Attention-based Partial Decoupling of Policy and Value for Generalization in Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this work, we introduce Attention-based Partially Decoupled Actor-Critic (APDAC), an actor-critic architecture for generalization in reinforcement learning, which partially separates the policy and the value function. To learn directly from images, traditional actor-critic architectures use a shared network to represent the policy and value function. While a shared representation for policy and value allows parameter and feature sharing, it can also lead to overfitting that catastrophically hurts generalization performance. On the other hand, two separate networks for policy and value can help to avoid overfitting and reduce the generalization gap, but at the cost of added complexity both in terms of architecture design and hyperparameter tuning. APDAC provides an intermediate tradeoff that combines the strengths of both architectures by sharing the initial part of the network and separating the later parts for policy and value. It also incorporates an attention mechanism to propagate relevant features to the separate policy and value blocks. Our empirical analysis shows that APDAC significantly outperforms the PPO baseline and achieves comparable performance with respect to the recent state-of-the-art method IDAAC on the challenging RL generalization benchmark Procgen.

1 Introduction

Deep reinforcement learning algorithms have shown human-level performance on a variety of different control tasks [Mnih et al., 2015, 2016, Haarnoja et al., 2018]. They can master complex tasks by exploring and specializing over a training task and environment given a large number of samples. Deployment of such intelligent systems in real-world applications requires significant generalization and faster adaptation capabilities with respect to similar but unseen scenarios or environments. However, generalizability of this magnitude has yet to be achieved for standard RL algorithms [Cobbe et al., 2021][Cobbe et al., 2020][Grigsby and Qi, 2020][Justesen et al., 2018].

Until recently, deep RL algorithms were trained and tested on the same environment. Thus, the issue of overfitting was not consistently observed and measured, but instead implicitly appreciated. Several recent works reveal the potential side effects of such a limited approach to evaluation in the assessment of generalization. These findings motivate the development of benchmarks that provide a better way to quantify an agent's ability to generalize [Nichol et al., 2018][Cobbe et al., 2020]. With the emergence of such benchmarks, it has become customary to train and test on different sets of similar scenarios to effectively evaluate generalization[Cobbe et al., 2021][Raileanu and Fergus, 2021].

Generalization is a fundamental aspect of representation learning in episodic tasks consisting of diverse levels. Compared to hand-designed levels, procedural content generation techniques enable generation of a nearly unlimited number of highly varied levels. In this work, we consider the problem of generalization to unseen scenarios or levels of procedurally generated environments given exposure

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

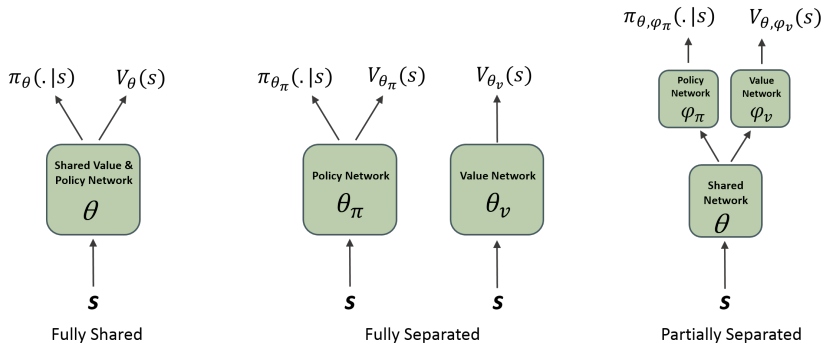

Figure 1: Comparison of architectures: (left) a fully shared network for policy and value (middle), two explicitly separate networks for policy and value (right), and our proposed partially separated network for policy and value function

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 the levels share the same end goal. Thus, significant generalization capability is needed to learn a robust policy that perform well on levels, episodes, or scenarios that have not yet been encountered at training time.

Recently, Raileanu and Fergus [2021] demonstrated a policy-value representation asymmetry, which suggests that value estimation requires more information compared to that needed to learn an optimal policy. Thus, shared representation of policy and value function can lead to overfitting [Raileanu and Fergus, 2021]. The learned representation can easily be biased towards the instance specific features responsible for accurate estimation of the value function. Consequently, the learned policy which generally requires only the minimal set of task relevant features loses its ability to generalize to unseen variations of the same task.

An alternative to the shared representation is to separate the policy and the value networks [Cobbe et al., 2021][Raileanu and Fergus, 2021]. This helps to disentangle the features necessary to properly estimate the value and policy function. However, the policy function cannot be learned in isolation, via standalone training: it requires gradients from the value function to learn the optimal policy [Raileanu and Fergus, 2021]. Thus, additional measures need to be taken to improve the policy network, which increases the complexity of overall training. Moreover, two networks entail increased memory requirements and training time.

As shown in Figure 1, we trade-off between these two extreme approaches - fully shared vs. fully separate networks - by combining the benefits of both while mitigating their disadvantages. We propose an *Attention-based Partially Decoupled Actor-Critic (APDAC)* that shares some early layer blocks of the network while separating the later (downstream) ones into policy and value subnetworks fed by the shared blocks. Additionally, we deploy an attention mechanism in the two separate branches, which further decouples policy and value function approximation. We attribute the benefits of our approach to the hierarchical representation of feature and the ability of attention mechanisms to effectively identify the components of an input pertinent to the optimization task. We also conduct an ablation study to better understand the significance of each component of our contribution.

In summary, the key contributions of this work are as follows: (i) we propose a new approach that partially shares and partially decouples the value and policy network; (ii) we develop an integrated attention mechanism to encourage distinct feature learning for policy and value with minimum overhead; (iii) we demonstrate competitive performance compared to the state-of-the-art methods on the Procgen benchmark.

2 Related Work

Many recent works have established that lack of generalization is a systemic issue in the domain of deep reinforcement learning and popular algorithms tend to overfit to the environment, resulting

in models which seem merely to memorize surface-level details of the environment rather than generalizable skills [Rajeswaran et al., 2017, Justesen et al., 2018, Grigsby and Qi, 2020, Raileanu and Rocktäschel, 2020]. Existing solutions to the generalization problem include L2 regularization [Hu et al., 2021], dropout [Igl et al., 2019a,b], data augmentation, [Cobbe et al., 2019] and batch normalization [Igl et al., 2019b]. As established in Cobbe et al. [2019], Procgen is a testing suite which uses procedural content generation to benchmark generalization to greater effect than traditional benchmarks, and became popular with recent works forwarding generalization in deep reinforcement learning [Igl et al., 2020, Wang et al., 2020, Raileanu and Fergus, 2021, Mazoure et al., 2021]. As discussed in Raileanu and Fergus [2021], sharing features between policy and value functions can lead to overfitting, reducing the model's ability to generalize to new, unseen environments. In contrast to the previous methods, Raileanu and Fergus [2021] Cobbe et al. [2021] make use of fully disconnected policy and value functions. This provides greater generalization and sample efficiency than earlier counterparts, as indicated by state-of-the-art performance on nearly all Procgen environments. However, this performance comes at the cost of a greater number of parameters than previous approches requiring more computing power. In addition, certain Procgen environments requires specific hyperparameters to produce reported performance. Our method provides results consistent with those in [Raileanu and Fergus, 2021] with fewer parameters and reducing the need for hyperparameter tuning.

Literatures exploring the potential of attention mechanisms in neural networks have 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 [Iqbal and Sha, 2019, Hu, 2019, Ramachandran et al., 2019]. Attention has been proven as an useful paradigm in the domain of natural language processing, seeing wide usage in NLP tasks such as sentiment classification, relation classification, and text summarization [Qin et al., 2017, Lei et al., 2018, Hu, 2019]. Attention has also been utilized in vision models to 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 [Fu et al., 2019, Bello et al., 2019, Ramachandran et al., 2019]. The use of attention mechanisms in the domain of deep reinforcement learning, however, is less prevalent. The closest similar works involve variations of A2C with a shared attention mechanism [Iqbal and Sha, 2019, Barati and Chen, 2019]. However, our work differs by combining the attention mechanism with a partially split policy and value function model which is designed to prevent overfitting and achieve generalization.

3 Preliminaries

While generic RL algorithms deal with a single (PO)MDP, the problem formulation of generalization consider there is a distribution of (PO)MDP which generates similar but distinct instances of (PO)MDPs and the overall goal is to perform 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. The agent is trained with a limited number of POMDPs, say n, then we have $\mathcal{M}_{train} = \{m_1, m_2, ..., m_n\}$, where $m_i \sim p$ and $i \in \{1, 2, ..., n\}$. 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 levels 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 giving the opportunity to evaluate how the model can generalize beyond the levels it encounters during training.

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 function followed by a shared component. Each of the partially separated policy and value sub-networks are enhanced with the inclusion of attention modules.

4.1 Partial Decoupling of Policy and Value function

Decoupling of the policy and the value function is crucial to overcome the problem of overfitting, which is the main drawback of a shared representation [Raileanu and Fergus, 2021]. However, simply employing two explicitly separate network has serious inherent drawbacks due to the dependency of the policy function approximation on the gradient of the value function. Cobbe et al. [2021] shows that, straightforward method of just using two separate networks for policy and value functions brings a performance decrease when compared to the shared network architecture. To address this issue, works which utilize the separate network model to optimize policy and value functions often use an auxiliary value [Cobbe et al., 2021] or advantage head [Raileanu and Fergus, 2021] in the policy network. 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 which plays the original role of the critic. Moreover, the two network models bring in additional number of hyperparameters such as the update frequency of the policy network, update frequency of the value network, coefficient 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 function may differ, the low-level features which constitute the high-level features are almost similar for both. The performance of generalization in RL increases with the number of convolutional layers. Thus, it has become customary to use very deep networks specially while learning directly from image observation [Cobbe et al., 2019] [Cobbe et al., 2020]. Deep convolutional neural networks (CNN) are capable of learning 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. So, we propose to bifurcate the network at the convolutional layer level instead of merely separating the policy and the value head as in the case of a fully shared network. This helps to decouple the high-level feature learning for policy and value on top of the same feature learned by the shared network. Thus, our network comprises three parts, the part of the network shared between policy and value parameterized by θ, the part dedicated to policy learning parameterized by ϕ_π, and another part dedicated to value function approximation which is parameterized by ϕ_v. The overall network is trained all together to optimize the following objective:

$$J_{APDAC}(\theta, \phi_\pi, \phi_v) = J_\pi(\theta, \phi_\pi) - \alpha_v L_V(\theta, \phi_v) + \alpha_s S_\pi(\theta, \phi_\pi) \tag{1}$$

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 PPO [Schulman et al., 2017]:

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

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 defined as follows:

$$L_V(\theta, \phi_v) = \hat{\mathbb{E}}_t \Big[V_{\theta,\phi_v}(s_t) - \hat{V}_t^{targ} \Big] \tag{3}$$

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

We argue that the additional overhead introduced by the reliance of the policy optimization on value gradient, increased number of hyperparameters, and higher memory footprint in case of separate network architecture can be overcome by a single network architecture which 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 into the common pitfalls of the fully shared network.

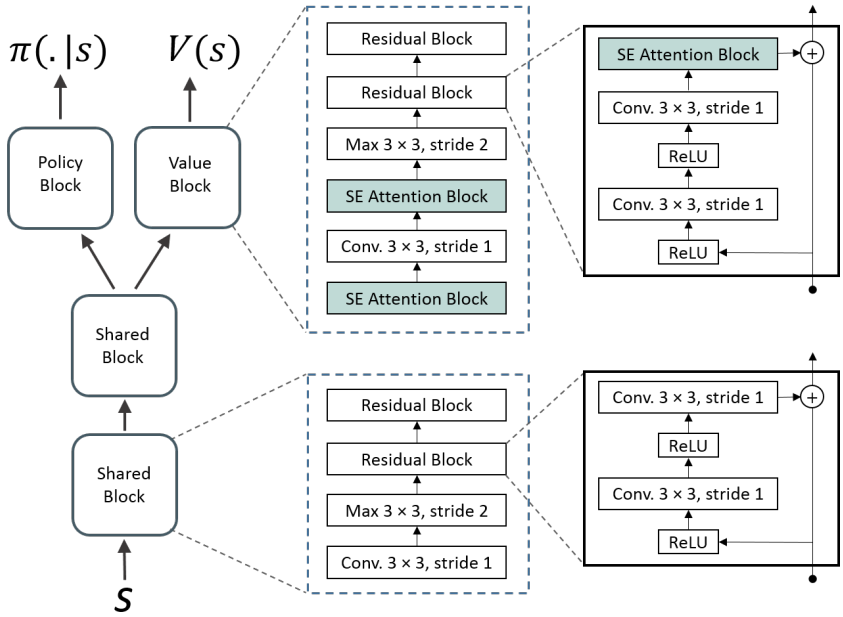

Figure 2: Details of our proposed architecture. Each block shown in the left side is almost identical to the IMPALA CNN architecture [Espeholt et al., 2018] except that the separated blocks for the policy and value function incorporate attention module both inside the residual block as well as before and after the first convolutional layer. The shared blocks do not have any attention module.

4.2 Relevant Feature Learning using Attention

To ensure maximum separation between the features learned by the partially separated policy and value sub-networks, we propose to incorporate individual attention mechanisms within the corresponding blocks of the network. Attention has been shown as an effective means to learn high-quality and meaningful representation. A good number of attention mechanisms have evolved depending on the type of feature domain they focus on. We propose to use attention modules similar to the Squeeze and Excitation (SE) network that explicitly models the inter-dependencies between the channels of its convolutional features [Hu et al., 2018]. The Squeeze and Excitation block leverages global information to put relative importance to the useful features compared to the less useful ones. The 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. This characteristic of the SE block has made it a suitable choice for our task, where we need to learn distinct features relevant to policy and value. Thus, we propose to deploy the SE block only in the split value and policy section of the network.

Our architecture incorporates the SE block in almost the same fashion as SENet for the Residual Blocks [Hu et al., 2018]; however, we add extra SE attention blocks for the convolutional layers outside of the Residual block (See Section 5.1 and Figure 2 for details). To utilize global information beyond the local receptive field of filters, the squeeze operation in the attention block first encodes a channel descriptor $z \in R^C$ by global average pooling. Each element of z is defined as:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j), \tag{4}$$

where $u_c \in \mathbb{R}^{H \times W}$ and $U = [u_1, u_2, ..., u_c]$ 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:

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z)), \tag{5}$$

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 **U** is rescaled as follows using the learned activation s:

$$\bar{x_c} = F_{scale}(u_c, s_c) = s_c u_c, \tag{6}$$

where $\bar{X} = [\bar{x_1}, \bar{x_2}, ..., \bar{x_c}]$. This way, a set of channel weights is learned to recalibrate the channel response. Thus, the attention block on the policy and value branch enables attended feature learning specific to the policy and value functions respectively. From our experiment, it is also evident that the attention mechanism coupled with the contribution of Section 4.1 helps to attain the same level of decoupling as the method with two separate networks.

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 in Cobbe et al. [2020], which consists of 16 distinct environments with procedurally generated levels. Procgen provides a difficulty setting to tune the difficulty of an environment's level generation that can be set to either "easy" or "hard". We experiment on the easy difficulty setting for 25 million total timesteps as per the recommendation of Cobbe et al. [2020]. 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 observation space is of the size 64 × 64 × 3 where the last dimension represents the standard RGB channel. The agent is required to learn the optimal policy directly from these image observations.

5.1 Network Architecture

Following previous works involving Procgen, we chose IMPALA's deeper residual CNN architecture as our backbone [Cobbe et al., 2020][Cobbe et al., 2021][Raileanu and Fergus, 2021]. Although this is a relatively large model, it strikes a good balance between the reward achieved by learned policies in the highly diverse environment and the required computational power [Cobbe et al., 2020]. This particular IMPALA CNN architecture has 15 convolutional layers divided into three blocks [Espeholt et al., 2018]. Each block has a similar configuration like Conv - Pooling - Residual Block - Residual Block as shown in the bottom-middle part of the Figure 2. Each residual block includes two Conv layers with 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 function 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 first convolutional block. Figure 2 shows the details of the network architecture.

We extend the implementation of IDAAC [Raileanu and Fergus, 2021] to experiment with our APDAC architecture. We also used the same PPO code used in Raileanu and Fergus [2021] which is actually built using the PyTorch implementation of Kostrikov [2018]. When training PPO and IDAAC, whenever applicable, we followed the same hyperparameter setup from Raileanu and Fergus [2021]. The only difference is that we reduced the number of mini batch sizes to minimize computational cost. Reported IDAAC results are produced using the best hyperparameters for each individual Procgen environment as established in Raileanu and Fergus [2021].

5.2 Generalization Performance on Test Distribution

In our experiments, we compare the performance of our network architecture, APDAC, with that of representatives of two baseline networks. PPO serves as a representative of the models with fully shared policy and value networks, whereas IDAAC represents those having separate policy and value networks[Schulman et al., 2017][Raileanu and Fergus, 2021]. In addition to decoupling the optimization of the policy and value function using two fully separate networks, IDAAC also uses a discriminator loss function to decrease the dependency on the instance specific aspects of the

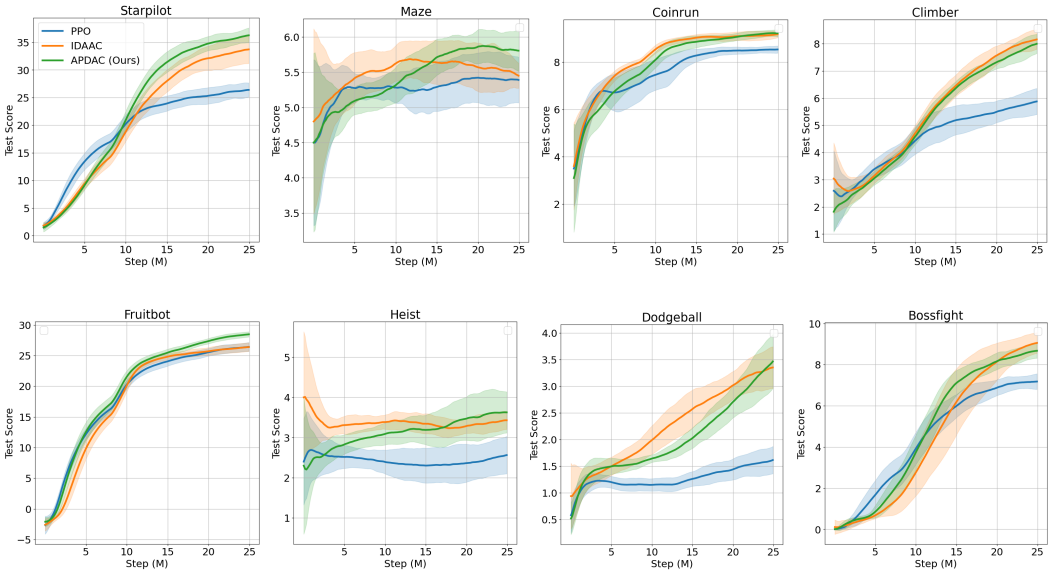

Figure 3: Test performance of PPO, IDAAC, and APDAC over eight Procgen environments. Mean and standard deviation are calculated over 10 trials, each with a different seed.

environment which are irrelevant to learning good policy. Figure 3 shows the rolling mean test score averaged over ten trials for each of the eight environments from the Procgen benchmark. The rolling standard deviation between these trials is calculated as well, with confidence intervals bounding one standard deviation above and below each curve. In these results, APDAC attains significant gains in the performance compared to the shared network approach shown as PPO. Furthermore, it performs competitively, and in some cases better than, the existing state-of-the-art, IDAAC, and does so with fewer required parameters. In our particular instanciation, While two separate network requires 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. As such, we conclude that APDAC succeeds as an efficient functional compromise between the two existing topologies.

5.3 Assessing the Generalization Gap

In addition to analyzing the performance on the test distribution, it is important to investigate the performance gap between train and test tasks or levels. This will provide an idea how good the model is performing on the unseen tasks compared to ones that belong to the training set. It is obvious from Figure 4 that APDAC is capable of reducing the train-test gap at the same level of IDAAC.

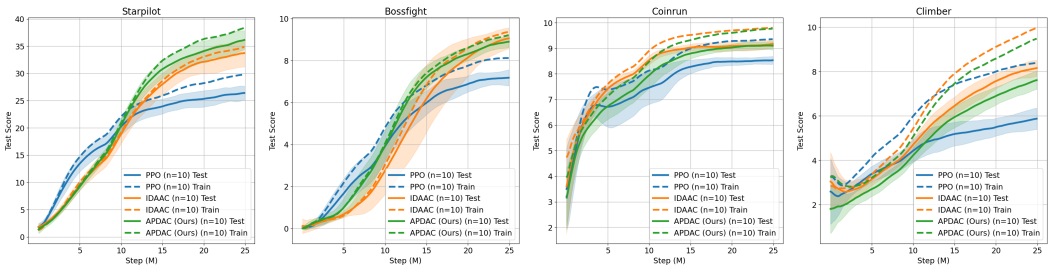

Figure 4: Train and test performance of PPO, IDAAC, and APDAC over four Procgen environments. Mean and standard deviation are calculated over 10 trials, each with a different seed.

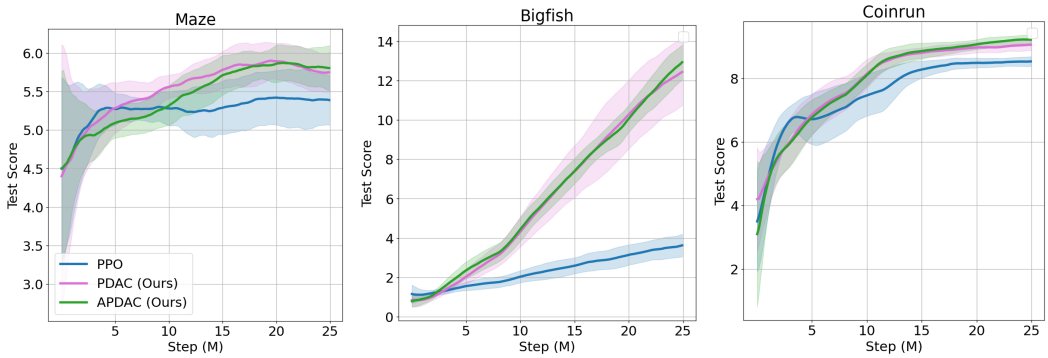

Figure 5: Ablation study for PPO, PDAC, and APDAC on the test distribution for three Procgen environments. Mean and standard deviation are calculated over 10 trials, each with a different seed.

5.4 Ablation

To evaluate the contribution 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 representation and do not incorporate the contribution mentioned in Section 4.2. We denote this model as Partially Decoupled Actor-Critic (PDAC). To determine the difference in performance brought by this ablation, we compare the results achieved by PPO, APDAC, and the ablation, PDAC using the same experimental setup as before. Figure 5 shows the ablation results from three example environments, in which we see APDAC performing better, to an extent, than PDAC. Indeed, it is clear from the comparison with PPO that the main performance gain of APDAC comes from the partial separation of policy and value network. As such, sharing the initial part of the network does not harm performance, and in fact reduces the number of parameters as compared to a network architecture consisting of separate value and policy networks. Also, it is evident from Figure 5 that the use of attention mechanism helps to reduce variance.

6 Conclusion

In this work, we discuss a current systemic issue in deep RL regarding lack of performance improvement via generalization. There is an asymmetry in the information required to train the policy and value functions. Thus, fully shared policy and value networks are prone to overfitting. However, the policy approximation requires gradients from the value function to learn the optimal policy, so a well-performing model cannot completely isolate the two. Our solution, APDAC, addresses these issues in an efficient way by partially decoupling the policy and value networks while also adding attention mechanisms to each sub-network in order to efficiently identify relevant important features. Our approach differs from current methods in the partial separation of its networks and keeping the number of convolutional layers lower than its fully separate counterpart. Our results demonstrate similar benefits of a fully decoupled approach while reducing the overall parameters and computational cost. Our Procgen results show that such a compromise is a promising way forward in the pursuit of generalization in deep RL on the grounds of both performance and efficiency. The limited marginal performance gains between APDAC and the ablation are an incremental aspect of this work; however, attention as it relates to the field of deep RL is still a growing field of study, and our work demonstrates an opportunity for the use of attention to achieve generalization. A hopeful future direction is to investigate more beneficial structures for this attention mechanism.

References

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.

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.

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.

Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.

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.

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`.

Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl, 2018.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. *arXiv preprint arXiv:2102.10330*, 2021.

Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards generalization and simplicity in continuous control. *arXiv preprint arXiv:1703.02660*, 2017.

Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

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.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *arXiv preprint arXiv:1910.12911*, 2019a.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *arXiv preprint arXiv:1910.12911*, 2019b.

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`.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv e-prints*, pages arXiv–2006, 2020.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *arXiv preprint arXiv:2010.10814*, 2020.

Bogdan Mazoure, Ahmed M Ahmed, Patrick MacAlpine, R Devon Hjelm, and Andrey Kolobov. Cross-trajectory representation learning for zero-shot generalization in rl. *arXiv preprint arXiv:2106.02193*, 2021.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970. PMLR, 2019.

Dichao Hu. An introductory survey on attention mechanisms in nlp problems. In *Proceedings of SAI Intelligent Systems Conference*, pages 432–448. Springer, 2019.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

Pengda Qin, Weiran Xu, and Jun Guo. Designing an adaptive attention mechanism for relation classification. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4356–4362. IEEE, 2017.

Zeyang Lei, Yujiu Yang, Min Yang, and Yi Liu. A multi-sentiment-resource enhanced attention network for sentiment classification. *arXiv preprint arXiv:1807.04990*, 2018.

Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

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.

Elaheh Barati and Xuewen Chen. An actor-critic-attention mechanism for deep reinforcement learning in multi-view environments. *arXiv preprint arXiv:1907.09466*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

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.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. *GitHub repository*, 2018.

A Hyperparameters

We conduct a hyperparameter search over the number of epochs per rollout $E \in [1, 3, 6]$ and found $E = 1$ works best with APDAC. The other hyperparameters are listed in the Table 1.

Table 1: List of hyperparameters used for APDAC trials

Hyperparameter	Values
# timesteps per rollout	256
# epochs per rollout	1
# minibatches per epoch	16
entropy bonus	0.01
clip range	0.2
learning rate	5e-4
γ	0.999
λ	0.95
total timesteps	25M
optimizer	ADAM
reduction ratio, r	8

B Results for All Procgen Environments

Figures 6 and 7 show the test and train performance for APDAC, IDAAC, and PPO on all the 16 games available in Procgen benchmark. Considering the test results, our proposed approach outperforms the shared representation-based baseline PPO by a significant margin on the majority of the games. Comparing the performance with the fully separate network-based approach IDAAC, we can see it achieves highly comparable performance on all games except one (Plunder). Another issue to note is that for two environments - leaper and miner - the IDAAC model performs lower than that reported in the original paper. We suspect that the reduced mini-batch batch size may be a probable reason behind this issue as the only hyperparameter that has been changed from the suggested hyperparameters for IDAAC is the mini-batch size.

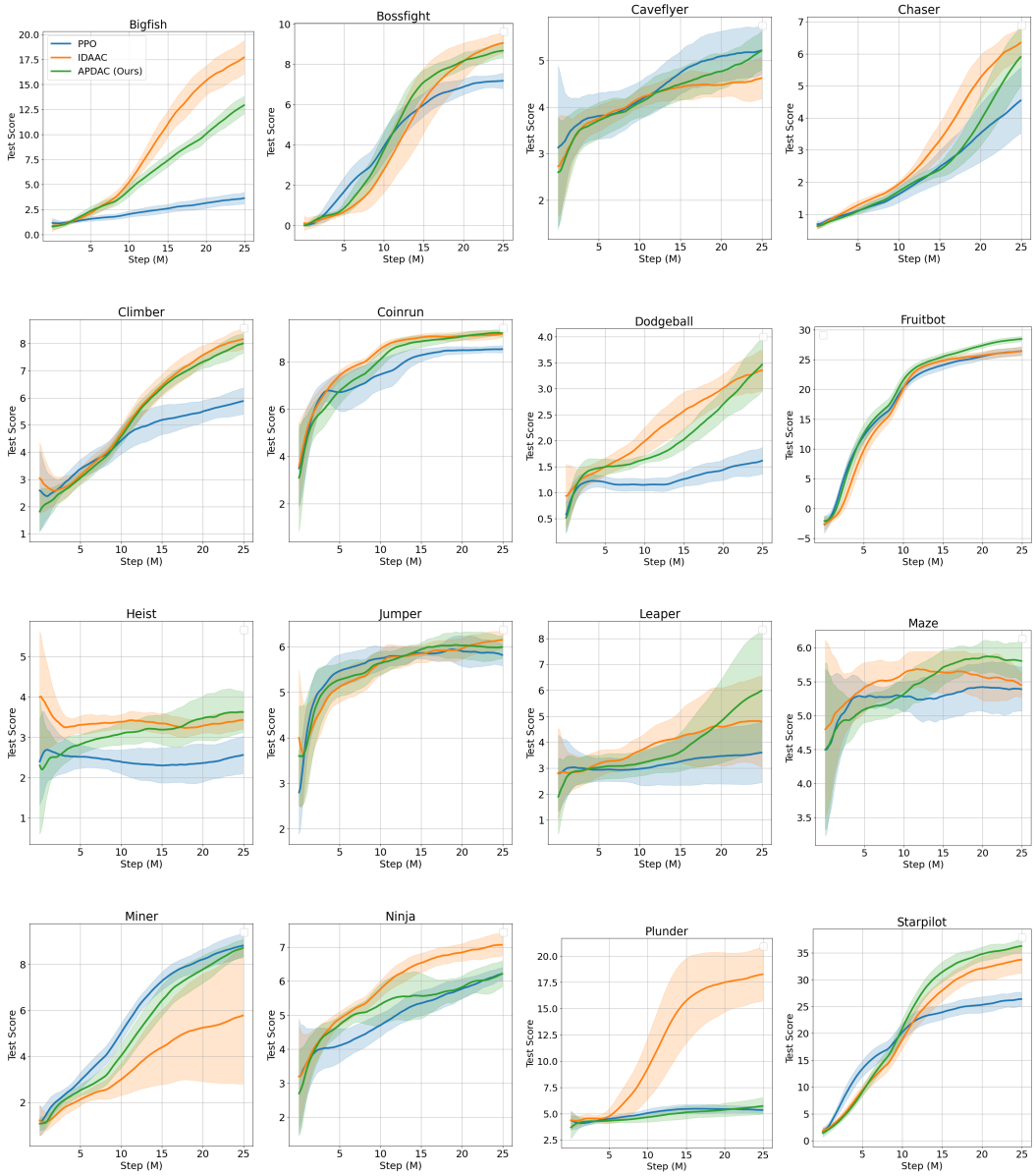

Figure 6: Testing performance for PPO, IDAAC, and APDAC across the entire Procgen benchmark. Mean and standard deviation are calculated over 10 trials, each with a different seed.

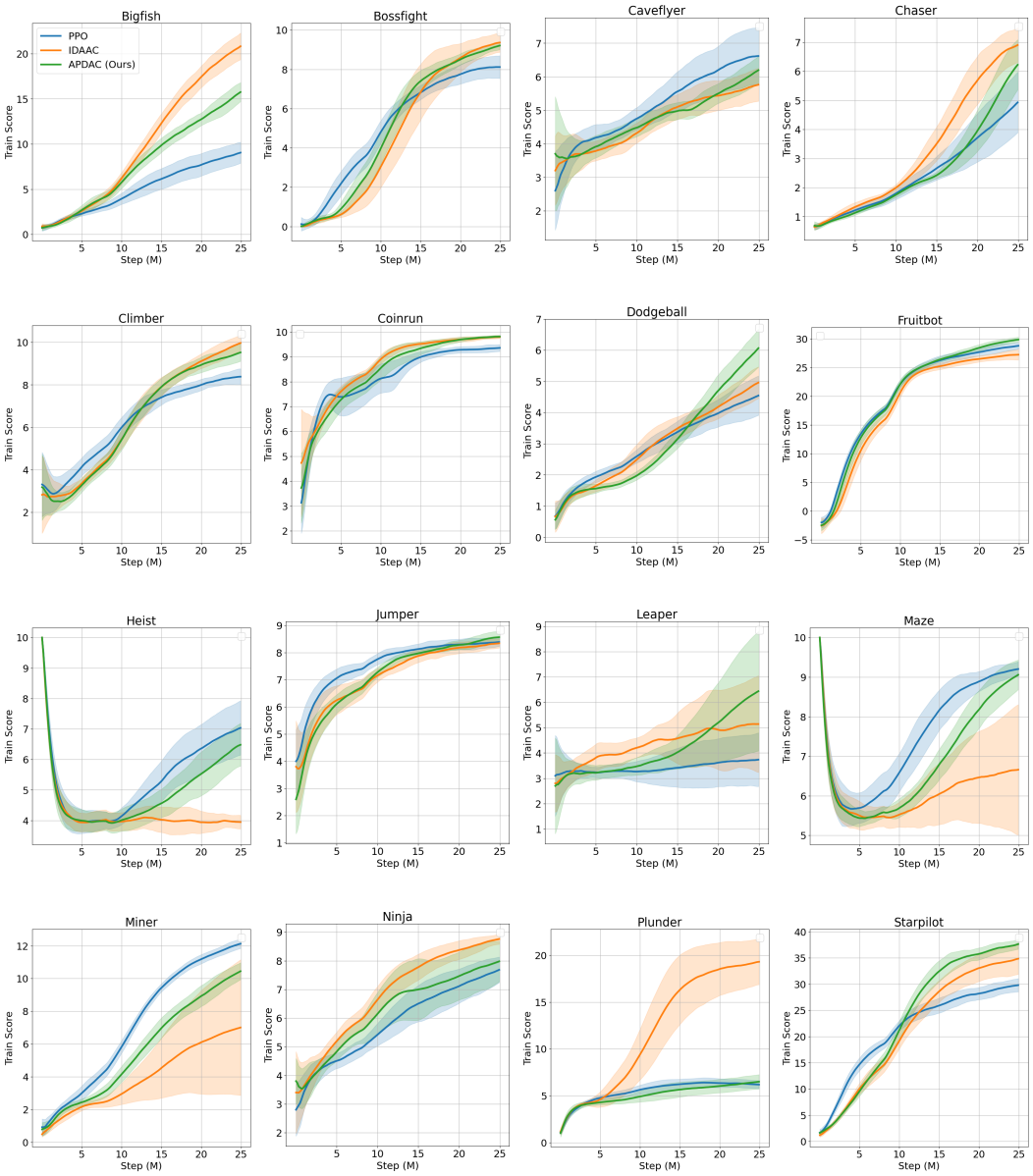

Figure 7: Training performance for PPO, IDAAC, and APDAC across the entire Procgen benchmark. Mean and standard deviation are calculated over 10 trials, each with a different seed.