effective discount factor is γ^k , and not just the original γ . From a practical perspective, following Fedus et al.⁴², we consider a finite set of γ (consequently γ^k), say n_γ number of γ s to approximate the advantage through Riemann sum,

$$A_{HRA}^{\Gamma_k}(s, a) \approx \sum_{\gamma_i} (\gamma_{i+1} - \gamma_i) A_{GAE}^{(\gamma_i)^k}(s, a) \quad (6.5)$$

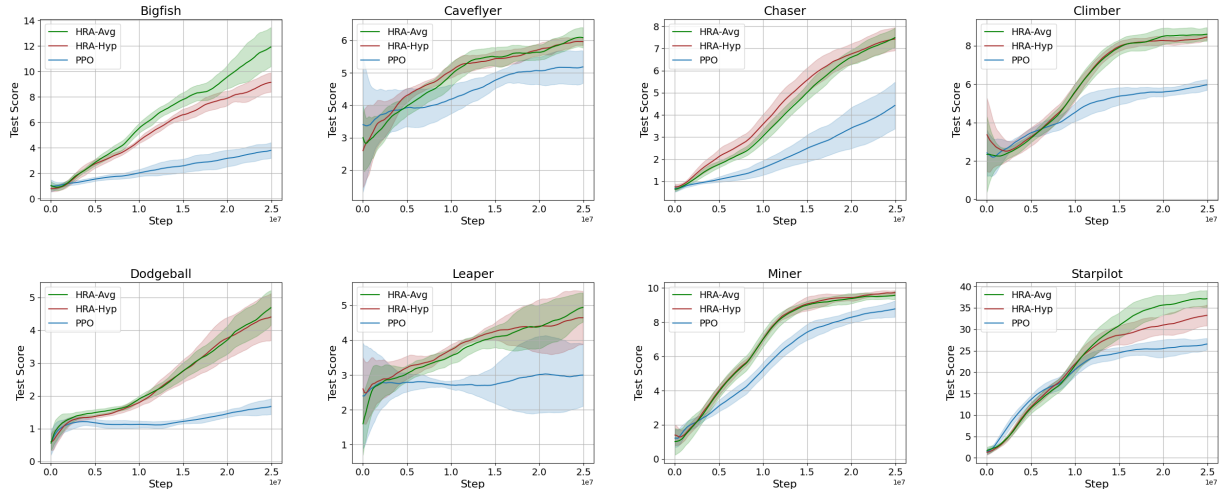


Figure 6.3: Test performance of proposed average advantage-based horizon regularization (HRA-Avg) and hyperbolic discounting-based horizon regularization (HRA-Hyp) against PPO on Procgen environments.

6.5 Experiments and Results

6.5.1 Network Architecture and Training

Following previous works, we use the IMPALA-CNN architecture as the actor-critic model for the PPO baseline which employs generalized advantage estimate². Figure 6.2 shows our implemented architecture. This CNN architecture has three identical blocks, shared by the actor and the critic, and each block has 5 convolutional layers. To implement our proposed approach HRA, we augment the same architecture with five value heads corresponding to five different γ values. We then calculate the advantage value for each of the value predictions. Finally, we take an average of all advantages for the average advantage-based regularization or integrate the advantage values to obtain the hyperbolic advantage. We use ADAM as the optimizer with a learning rate of 0.0005. For PPO, we experiment with two sets of discount factors $\gamma = [0.85, 0.90, 0.95, 0.975, 0.999]$ and $\gamma = [0.90, 0.95, 0.97, 0.98, 0.99]$. The latter demonstrates better performance and we observe using too low values of γ negatively impacts the performance. For HRA with APDAC, we use comparatively larger values for the effective discount factors with $k = 0.025$, while for PPO-based HRA we use $k = 0.1$.

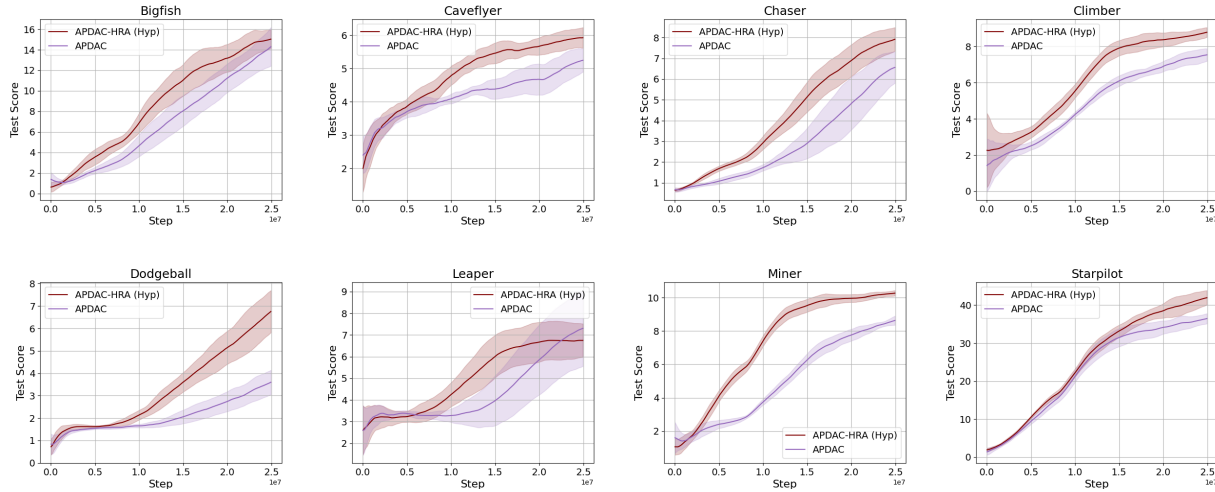


Figure 6.4: Test performance of APDAC and its counterpart that uses horizon regularized advantage through hyperbolic discounting. Means and standard deviations are calculated over 5 trials.

Our code is publicly available.¹

6.5.2 Evaluation Benchmarks

We evaluate our approach on three procedurally generated environments including Procgen², Crafter²⁴, and Minigrid⁴⁸. In all the environments, the agent aims to learn the optimal action probability based on the input image observation and obtained reward. Procgen offers an infinite number of diverse procedurally generated levels that make it suitable to investigate the generalization capability of a trained agent. We train the model for 25M time steps with the difficulty mode set to *easy*. Unless mentioned otherwise, we train the agent on 200 levels and test on the full distribution of the levels going beyond the training ones. Crafter offers a highly diverse environment with multiple achievement targets. Crafter is like an open-world survival game where each episode gets configured with a different sequence of resources, terrain types, and creatures, thus requiring generalization and long-term reasoning to perform better in any new episodes. For Minigrid, we train the agent on *Multiroom* and *Fourroom* environments. The agent needs to navigate to the randomly generated goal

¹<https://github.com/nasiknafi/horizon-regularized-advantage>

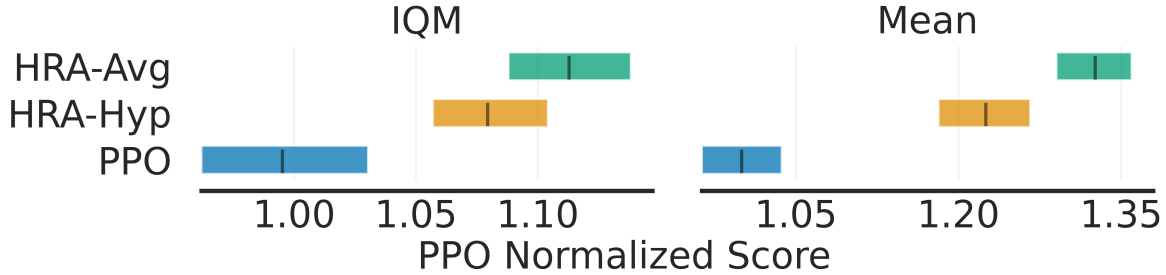


Figure 6.5: Performance comparison of proposed HRA-Avg and HRA-Hyp with PPO across all 16 environments using IQM and Mean of PPO normalized score.

location from the random initial positions. Thus, the chosen Minigrid environments can create episodes with varied lengths. We train the agent for 1M timesteps for Crafter and Minigrid.

6.5.3 Generalization Performance on Test Distribution

Figure 6.3 shows the experimental results on the test distribution of levels for 8 environments from Procgen and presents rolling mean test scores and standard deviations calculated over 5 trials. The results indicate that the proposed two horizon regularized versions of PPO, (i) that uses an average of multiple horizon advantage (HRA-Avg) and (ii) that achieves horizon regularization through hyperbolically discounted advantage (HRA-Hyp), significantly outperform the PPO baseline on the test levels. Since PPO was not specifically designed for generalization, we further incorporated our approach with APDAC²⁶, a recent generalization-specific approach, to get an understanding of the benefit of our method while integrating it with other existing solutions to generalization. Figure 6.4 shows that our proposed hyperbolic advantage-based counterpart, APDAC-HRA (Hyp), performs better than APDAC on the test distribution of most environments from Procgen. Figure 6.6 and 6.7 show that HRA achieves significantly better performance than PPO on Crafter and Minigrid benchmarks respectively.

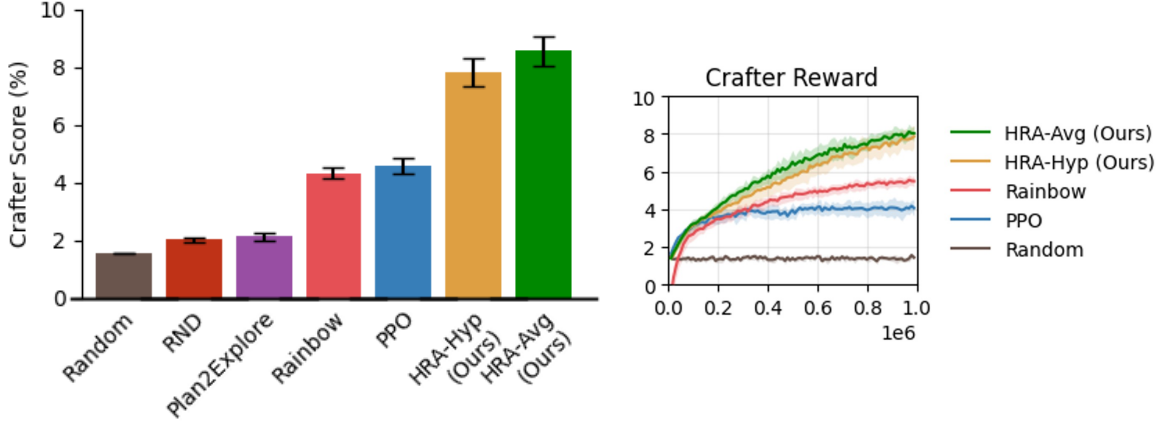


Figure 6.6: (Left) Crafter score of our HRA-Avg and HRA-Hyp compared to standard PPO and Rainbow along with other approaches; (right) Comparison of the reward achieved by each agent during 1M timestep. In terms of both metrics, both versions of our proposed approach perform significantly better.

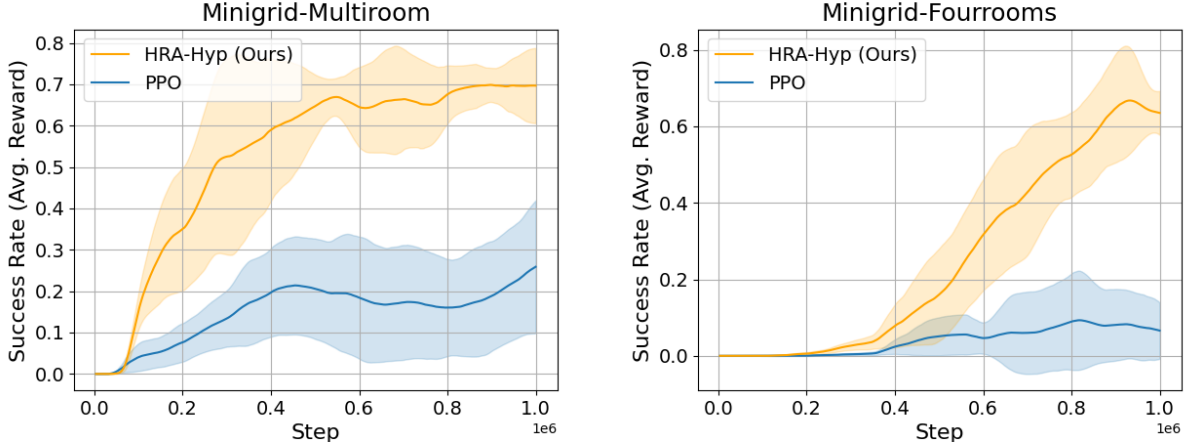


Figure 6.7: Evaluation of PPO and proposed HRA on two Minigrid environments. HRA significantly outperforms standard PPO.

6.6 Analysis and Ablations

6.6.1 Comparing Statistical Uncertainty

Figure 6.5 shows the experimental results on the full distribution of levels in terms of the aggregate metrics considered across all 16 environments as proposed by⁶⁷. Figure 6.5 indicates that both HRA-Avg and HRA-Hyp achieve higher mean PPO normalized scores. 95%

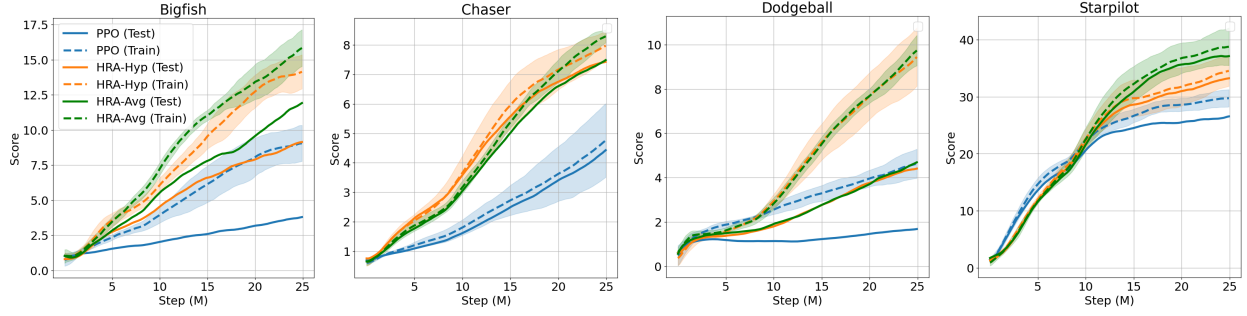


Figure 6.8: Train and test performance of PPO, HRA-Hyp, and HRA-Avg for four Procgen environments. Means and standard deviations are calculated over 5 trials, each with a different seed.

bootstrap confidence intervals do not overlap with each other. We also present Interquartile Mean (IQM) which is a more robust metric than the generic mean considering statistical uncertainty. For IQM, while the confidence intervals for the proposed two regularization methods overlap, they are far apart from the PPO.

6.6.2 Analyzing the Computational Overhead

To implement HRA, we need to add n number of different value heads for different value functions corresponding to each γ . We only add one fully connected layer for each γ . Hence, there is an increase in the number of parameters. However, we observe this introduces a very minimal difference in computational cost compared to the standard single-head version (PPO/APDAC with a single discount factor). This is mainly because the convolutional layers are all shared across different value heads. We notice an approximately 20%-25% increase in computing time for Procgen environments and a 10% increase in the case of Crafter and Minigrid. Indeed, this is marginal compared to the performance gain because generalization-specific approaches such as ^{3;16;46} require 100%-200% increased computation time compared to basic PPO.

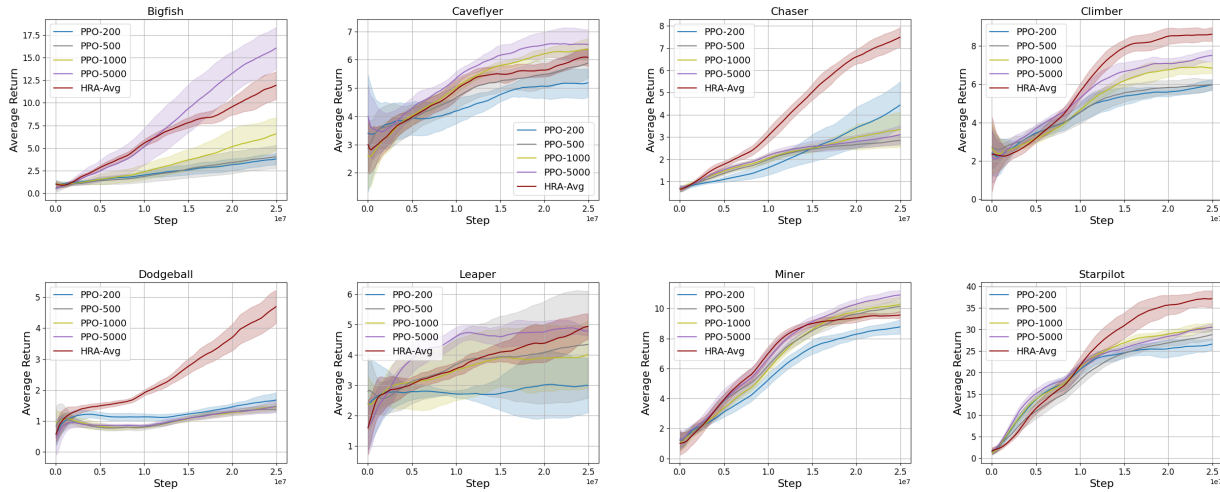


Figure 6.9: Test rewards of proposed HRA-Avg compared to traditional exponential discounting-based PPO trained on the increased number of levels. HRA-Avg has been trained on only 200 levels.

6.6.3 Assessing the Generalization Gap

We compare the train-test performance gap of the baseline PPO with the two horizon regularization methods. Figure 6.8 presents the train and test performance of all three methods across four Procgen games. HRA-Avg and HRA-Hyp outperform the PPO on both train and test level performance. It also shows that our approaches perform competitively with the baseline to reduce the train-test gap in many cases, however, there are few exceptions. Figure 6.9 shows that, in most of the environments, our proposed method HRA-Avg can achieve test rewards that are higher than the rewards of the PPO trained on more levels (more training levels inherently improve generalization). It is evident that training on only 200 levels, horizon regularization can achieve performance gain that is even higher than PPO trained on 5000 levels of Procgen.

6.6.4 Analyzing the Auxiliary Task of Multi-horizon Learning

Fedus et al.⁴² shows the potential of learning over multiple horizons, which serves as an auxiliary task. Figure 6.10 shows the test performance of our hyperbolic advantage-based approach (HRA-Hyp) against a PPO implementation that learns over multiple horizons e.g.

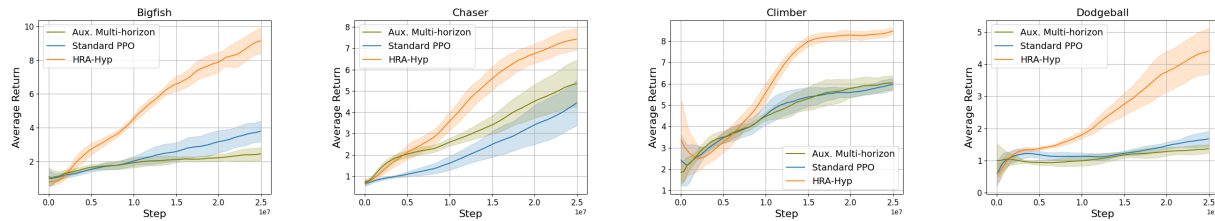


Figure 6.10: Test performance of the auxiliary task of learning over multiple horizons that calculates the advantage only with the largest gamma vs. our hyperbolic advantage estimation that combines multiple advantage estimates.

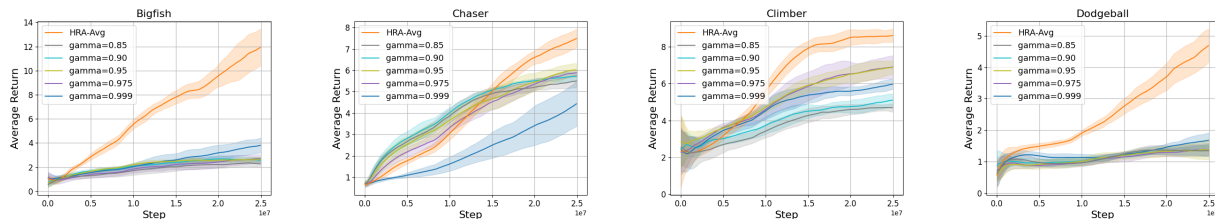


Figure 6.11: Analysis of test scores for exponential discounting using different discount factors γ vs. horizon regularization.

for five different discount factors using exponential reward discounting and it calculates the advantage estimate based on the value corresponding to the largest gamma. From the experimental results, it is evident that learning value functions over multiple horizons as an auxiliary task and estimating the advantage according to only the max gamma (largest gamma) value does not perform well. Even adding such an auxiliary task may degrade the performance. Instead, an advantage value estimated over multiple horizons (e.g. hyperbolically discounted advantage) leads to a more generalizable policy.

6.6.5 Comparison with Different Gamma

To further analyze the benefit of our proposed horizon regularized advantage estimation in the procedurally generated environments, in Figure 6.11, we present a comprehensive comparison against multiple PPO models each having a distinct single γ value. We select the gamma values such that they correspond to the gamma values that have been used collectively to approximate the advantage. It is crucial to observe from the figure that there

is no common single gamma or discount factor across the environments that perform better in general. This demonstrates that every environment has its hazard distribution and that uncertainty about the hazard cannot be modeled using a single discount factor. Our proposed HRA, leveraging the average of multiple advantage estimates from multiple discount factors, achieves consistently better rewards than any single exponential discounting-based ones. Thus our results are consistent with our claim that estimating the advantage by combining multiple horizons helps the agents to address the uncertainty.

6.7 Conclusion

This work presents an actor-critic method that uses horizon regularized advantage with the specific aim of improving the generalization ability of an agent; and evaluates it for the generalization tasks. We argue that since the task completion time in a procedurally generated environment is more uncertain, having an agent that considers a regularized advantage over multiple discount factors or effective horizon would perform better on unseen levels. Throughout the training, the agent learns the value estimate simultaneously over multiple horizons through the exponential discount factors $\gamma_0, \gamma_1, \dots, \gamma_n$, then combines the advantages resulting from those value functions. We present two schemes for horizon regularization-based advantage estimates - simple arithmetic mean of multiple GAE and hyperbolic approximation of GAE from multiple discount factors. The introduction of horizon regularized advantage enables the effective use of multiple discount factors with policy gradient (actor-critic) methods. We evaluate our proposed method using PPO and APDAC, and the results show that the modified agent performs well on most of the tasks from the Procgen benchmark. Further, our approach significantly outperforms the PPO baseline on the Crafter and Minigrid tasks. We plan on extending this work by investigating more complex discounting schemes for generalization.

Chapter 7

Conclusion

In this thesis, I have discussed how improving the fundamental aspects of RL, such as representation learning and value estimation, can help the agent in generalization. I have argued that we must look at both the architectural and algorithmic components to set proper inductive bias. In Chapter 3, I showed how high-level abstraction and structural arrangements of the policy and value networks can lead to an efficient unified actor-critic architecture. This also reveals the capability of the attention mechanism to look into relevant spatial regions and improve explainability in RL. Chapter 4 unveils the power of non-contrastive representation learning by minimally adding it to the underlying RL algorithm. I have demonstrated how instability in data augmentation-based representation learning methods can be resolved by addressing architectural redundancy and reordering computation sequences. In this dissertation, I have also explored the impact of value variance that emerged from the presence of uncertainty in episode length. Considering this prior structural knowledge in value or advantage estimation can lead to better generalization. To incorporate that information, the proposed approaches leverage the discount factor that can effectively create multiple value targets to use in value estimation. Finally, in Chapter 6, I show that a more involved method could use value estimates corresponding to multiple discount factors and an advantage estimation through the mixture of multiple advantages.

7.1 Limitations

Over the last few years, there has been a surge in the investigation of the generalization problem in RL. This resulted in empirical research of different facets of RL that impact generalization. However, limited research has been done to establish a theoretical connection, especially beyond the tabular settings¹⁴. As a mostly empirical effort, this dissertation also falls short in this regard. Thus, much research is yet required to theoretically justify the hypothesis and findings of this dissertation.

Due to the stochastic nature of RL environments and the representation of policy as a distribution over actions, RL algorithms have significant variability in agent performance during different trials. Thus, it is essential to use a sufficient number of trials and robust statistical methods to draw reliable conclusions¹⁰⁹. The statistical significance of the results helps to distinguish real improvements from random variations, ensuring that observed performance gains are not merely due to chance but rather the proposed modifications can be attributed. While this dissertation uses more robust evaluation metrics and presents the confidence interval of the results, it does so using 5 independent trials instead of using 10 seeds due to the limitation of computing resources.

The algorithmic solutions proposed in this dissertation can be used with different actor-critic methods than PPO and to some extent with value-based approaches. However, the experimental results presented are mainly based on PPO. Thus, the opportunity for experimental validation using other popular RL methods such as SAC¹³, DQN¹¹ remains open. Going beyond discrete action spaces, further experimentation with the continuous action domain¹¹⁰ may reveal important insights. In addition, validation using real-world tasks such as robotic manipulation or autonomous driving is necessary to prove their effectiveness.

The proposed approaches in this dissertation are based on some assumptions (or observed characteristics) regarding the variation between the training and testing scenarios. For example, the research in Chapter 4, assumes that observations in the testing levels in procedurally generated environments will vary in visual appearances. In chapters 5 and 6, the assumption involves the length of the episodes and considers that episode duration may

vary even while completing the same task. Thus, the proposed approaches in this dissertation may not produce intended results if those assumptions do not hold for a particular environment. Therefore, much room for improvement still exists to build a general algorithm that will work in diverse environments and achieve generalization.

7.2 Directions for Future Work

7.2.1 Neural Architecture Search for Optimal Separation

Automating the design of neural network architecture is called Neural Architecture Search (NAS)^{111;112}. NAS-based approaches explore and optimize the network architecture space and find the best-performing model for a given task. NAS can help in discovering more novel and effective architectures than manually designed architecture. This significantly reduces the overhead associated with the generic trial-and-error approach and dependency on human expertise. Architecture search spaces are generally designed using network attributes such as filters, layers, layer blocks, and hyperparameters.

In Chapter 3, we consider the four most plausible architectural variations for policy and value networks based on IMPALA¹ CNN network blocks. However, in partial separation, every layer becomes a potential point to bifurcate the network. Manually trying all the combinatorially possible options is impractical. However, exhaustively searching the possible architectural variations through NAS using weight sharing and early stopping can lead to a network that optimally separates the policy and value network.

7.2.2 Enabling Improved Contrastive Learning

As shown in Chapter 4, non-contrastive learning based on data augmentation can be leveraged to achieve better representation for handling the variation in diverse environments. However, data augmentation only captures the visual variations such as color, field of view, perspective, pattern, and position in observations. Thus, through augmentation invariance, the proposed approach only learns that these visual perturbations do not change the under-

lying state. However, the notion of which states are dissimilar is not enforced. Our research shows that simply using contrastive learning is vulnerable to representation error. Thus, alternative approaches to identifying dissimilar states will help determine negative pairs in contrastive settings. In the regime of state abstraction, two states are considered bisimilar if they lead to the same expected reward and equivalent distributions over the next bisimilar states¹¹³. Based on this, the bisimulation metric captures and measures the behavioral similarity between two states^{114;115}. Thus, deciding negative pairs using bisimulation metric-based similarity measures can be an effective approach. The *minimalist* approach used in Chapter 4 holds the potential for isolating improvements in a reductionist way, by decomposing the system to its constituent facets of behavioral similarity and allowing them to be systematically explored via ablation studies.

7.2.3 Connecting Hazardous and Contextual MDPs

Hazardous MDP (HMDP), represented by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{H})$ ⁴², is suitable for modeling decision-making tasks under uncertainty where the model is augmented with a hazard distribution $\mathcal{H}$. According to the relation between hazard rate and discount factor $\gamma = e^\lambda$, this is analogous to sampling a discount factor for each episode. Thus when the episode progresses according to the hazard-modified transition function that can be represented as $P_\gamma(s'|s, a)$. This denotes that we can consider Hazardous MDP as a collection of MDP where each MDP has its own discount factor (hazard rate) $\mathcal{M}_m = (\mathcal{S}, \mathcal{A}, P_{\gamma_m}, \mathcal{R}, \gamma_m)$. If we consider this hazard rate as a context variable that defines different MDPs, then a connection can be established between Contextual MDP (CMDP) and HMDP. Grounding such connections theoretically will open up new avenues to leverage previous solutions designed for HMDP to solve CMDP.

7.2.4 Mixture-of-Experts for Multi-horizon Advantage Estimates

Chapter 6 reveals that combining advantage estimates from different horizons (discount factors) can be a good learning signal for the policy. In this research, I follow a simple mixing strategy of the expert networks like model averaging or weighted sum that approximates

hyperbolic discounting. There are, however other ways of combining multi-horizon estimates¹¹⁶. For example, a gating network can be leveraged to learn how much each expert should contribute to the final value¹¹⁷.

7.2.5 Combining Multiple Improvements

Generalization in reinforcement learning (RL) is a complex and multi-faceted issue. Thus, the lack of generalization can be addressed through a variety of strategies¹⁴. Creating training data that closely resembles the testing data or explicitly tackling the discrepancies between training and testing environments is the key underlying principle of many existing approaches. Additionally, improvements specific to RL, such as robust optimization techniques, have proven effective in enhancing generalization performance. Common strategies to improve generalization include data augmentation, domain randomization, regularization, invariance learning, and effective exploration. Each of these solutions individually contributes to substantial performance enhancements. Since each method addresses the generalization problem from a unique perspective and under certain assumptions, they often complement each other and have the potential to be combined for significant performance improvements. Many solutions are typically implemented within a common framework like all the proposed approaches in this dissertation are developed leveraging the standard PPO algorithm. This may facilitate the integration of multiple solutions into a unified one. Thus, identifying complementary techniques that target different limitations and developing a more generalized agent that combines these techniques similar to Hessel et al.⁸ represents a promising direction for future research.

Bibliography

- [1] 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.
- [2] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.
- [3] Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 8787–8798. PMLR, 2021.
- [4] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.
- [5] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.
- [6] Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5402–5415, 2021.
- [7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

- [8] 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 *Proceedings of the AAAI conference on artificial intelligence*, volume 32 (1), 2018.
- [9] Ron Amit, Ron Meir, and Kamil Ciosek. Discount factor as a regularizer in reinforcement learning. In *International conference on machine learning*, pages 269–278. PMLR, 2020.
- [10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [11] 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.
- [12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.
- [13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [14] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.
- [15] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In *2019 International Conference on*

- Robotics and Automation (ICRA)*, pages 6008–6014, 2019. doi: 10.1109/ICRA.2019.8793735.
- [16] Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.
 - [17] Jake Grigsby and Yanjun Qi. Measuring visual generalization in continuous control from pixels. *CoRR*, abs/2010.06740, 2020. URL <https://arxiv.org/abs/2010.06740>.
 - [18] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.
 - [19] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.
 - [20] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.
 - [21] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.
 - [22] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl, 2018.
 - [23] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2): 945–990, 2022.
 - [24] Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

- [25] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.
- [26] Nasik Muhammad Nafi, Creighton Glasscock, and William Hsu. Attention-based partial decoupling of policy and value for generalization in reinforcement learning. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 15–22. IEEE, 2022.
- [27] Nasik Muhammad Nafi, Creighton Glasscock, and William Hsu. Attention-based partial decoupling of policy and value for generalization in reinforcement learning. In *Deep RL Workshop NeurIPS 2021*.
- [28] Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Analyzing the sensitivity to policy-value decoupling in deep reinforcement learning generalization. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 2625–2627, 2023.
- [29] Nasik Muhammad Nafi, Raja Farrukh Ali, and William Hsu. Analyzing the sensitivity to policy-value decoupling in deep reinforcement learning generalization. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.
- [30] Nasik Muhammad Nafi and William Hsu. Reinforcement learning with augmentation invariant representation: A non-contrastive approach. In *NeurIPS 2023 Workshop on Generalization in Planning*, 2023.
- [31] Nasik Muhammad Nafi, Giovanni Poggi-Corradini, and William Hsu. Policy optimization with augmented value targets for generalization in reinforcement learning. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2023.
- [32] Nasik Muhammad Nafi, Raja Farrukh Ali, William Hsu, Kevin Duong, and Mason Vick. Policy optimization using horizon regularized advantage to improve generalization in reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 1427–1435, 2024.

- [33] 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*, 2022.
- [34] Raja Farrukh Ali, Kevin Duong, Nasik Muhammad Nafi, and William Hsu. Multi-horizon learning in procedurally-generated environments for off-policy reinforcement learning (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 16150–16151, 2023.
- [35] Nasik Muhammad Nafi, Avishek Bose, Sarthak Khanal, Doina Caragea, and William H Hsu. Abstractive text summarization of disaster-related document. In *ISCRAM 2020 Conference Proceedings–17th International Conference on Information Systems for Crisis Response and Management*, 2020.
- [36] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.
- [37] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, Second Edition*. MIT press Cambridge, 2018.
- [38] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.
- [39] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.
- [40] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.
- [41] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT press Cambridge, 1998.

- [42] 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.
- [43] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.
- [44] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.
- [45] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.
- [46] Seungyong Moon, JunYeong Lee, and Hyun Oh Song. Rethinking value function learning for generalization in reinforcement learning. *arXiv preprint arXiv:2210.09960*, 2022.
- [47] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.
- [48] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for gymnasium. *Advances in Neural Information Processing Systems*, pages 8024–8035, 2018.
- [49] Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

- [50] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards generalization and simplicity in continuous control. *arXiv preprint arXiv:1703.02660*, 2017.
- [51] Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.
- [52] Tianyang Hu, Wenjia Wang, Cong Lin, and Guang Cheng. Regularization matters: A nonparametric perspective on overparametrized neural network. In *International Conference on Artificial Intelligence and Statistics*, pages 829–837. PMLR, 2021.
- [53] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschitschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.
- [54] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/cobbe19a.html>.
- [55] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*, 2020.
- [56] Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33:7968–7978, 2020.
- [57] Bogdan Mazouze, Ahmed M Ahmed, R Devon Hjelm, Andrey Kolobov, and Patrick MacAlpine. Cross-trajectory representation learning for zero-shot generalization in rl. In *International Conference on Learning Representations*, 2021.

- [58] Dichao Hu. An introductory survey on attention mechanisms in nlp problems. In *Proceedings of SAI Intelligent Systems Conference*, pages 432–448. Springer, 2019.
- [59] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019.
- [60] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019.
- [61] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970. PMLR, 2019.
- [62] Elaheh Barati and Xuewen Chen. An actor-critic-attention mechanism for deep reinforcement learning in multi-view environments. *arXiv preprint arXiv:1907.09466*, 2019.
- [63] Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 414–424, 2020.
- [64] Anthony Manchin, Ehsan Abbasnejad, and Anton van den Hengel. Reinforcement learning with attention that works: A self-supervised approach. In *International Conference on Neural Information Processing*, pages 223–230. Springer, 2019.
- [65] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.
- [66] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

- [67] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.
- [68] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- [69] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.
- [70] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=GY6-6sTvGaf>.
- [71] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2021.
- [72] Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann LeCun. On the duality between contrastive and non-contrastive self-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=kDEL91Dufpa>.
- [73] Ravid Shwartz-Ziv, Randall Balestriero, Kenji Kawaguchi, Tim GJ Rudner, and Yann LeCun. An information theory perspective on variance-invariance-covariance regularization. *Advances in Neural Information Processing Systems*, 36, 2024.
- [74] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

- [75] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.
- [76] David Bertoin and Emmanuel Rachelson. Local feature swapping for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2022.
- [77] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.
- [78] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021.
- [79] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.
- [80] Clare Lyle, Mark Rowland, Will Dabney, Marta Kwiatkowska, and Yarin Gal. Learning dynamics and generalization in deep reinforcement learning. In *International Conference on Machine Learning*, pages 14560–14581. PMLR, 2022.
- [81] Zhiwei Jia, Xuanlin Li, Zhan Ling, Shuang Liu, Yiran Wu, and Hao Su. Improving policy optimization with generalist-specialist learning. In *International Conference on Machine Learning*, pages 10104–10119. PMLR, 2022.
- [82] Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. Pre-trained image encoder for generalizable visual reinforcement learning. *Advances in Neural Information Processing Systems*, 35:13022–13037, 2022.

- [83] Jason Wang, Luis Perez, et al. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*, 11(2017): 1–8, 2017.
- [84] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems*, 34:29935–29948, 2021.
- [85] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- [87] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [88] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33:6256–6268, 2020.
- [89] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.
- [90] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

- [91] Guozheng Ma, Zhen Wang, Zhecheng Yuan, Xueqian Wang, Bo Yuan, and Dacheng Tao. A comprehensive survey of data augmentation in visual reinforcement learning. *arXiv preprint arXiv:2210.04561*, 2022.
- [92] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv preprint arXiv:1910.05396*, 2019.
- [93] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.
- [94] Hanping Zhang and Yuhong Guo. Generalization of reinforcement learning with policy-aware adversarial data augmentation. *arXiv preprint arXiv:2106.15587*, 2021.
- [95] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.
- [96] Fabian Paischer, Thomas Adler, Vihang Patil, Angela Bitto-Nemling, Markus Holzleitner, Sebastian Lehner, Hamid Eghbal-Zadeh, and Sepp Hochreiter. History compression via language models in reinforcement learning. In *International Conference on Machine Learning*, pages 17156–17185. PMLR, 2022.
- [97] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.
- [98] Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.
- [99] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task

- and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.
- [100] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2):209–232, 2002.
 - [101] Peter D Sozou. On hyperbolic discounting and uncertain hazard rates. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 265(1409):2015–2020, 1998.
 - [102] Debmalya Mandal, Goran Radanovic, Jiarui Gan, Adish Singla, and Rupak Majumdar. Online reinforcement learning with uncertain episode lengths. In *37th AAAI Conference on Artificial Intelligence*. AAAI, 2023.
 - [103] Bogdan Mazouze, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *Advances in Neural Information Processing Systems*, 33:3686–3698, 2020.
 - [104] Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In *International Conference on Machine Learning*, pages 767–777. PMLR, 2020.
 - [105] Yiding Jiang, J Zico Kolter, and Roberta Raileanu. On the importance of exploration for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.
 - [106] Partha Dasgupta and Eric Maskin. Uncertainty and hyperbolic discounting. *American Economic Review*, 95(4):1290–1299, 2005.
 - [107] William H Alexander and Joshua W Brown. Hyperbolically discounted temporal difference learning. *Neural computation*, 22(6):1511–1527, 2010.
 - [108] Zeb Kurth-Nelson and A David Redish. Temporal-difference reinforcement learning with distributed representations. *PLoS One*, 4(10):e7362, 2009.
 - [109] Rishab Agarwal. rliable. <https://github.com/google-research/rliable>, 2021.

- [110] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite—a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.
- [111] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.
- [112] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems*, 34(2):550–570, 2021.
- [113] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.
- [114] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.
- [115] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.
- [116] Christopher John Merz. *Classification and regression by combining models*. University of California, Irvine, 1998.
- [117] Steven Nowlan and Geoffrey E Hinton. Evaluation of adaptive mixtures of competing experts. *Advances in neural information processing systems*, 3, 1990.