Policy Optimization with Augmented Value Targets for Generalization in Reinforcement Learning

Nasik Muhammad Nafi
Department of Computer Science
Kansas State University
Manhattan, KS, USA
nnafi@ksu.edu

Giovanni Poggi-Corradini
Department of Computer Science
Kansas State University
Manhattan, KS, USA
giovannipc@ksu.edu

William Hsu
Department of Computer Science
Kansas State University
Manhattan, KS, USA
bhsu@ksu.edu

***Abstract*—Our work aims to improve the generalization performance of a reinforcement learning (RL) agent in unseen environment variations. The value function used in RL agents is frequently overfitted, leading to poor generalization performance. In this work, we argue that the task completion time is highly impacted by the varying environmental conditions, thus resulting in variation in episode lengths, and consequently, the value estimation. Therefore, learning from a limited variation of the environments, the agent gets biased to the value estimates that correspond to the observed episode lengths. To this end, we introduce Augmented Value Targets (AVaTar), which generates multiple value function targets considering the possibility of episode length variation and optimizes the value function with the average of these targets. We demonstrate that optimizing the average of the augmented targets is computationally more feasible than independently leveraging those pseudo-targets. Evaluations on the Procgen and Crafter benchmark show that our proposed approach is effective in generalizing the value estimates over unseen contexts and significantly outperforms the standard policy gradient algorithm Proximal Policy Optimization (PPO). Furthermore, comparison and integration with the recent generalization-specific approach UCB-DrAC indicate that AVaTar outperforms UCB-DrAC in most of the environments from Procgen.**

***Index Terms*—reinforcement learning, generalization, value estimation, overfitting, target augmentation, policy optimization**

I. 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 that are designed 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 the same [31]. Leveraging these CMDP-based environments (often referred to as multi-environment), several research has shown that even in the case of minor changes or perturbations to the environment RL agents fail to generalize [5], [23].

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 episode or trajectory. Thus, regularization has been shown as an effective technique not only for singleton environments but also for CMDP-based multi-environment settings [3], [12]. However, regularization suffers from premature convergence leading to suboptimal solutions. [18] shows that learning a good estimate of the value function for multi-environment is harder than for singleton environments. Due to the variation, 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 the episodes vary based on contexts and the agent needs to learn a value function (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 1 shows the distribution of episode lengths for four Procgen 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 that 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. 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

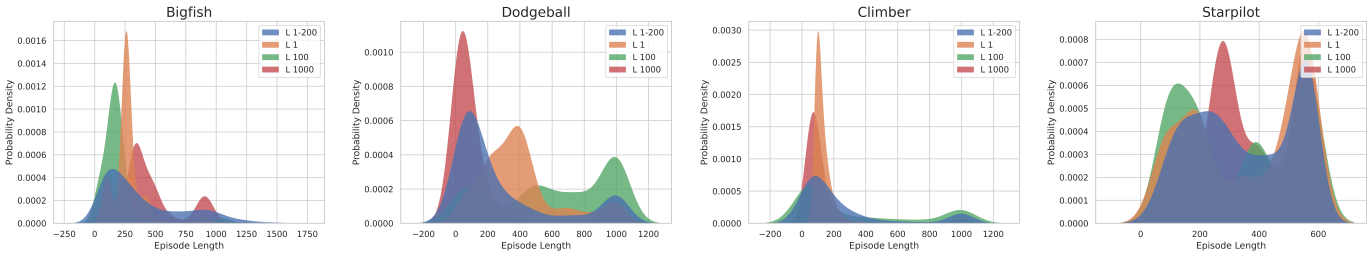

Fig. 1: Distribution of task completion time or episode lengths of four Procgen environments for a fully trained PPO agent estimated based on 1000 episodes randomly sampled from the 1 to 200 training levels vs. only level 1, level 100, and level 1000. The difference in the distribution over multiple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

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) [27]. Moreover, we show that our proposed approach achieves better performance compared to the recent generalization-specific method UCB-DrAC [24] on Procgen, and when integrated with the same method it further improves the performance.

II. Preliminaries

A. 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 shares similar characteristic 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 Procgen 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. The aim is to perform better in unseen contexts beyond the training ones, where the contexts define the variation in backgrounds, entities, and dynamics.

B. Proximal Policy Optimization

Proximal Policy Optimization (PPO) is the current standard baseline for policy gradient approaches [27]. PPO uses a shared network for policy and value function approximation. If the network is parameterized by θ, then PPO optimizes the following objective function :

$$J_{PPO}(\theta) = J_\pi(\theta) - \alpha_v L_V(\theta) + \alpha_s S_\pi(\theta) \tag{1}$$

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. PPO utilizes the Trust Region Policy Optimization (TRPO) [26] method, which is designed to maximize the following surrogate objective function: $J_\pi(\theta) = \hat{\mathbb{E}}_t\Big[r_t(\theta)\hat{A}_t\Big]$ 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. PPO shows the benefit of restricting excessively large policy updates. PPO proposes to clip the value of $r_t(\theta)$ to the intervals of $[1-\epsilon, 1+\epsilon]$ and selects the minimum between the clipped value and the original value of $r_t(\theta)$. The clipped surrogate objective function that is optimized by the PPO is as follows:

$$J_\pi(\theta) = \hat{\mathbb{E}}_t\Big[min\big(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big] \tag{2}$$

III. 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.

A. 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 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 it will contribute to changing values for earlier states in a trajectory [23]. 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 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 length of the episodes, 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.

B. 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,

$$0.9^{50} = 0.005154 = 0.9487^{100}. \tag{3}$$

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

Algorithm 1 AVaTar: Augmented Value Targets

Hyperparameters: Total number of updates N, replay buffer size T, number of epochs per update E, number of discount factors n_γ, initial network parameters θ, Gaussian mean μ, std σ, truncation range (r_l, r_u)
for $n = 1, \ldots, N$ **do**
  Collect $\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{T}$ using $\pi(\theta)$
  Sample n_γ number of discount factors from a truncated Gaussian Normal with given parameters
  Compute augmented value targets $\hat{V}_t^{\gamma_i}$ for all sampled discount factors γ_i where $i = 1, ..., n_\gamma$ and for each state s_t
  Calculate the average of the value targets $\hat{V}_t^{avg}$ using all $\hat{V}_t^{\gamma_i}$ and the advantage estimation $\hat{A}_t^{avg}$
  for $i = 1, \ldots, E$ **do**
    $L_{\mathrm{V}}(\phi) = \hat{\mathbb{E}}_t \left[\left(V_\theta(s_t, a_t) - \hat{V}_t^{avg} \right)^2 \right]$
    $J_{\mathrm{AVaTar}}(\theta) = J_\pi(\theta) + \alpha_{\mathrm{v}} L_{\mathrm{V}}(\theta) - \alpha_\pi S_\pi(\theta)$
    $\theta \leftarrow \arg\max_\theta J_{\mathrm{AVaTar}}$
  end for
end for

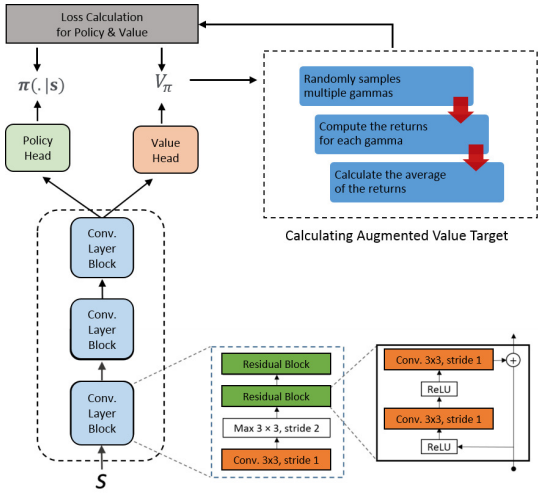

Fig. 2: Details of the architecture used and the overall approach. Each block shown on the left side is almost identical to the IMPALA CNN architecture [7]. Our calculation takes place based on the value prediction.

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.

C. Efficient Computation using Average Augmented Target

Suppose we have computed n_γ different value targets (cumulative discounted returns) for n_γ distinct discount factors, $\gamma_1, \gamma_2, ..., \gamma_{n_\gamma}$. 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:

$$L_v^{avg} = \frac{1}{n_\gamma} \sum_{i=0}^{n_\gamma} (V_i - V_s)^2 \tag{4}$$

Here, V_s is common for all the samples because we are updating against a fixed sample but with different value targets. From 4, using basic algebra it follows that,

$$\frac{1}{n_\gamma} \sum_{i=0}^{n_\gamma} (V_i - V_s)^2 > \left(\frac{\sum_{i=0}^{n_\gamma} (V_i - V_s)}{n_\gamma} \right)^2 \tag{5}$$

Further, we can rewrite the equation 5 as

$$\frac{1}{n_\gamma} \sum_{i=0}^{n_\gamma} (V_i - V_s)^2 > \left(\frac{\sum_{i=0}^{n_\gamma} V_i}{n_\gamma} - V_s \right)^2 \tag{6}$$

The right-hand side of the equation. 6 shows that if we take the average of the generated value targets $\hat{V}_t^{avg} = \frac{\sum_{i=0}^{n_\gamma} V_{\gamma_i}}{n_\gamma}$ 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 γs. 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 4, 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 1 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).

D. Network Architecture

We implement our model using large IMPALA-CNN architecture as our backbone motivated by the recent works on Procgen [4], [6], [23]. This relatively large model maintains a

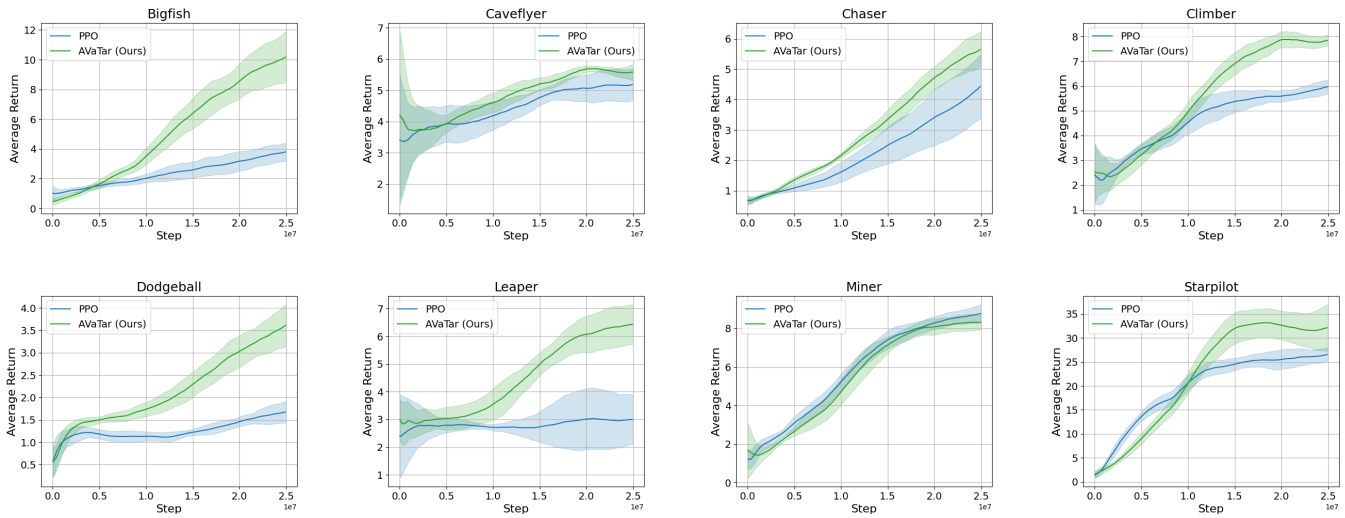

Fig. 3: Test performance our proposed Augmented Value Targets (AVaTar) and the standard PPO on Procgen environments.

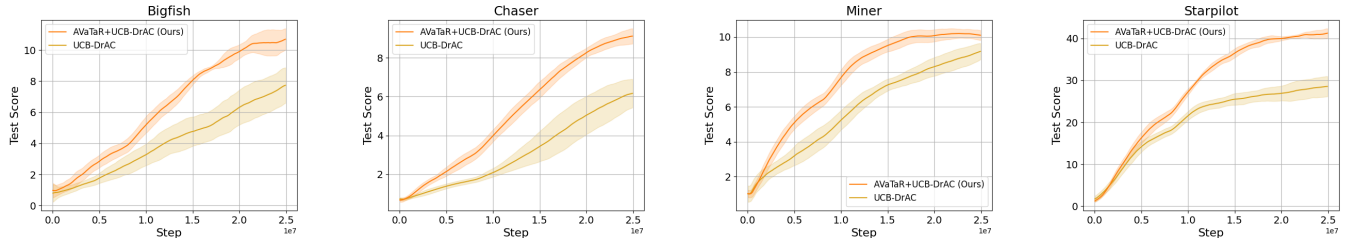

Fig. 4: Comparison of the generalization-specific SOTA approach UCB-DrAC [24] and proposed corresponding Augmented Value Targets (AVaTar) version *AVaTar + UCB-DrAC* over four Procgen environments.

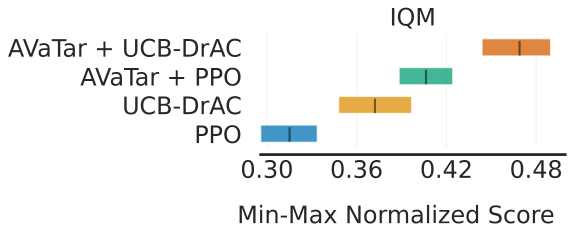

Fig. 5: Comparison of Interquartile Mean (IQM) of the Min-Max normalized score across all 16 Procgen environments. Both AVaTar versions outperform PPO [27] and UCB-DrAC [24]. The colored region shows the 95% confidence interval.

good balance between the reward achieved by learned policies in the highly diverse environment and required computational power [4]. This deeper IMPALA CNN architecture incorporates 15 convolutional layers divided into three blocks [7] containing multiple convolutional layers. Each block includes a Conv - Pooling - Residual Block - Residual Block configuration, as detailed in the bottom-middle part of Figure 2. Each residual block has two Conv layers with a ReLU activation layer. Finally, the policy head and the value head consist of fully 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 I. For the baselines, we use the optimal hyperparameters from the corresponding literature.

IV. Results and Discussions

We evaluate our approach using the two benchmarks Procgen [4] and Crafter [10]. They offer an expansive variety of procedurally generated levels, making it an ideal choice to explore an agent's generalization capabilities. Following the guideline from [4], 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

[1] https://github.com/nasiknafi/AVaTar

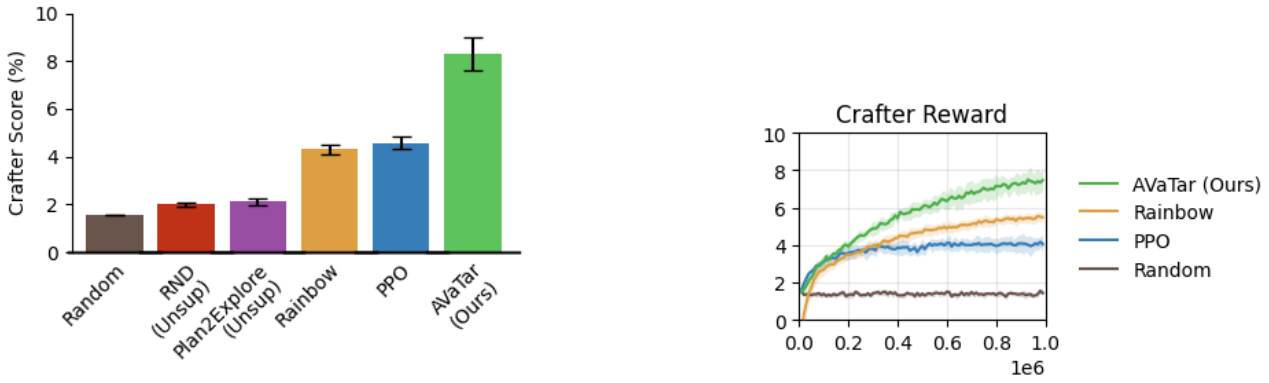

Fig. 6: (Left) Crafter score of our AVaTar compared to standard PPO [27] and popular Rainbow [11] along with other approaches; (right) Comparison of the reward achieved by each agent during 1M timestep. Proposed AVaTar outperforms other approaches considering both metrics. We use 10 different seeds for our experiments.

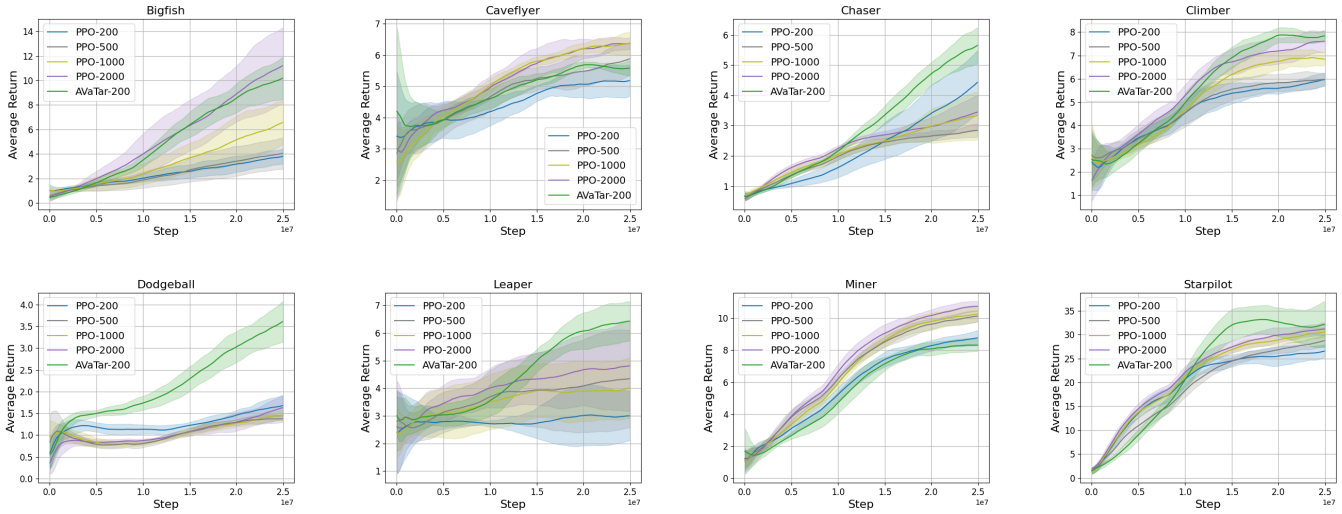

Fig. 7: Test performance of standard PPO with the increasing number of training contexts (levels) and our proposed Augmented Value Targets (AVaTar) with only 200 contexts over eight Procgen environments. It is evident that while training on 200 levels AVaTar can achieve performance gain that is even better than the PPO with 2000 training levels.

evaluation protocol [10], we trained the agent for 1M timesteps and report the Crafter score and the achieved reward.

A. Generalization Performance on Test Distribution

We first compare our proposed AVaTar with the standard and widely used policy optimization baseline PPO [27] for all the 16 environments in Procgen. Figure 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 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) [24]. UCB-DrAC dynamically determines the best 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-regularization [5]. Figure 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 as proposed by [1]. This score has been calculated considering all the 16 Procgen

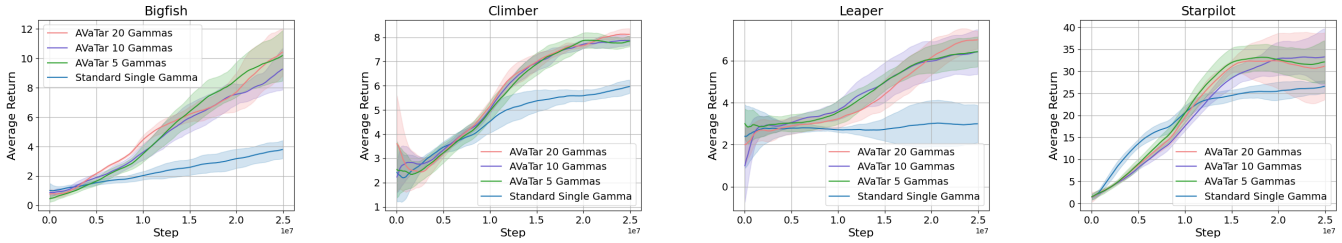

Fig. 8: Test performance of our proposed Augmented Value Targets (AVaTar) with the varying number of gammas $n_\gamma = \{5, 10, 20\}$. We observe that the performance differences with varying gamma are marginal compared to the performance gain with respect to the fixed gamma (the standard PPO).

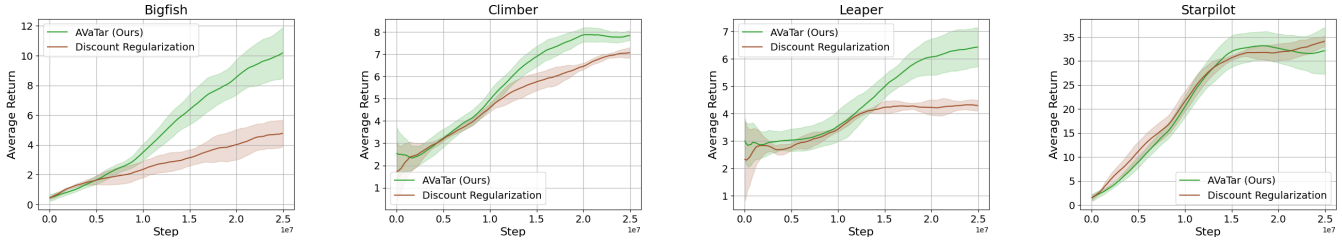

Fig. 9: Comparison of proposed Augmented Value Targets (AVaTar) with discount regularization. Discount regularization uses a fixed but slightly lower discount factor as a regularizer [2].

TABLE I: List of hyperparameters used for AVaTar trials

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

environments, thus showing the aggregate performance across the whole Procgen benchmark. The results indicate that both AVaTar versions achieve higher IQM than UCB-DrAC and PPO. [1] 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 6 shows that AvaTar outperforms both PPO [27] and Rainbow [11] 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 same achievements.

B. 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 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.

C. Additional Analysis

Figure 8 presents the results of varying numbers of discount factors. We experiment with three different values for the number of γs ($n_\gamma = \{5, 10, 20\}$). 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 9 shows the results of comparison with a discount regularization-based approach [2]. Proposed AVaTar outperforms such fixed lower discount factor (0.995 instead of standard 0.999) based regularization [2].

V. 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 [5], [8], [21]. However, the ideal and intelligent RL agents should strive to avoid overfitting and be able to generalize effectively to previously unseen data [9], [16], [17], [25].

Regularization techniques such as dropout [13], batch normalization [5], [12], [13], data augmentation [5], [24], [29]–[31], feature-swapping regularization [3], and policy distillation [14], [17] 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 representations [15]. Additionally, the generalist-specialist training framework [15] and the use of language models with history compression [22] 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 data [18]. As discussed in [23], sharing features between the policy and value functions can lead to overfitting, thus impairing the model's capacity to generalize. To address this issue, [23] 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 [20], [28]. [2] shows that a lower discount factor can act as a regularizer while [19] shows that hyperbolically discounted advantage can help to improve generalization. In this work, we leverage discounting, however, to generate pseudo value targets.

VI. 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 policy gradient-based algorithm and the state-of-the-art generalization-specific algorithm. 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.

References

[1] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in Neural Information Processing Systems*, 2021.

[2] R. Amit, R. Meir, and K. Ciosek, "Discount factor as a regularizer in reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 269–278.

[3] D. Bertoin and E. Rachelson, "Local feature swapping for generalization in reinforcement learning," in *International Conference on Learning Representations*, 2022.

[4] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 2048–2056.

[5] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1282–1289.

[6] K. W. Cobbe, J. Hilton, O. Klimov, and J. Schulman, "Phasic policy gradient," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2020–2027.

[7] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1407–1416.

[8] J. Farebrother, M. C. Machado, and M. Bowling, "Generalization and regularization in dqn," *arXiv preprint arXiv:1810.00123*, 2018.

[9] J. Grigsby and Y. Qi, "Measuring visual generalization in continuous control from pixels," *CoRR*, vol. abs/2010.06740, 2020. [Online]. Available: https://arxiv.org/abs/2010.06740

[10] D. Hafner, "Benchmarking the spectrum of agent capabilities," *arXiv preprint arXiv:2109.06780*, 2021.

[11] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[12] T. Hu, W. Wang, C. Lin, and G. Cheng, "Regularization matters: A nonparametric perspective on overparametrized neural network," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 829–837.

[13] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, and K. Hofmann, "Generalization in reinforcement learning with selective noise injection and information bottleneck," *Advances in neural information processing systems*, vol. 32, 2019.

[14] M. Igl, G. Farquhar, J. Luketina, W. Boehmer, and S. Whiteson, "The impact of non-stationarity on generalisation in deep reinforcement learning," *arXiv preprint arXiv:2006.05826*, 2020.

[15] Z. Jia, X. Li, Z. Ling, S. Liu, Y. Wu, and H. Su, "Improving policy optimization with generalist-specialist learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 10 104–10 119.

[16] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, "Illuminating generalization in deep reinforcement learning through procedural level generation," *arXiv preprint arXiv:1806.10729*, 2018.

[17] C. Lyle, M. Rowland, W. Dabney, M. Kwiatkowska, and Y. Gal, "Learning dynamics and generalization in deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 14 560–14 581.

[18] S. Moon, J. Lee, and H. O. Song, "Rethinking value function learning for generalization in reinforcement learning," *arXiv preprint arXiv:2210.09960*, 2022.

[19] N. M. Nafi, R. F. Ali, and W. Hsu, "Hyperbolically discounted advantage estimation for generalization in reinforcement learning," in *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*, 2022.

[20] N. M. Nafi, C. Glasscock, and W. 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)*, 2022, pp. 15–22.

[21] C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song, "Assessing generalization in deep reinforcement learning," *arXiv preprint arXiv:1810.12282*, 2018.

[22] F. Paischer, T. Adler, V. Patil, A. Bitto-Nemling, M. Holzleitner, S. Lehner, H. Eghbal-Zadeh, and S. Hochreiter, "History compression via language models in reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 17 156–17 185.

[23] R. Raileanu and R. Fergus, "Decoupling value and policy for generalization in reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8787–8798.

[24] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, "Automatic data augmentation for generalization in deep reinforcement learning," *arXiv preprint arXiv:2006.12862*, 2020.
[25] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade, "Towards generalization and simplicity in continuous control," *arXiv preprint arXiv:1703.02660*, 2017.
[26] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[28] N. M. Nafi, R. F. Ali, and W. Hsu, "Analyzing the sensitivity to policy-value decoupling in deep reinforcement learning generalization," in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.
[29] K. Wang, B. Kang, J. Shao, and J. Feng, "Improving generalization in reinforcement learning with mixture regularization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7968–7978, 2020.
[30] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *International Conference on Learning Representations*, 2020.
[31] H. Zhang and Y. Guo, "Generalization of reinforcement learning with policy-aware adversarial data augmentation," *arXiv preprint arXiv:2106.15587*, 2021.