Analyzing the Sensitivity to Policy-Value Decoupling in Deep Reinforcement Learning Generalization

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

Raja Farrukh Ali
Department of Computer Science
Kansas State University
Manhattan, KS, USA
rfali@ksu.edu

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

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10651435

***Abstract*—The existence of policy-value representation asymmetry negatively affects the generalization capability of traditional actor-critic architectures that use a shared representation of policy and value. To address this representation asymmetry, fully decoupled/separated networks for policy and value have been proposed, though they come with increased computational overhead. Recent research has suggested that partial separation of networks results in similar generalization performance with reduced computational costs. Thus, the questions arise: *Do we really need two separate networks? Is there any particular scenario where only full separation works? Does increasing the degree of separation in a partially separated network improve generalization?* To answer these questions, we present the first comprehensive study of the generalization performance of four different extents of decoupling of the policy and value networks, namely: fully shared, early separation, late separation, and full separation on the challenging RL generalization benchmark Procgen, a suite of 16 procedurally-generated environments, and the Crafter benchmark. Interestingly, we observe that early separation does not produce the expected generalization. Our findings suggest that, unless there is a distinct or explicit predetermined source of value estimation, partial late separation is an effective strategy for capturing necessary policy-value representation asymmetry and obtaining competitive generalization in unseen scenarios.**

***Index Terms*—reinforcement learning, generalization, actor-critic, policy-value decoupling, representation learning**

I. Introduction

Deep reinforcement learning algorithms often suffer from poor generalization performance when applying the learned policy to an unseen scenario. Raileanu and Fergus [27] 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 features [27]. Ignoring this asymmetry while learning a joint representation for 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. Thus the learned policy overfits the training instances and performs poorly in terms of generalization.

Previously proposed solutions to this asymmetry use two different networks for policy and value that capture distinct features [8], [27]. However, the use of two separate networks creates a significant bottleneck regarding computation time. Also, the dependency of policy function on the value gradients requires extra precaution in designing the network. Nafi et al. [22] propose a workaround that acts as a compromise through partial separation of the policy and value networks, and attains competitive performance compared to the fully separated counterpart while requiring less computational time. This also eliminates the need for an additional value head in the policy network as required in full separation.

We observe that while all these approaches offer a solution to the specific problem they identified, no solution is perfect across all environments considering both aspects - generalization performance and required computation time. Full separation of policy and value introduces very high computational overhead. In our experiments on a single GPU, for most of the environments, we observed 4 times higher running time for the fully separated approach, IDAAC [27], compared to the fully shared baseline PPO [6] or partially separated approach APDAC [22], as shown in Table I. On the other hand, partial separation achieves competitive performance in most environments but fails in some cases.

Thus, 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 work, 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. Our contributions can be summarized as follows:

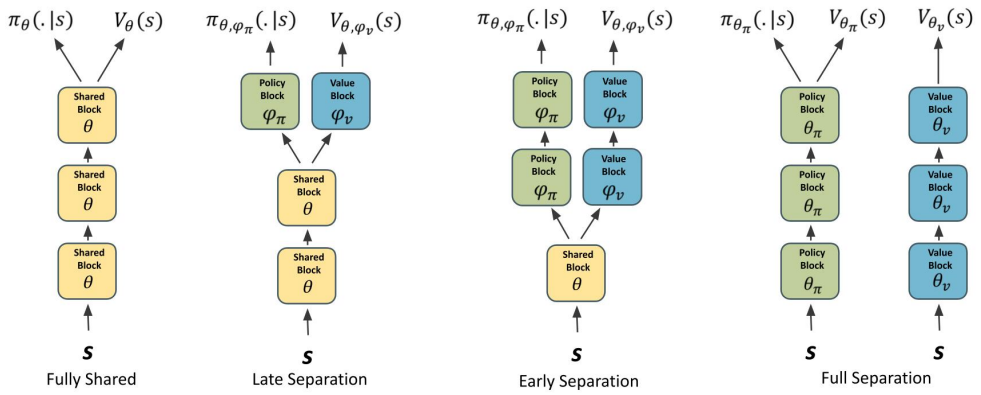

Fig. 1: Different network architectures highlighting the extent of decoupling between policy and value networks.

- We instantiate four architectures with different degrees of separation: fully shared (no separation), early separation, late separation, and full separation (see 1); 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.

II. Related Work

A lot of emphasis in deep reinforcement learning has lately been placed on an agent's ability to learn policies that are robust and generalizable [7], [10], [25]. This led to the development of intelligent agents that avoid overfitting and can generalize well to unseen data [11], [17], [29]. Some of the methods that have been proposed include regularization techniques like dropout [14], batch normalization [7], [13], network randomization [18], and data augmentation [7], [28], [34], [35]. A feature-swapping regularization technique to avoid observational overfitting is proposed in [5] whereas [15] use policy distillation to improve generalization. [1], [37] use bisimulation metrics to study similarity between states to learn task-relevant representations. [16] introduces a generalist-specialist training framework that alters between discovering general skills and achieving specialization. [26] proposes the use of language models to allow for generalization. The role of the discounting mechanism in generalization has been explored from different facets [3], [20], [21], [23]. One recent work identifies the value network as being much more prone to overfitting and proposes a Delayed-Critic Policy Gradient (DCPG) method which trains the value function less frequently with more training data compared to the policy [19].

Another aspect of research focuses on the network architecture, such as the work of [27] that uses decoupled policy and value networks to improve generalization and show that sharing policy and value functions leads to overfitting. In this context, [22] proposes the use of partially decoupled actor and critic networks that reduce the overall computation time while performing comparably to the fully decoupled architecture. [8] introduces phase-wise training using two different networks and optimizes the value function through an auxiliary phase. Several other works use decoupled architecture, however, the main objective of such works is to improve sample efficiency [4] [36]. Recently, [24] show that recurrent neural networks can be used with a decoupled architecture to perform better in POMDP settings which includes performance gain in terms of generalization. However, to the best of our knowledge, there is no work that attempts to measure the variation in performance due to different degrees of decoupling.

III. Methodology

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 1 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 architecture [9] as the base network, following the previous state-of-the-art works [8], [27], as well as the original results of Procgen benchmark [6]. 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.

A. 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 1). This is similar to traditional Proximal Policy Optimization (PPO) [32] implementations e.g. [6], [27]. Given the network parameter θ, we optimize the standard objective function used in policy gradient approaches:

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

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) [30] method, which maximizes a surrogate objective function defined as: $J_{\pi}(\theta) = \hat{\mathbb{E}}_t\Big[r_t(\theta)\hat{A}_t\Big]$ 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 $\hat{A}_t$ 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:

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

For the fully shared architecture, we optimize the same clipped surrogate objective as part of Equation 1.

B. 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 1):

$$J_{ES}(\theta, \phi_{\pi}, \phi_v) = J_{\pi}(\theta, \phi_{\pi}) - \alpha_v L_V(\theta, \phi_v) + \alpha_s S_{\pi}(\theta, \phi_{\pi}) \tag{3}$$

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.

C. 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 1):

$$J_{LS}(\theta, \phi_{\pi}, \phi_v) = J_{\pi}(\theta, \phi_{\pi}) - \alpha_v L_V(\theta, \phi_v) + \alpha_s S_{\pi}(\theta, \phi_{\pi}) \tag{4}$$

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. [22]. The difference between early separation and late separation is that the number of shared parameters represented by θ is higher in late separation.

D. 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 Fergus [27] 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 process [8]. 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:

$$J_{FS}(\theta_{\pi}) = J_{\pi}(\theta_{\pi}) - \alpha_A L_{A_{\pi}}(\theta_{\pi}) + \alpha_s S_{\pi}(\theta_{\pi}) \tag{5}$$

Here $L_{A_{\pi}}(\theta_{\pi})$ is the advantage loss coming from the additional advantage head used to support the policy network [27]. 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.

IV. Results and Discussions

We conduct our experiments on all the 16 game environments available in the demanding Procgen benchmark [6] and the single environment from Crafter benchmark [12]. 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 in [6] and [12] for Procgen and Crafter respectively. Our code is publicly available. [1] 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.

[1] https://github.com/nasiknafi/sensitivity-to-decoupling

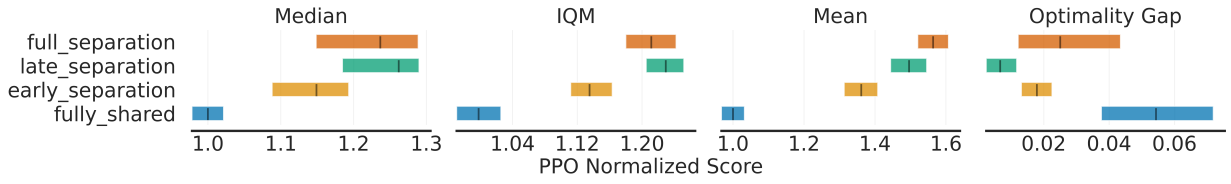

Fig. 2: PPO normalized scores for the four architecture variants across all the 16 environments in the Procgen benchmark.

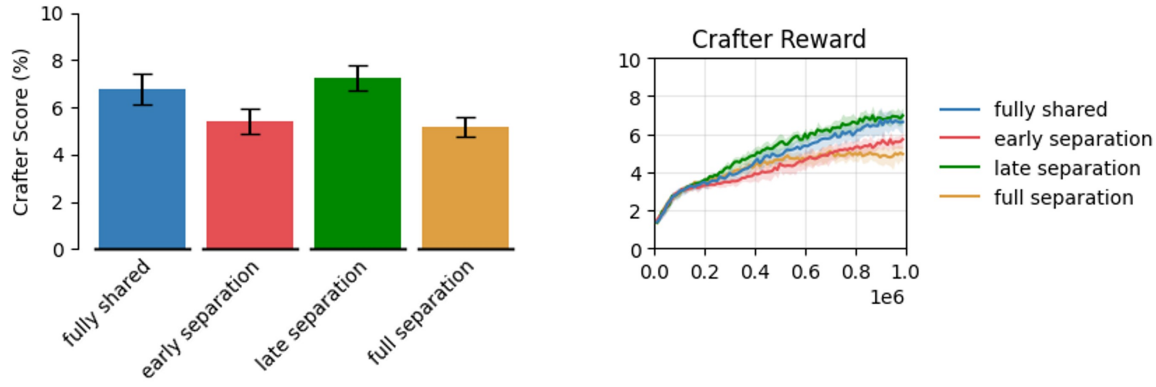

Fig. 3: (Left) Crafter score for the four architectures with different degrees of separation for policy and value; (right) Comparison of the reward achieved by each agent during 1M timestep. In terms of both metrics, *late separation* performs better.

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.

A. 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 by [2].

Figure 2 shows the evaluation metrics in terms of the PPO-normalized scores across the benchmark. As described previously in Section III-A, *fully shared* approach refers to the PPO algorithm. Thus, Figure 2 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 2 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. [2] 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 [8], [27]. We present more details related to this issue in section IV-C 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

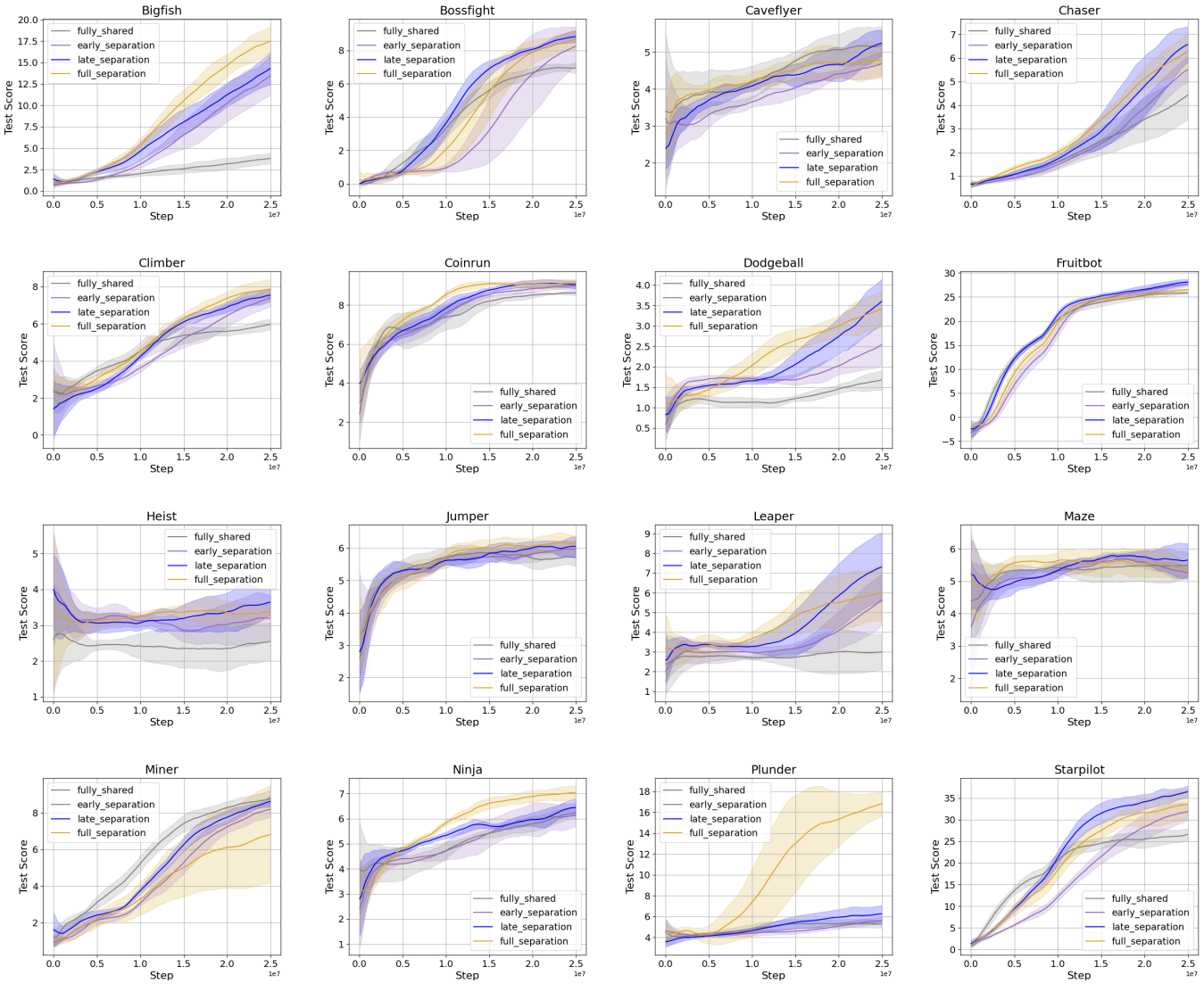

Fig. 4: Test performance for fully shared, early separated, lately separated, and fully separated architecture across all 16 environments from the Procgen benchmark. Mean and standard deviation are calculated over 5 trials with different seeds.

TABLE I: 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

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 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 I 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 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 achievements [12].

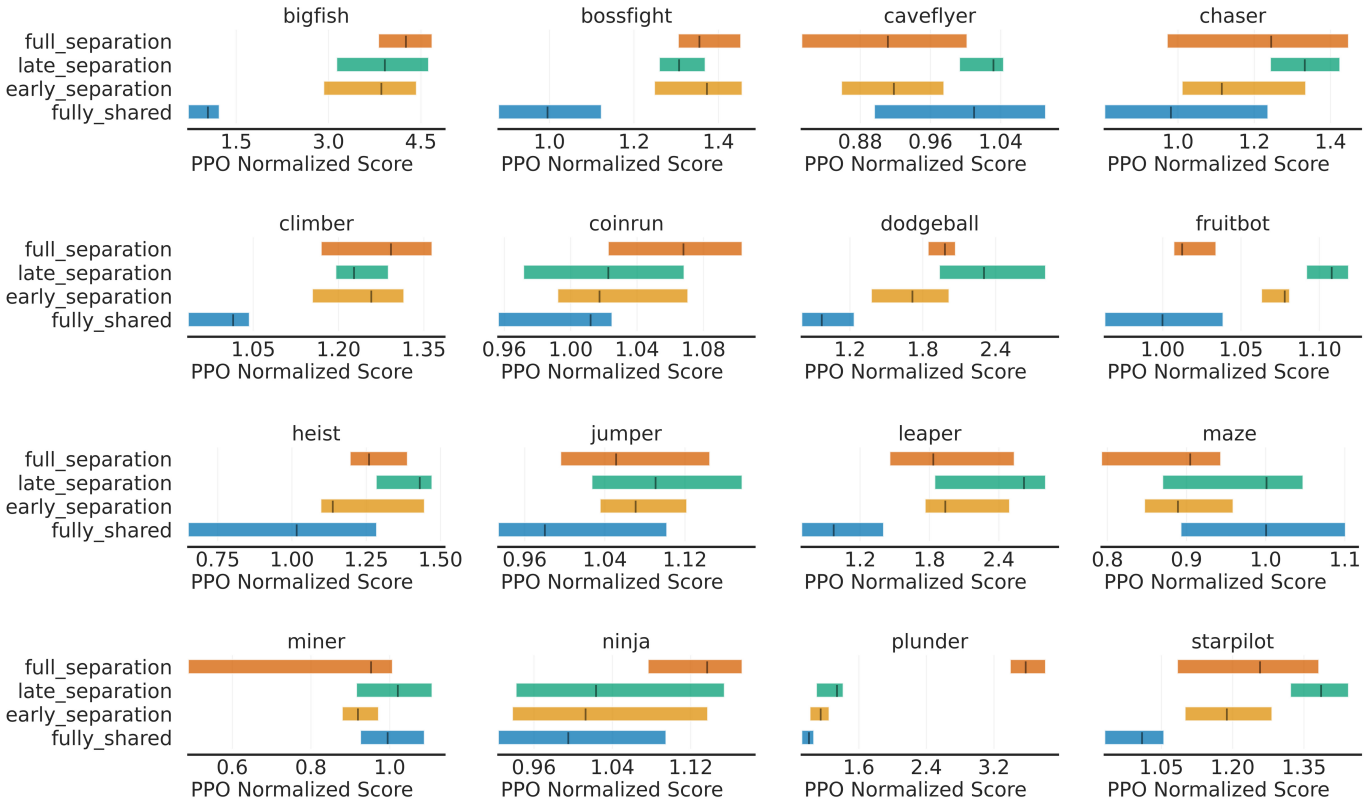

Fig. 5: Interquartile Mean (IQM) with 95% confidence interval for PPO (fully shared) normalized test rewards for all the 16 environments from the Procgen benchmark.

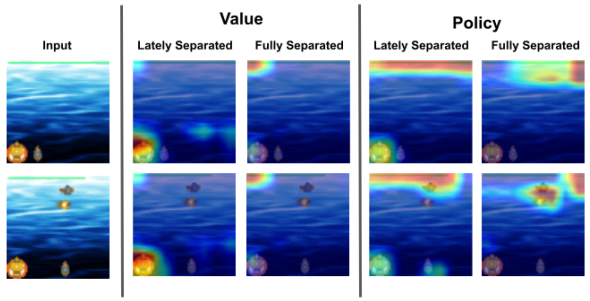

Fig. 6: Learned representation of the policy and value using lately separated and fully separated architectures for the *Plunder* environment (at two different timesteps). The red-marked regions correspond to the higher value of gradients while blue regions correspond to the lower value of gradients. The learned policy representations using lately (partially) separated 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.

B. 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 4 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 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 5.

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 6). When the timer runs out, the episode ends at that instant. Thus, this on-screen timer acts as a life bar for the agents. 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 6 presents examples of the

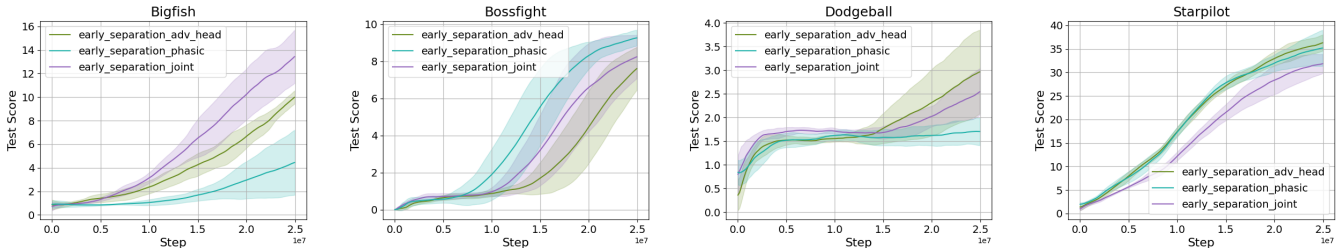

Fig. 7: Test performance of the *early separation* architecture and its two variants on four environments from the Procgen benchmark. *Early separation joint*, the baseline, uses the loss function as defined in eq. 3, *early separation phasic* uses separate optimization phases for policy and value loss, *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.

learned policy and value representation highlighted through Grad-CAM [33] 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. 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.

C. 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 7 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 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 [8], 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:

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

where $\hat{A}_t$ is the generalized advantage estimate (GAE) [31] 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 7 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*.

V. Conclusion

This work instantiates and empirically studies the possible four general categories of policy-value networks for reinforcement learning, seeking to infer meaningful insights regarding the impact of partial and full decoupling on generalization through systematic comparison. We present a comparative

analysis of the models that use different levels of decoupling/separation for the policy and value function in procedurally generated environments. Our work clearly identifies when to leverage two fully separate networks even though it might entail increased computational complexity. We identify the counterintuitive phenomenon that early separation fails even though it separates a large portion of the network. Thus, late separation acts as an efficient trade-off. Our contribution thus focuses on evaluations that will help deep reinforcement learning practitioners in deciding the type of network and the extent of decoupling to deploy to achieve better generalization when dealing with a new environment.

REFERENCES

[1] R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare, "Contrastive behavioral similarity embeddings for generalization in reinforcement learning," *arXiv preprint arXiv:2101.05265*, 2021.

[2] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in neural information processing systems*, vol. 34, pp. 29 304–29 320, 2021.

[3] R. F. Ali, K. Duong, N. M. Nafi, and W. 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, 2023, pp. 16 150–16 151.

[4] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, "What matters for on-policy deep actor-critic methods? a large-scale study," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=nIAxjsniDzg

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

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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

[12] D. Hafner, "Benchmarking the spectrum of agent capabilities," in *International Conference on Learning Representations*, 2021.

[13] 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.

[14] 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," *arXiv preprint arXiv:1910.12911*, 2019.

[15] 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.

[16] 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.

[17] 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.

[18] K. Lee, K. Lee, J. Shin, and H. Lee, "Network randomization: A simple technique for generalization in deep reinforcement learning," in *International Conference on Learning Representations*, 2019.

[19] S. Moon, J. Lee, and H. O. Song, "Rethinking value function learning for generalization in reinforcement learning," *Advances in Neural Information Processing Systems*, 2022.

[20] 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.

[21] N. M. Nafi, R. F. Ali, W. Hsu, K. Duong, and M. 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*, ser. AAMAS '24. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2024, p. 1427–1435.

[22] 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)*. IEEE, 2022, pp. 15–22.

[23] N. M. Nafi, G. Poggi-Corradini, and W. Hsu, "Policy optimization with augmented value targets for generalization in reinforcement learning," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8.

[24] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free rl can be a strong baseline for many pomdps," in *International Conference on Machine Learning*. PMLR, 2022, pp. 16 691–16 723.

[25] 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.

[26] 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 *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 17–23 Jul 2022, pp. 17 156–17 185.

[27] 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.

[28] 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.

[29] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade, "Towards generalization and simplicity in continuous control," *arXiv preprint arXiv:1703.02660*, 2017.

[30] 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.

[31] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[34] 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.

[35] 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.

[36] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 10 674–10 681.

[37] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," *arXiv preprint arXiv:2006.10742*, 2020.