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

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

Creighton Glasscock
Department of Computer Science
Kansas State University
Manhattan, KS, USA
creightong@ksu.edu

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

***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 functions. To learn directly from images, traditional actor-critic architectures use a shared network to represent the policy and value functions. While a shared representation allows parameter and feature sharing, it can also lead to overfitting that catastrophically damages 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 computation time. APDAC is a hybrid architecture that builds upon the combined strengths of both architectures by sharing initial layer blocks of the network and separating the later ones for policy and value. APDAC incorporates an attention mechanism to enable robust representation learning. We present meaningful visualization of the policy and value that explains the perception of the trained agent. Our empirical analysis, including an ablation study, shows that APDAC significantly outperforms the standard PPO baseline on the challenging RL generalization benchmark Procgen and achieves performance that is competitive with the recent state-of-the-art method (IDAAC) while using fewer convolutional layers and requiring less computational time. Our code is available at https://github.com/nasiknafi/apdac.**

***Index Terms*—deep reinforcement learning, generalization, policy-value asymmetry, partial separation, attention, representation learning**

I. Introduction

Deep reinforcement learning (RL) algorithms have shown human-level performance on a variety of control tasks [1]–[3]. They can master complex tasks by exploring and specializing in a training environment given a large number of samples. Deploying such intelligent systems in real-world applications requires significant generalization and faster adaptation capabilities for similar but unseen scenarios or environments. However, generalizability of this magnitude has yet to be achieved for standard RL algorithms [4]–[7].

In this work, we consider the problem of generalization to unseen scenarios or levels of procedurally generated environments given exposure to a limited number of levels during training. The levels vary in background, dynamics, game assets, and attributes of the entities (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 performs well on levels, episodes, or scenarios not encountered during training. Learning a policy representation that encodes task-relevant attributes is critical to better generalization in such cases.

Recently, [8] demonstrated a policy-value representation asymmetry, which suggests that value estimation requires more information than what is needed to learn an optimal policy. Thus, shared representation of policy and value function can lead to overfitting [8]. An alternative to shared representation is to separate the policy and the value networks [5], [8]. 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 stand-alone training. It requires gradients from the value function to learn the optimal policy [8]. Thus, additional measures must be taken to improve the policy network, which increases the complexity of overall training. Moreover, the two networks entail increased memory requirements and training time.

As shown in Figure 1, we design a hybrid actor-critic architecture that provides an excellent alternative to the two extreme approaches, fully shared and fully separate networks, by combining the benefits of both while mitigating their disadvantages. We propose *Attention-based Partially Decoupled Actor-Critic (APDAC)* that shares some early layers of the network while separating the later (downstream) ones into policy and value sub-networks fed by the shared blocks. This partial decoupling recognizes the asymmetry in the policy-value representation and enables distinct high-level feature learning for both policy and value. Additionally, we deploy an attention mechanism in the two separate branches to facilitate the extraction of relevant features for the policy and value functions. In order to perform better on new unseen task episodes, learning minimal and compact representations is crucial [8]. A minimal set of features allows a deep RL network to avoid spurious correlation between generic attributes of the episodes (e.g., background color) and the value/policy function. Consequently, this minimal representation helps to better capture the relation between task-relevant attributes and the value/policy function. Our ablative experiments demonstrate how attention within these sub-networks accounts for gains over the state-of-the-art baseline by focusing on a minimal set

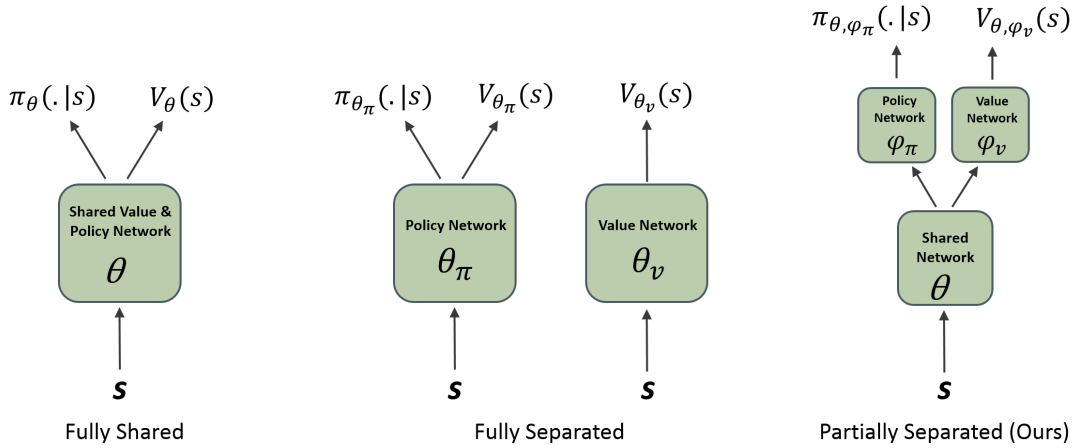

Fig. 1: Comparison of architectures: (left) a fully shared network for policy and value, (middle) two explicitly separate networks for policy and value, and (right) our proposed partially separated network for policy and value function.

of spatial features. We attribute the benefits of our approach to the hierarchical representation of features and the ability of attention mechanisms to effectively identify the components of an input pertinent to the optimization task. Further, we produce visualizations of the policy and value that reveal crucial insights into the learned policies and value functions.

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

II. Related Work

Many recent works have established that lack of generalization is a systemic problem in deep reinforcement learning and popular algorithms tend to overfit to the environment, resulting in models that seem merely to memorize surface-level details of the environment rather than learn generalizable skills [6], [7], [9], [10]. Existing solutions to the generalization problem include L2 regularization [11], dropout [12], data augmentation [13], batch normalization [12], and hyperbolic discounting [14]. It has been shown that the performance of generalization in RL increases with the number of convolutional layers while learning directly from image observation [4]. As established in [13], Procgen is a testing suite that uses procedural content generation to benchmark generalization to greater effect than traditional benchmarks. Procgen became popular in recent works advancing generalization in deep reinforcement learning [8], [15]–[17]. As discussed in [8], sharing features between policy and value functions can lead to overfitting, reducing a model's ability to generalize to new, unseen environments. In contrast to the previous methods, [5], [8] make use of fully disconnected policy and value functions. This provides greater generalization and sample efficiency than earlier counterparts. However, this performance comes at a price; more parameters than previous approaches require more computing power. In addition, certain Procgen environments require specific hyperparameters to produce reported performances. Our method provides results consistent with those in [8] with fewer convolutional layers while reducing the need for hyperparameter tuning.

Literature exploring the potential of attention mechanisms in neural networks has found success across a wide array of domains, including natural language processing and vision, both as part of convolutional layers and as stand-alone layers [18], [19]. Attention has also been utilized in vision models with great success, yielding strong performance while requiring less computing power and fewer input parameters, with self-attention and dual attention models being used in pursuits such as image classification and scene segmentation [19], [20]. The use of attention mechanisms in deep reinforcement learning, however, is less prevalent. Some variations of A2C incorporates a shared attention mechanism [21], [22]. [23] proposed a self-attention mechanism to rank important patches of the observation that influence the policy. [24] presented a spatio-temporal self-attention mechanism for reinforcement learning in the case of RNN-based policy. However, our work differs by combining the attention mechanism with a partially split policy and value network designed to prevent overfitting and achieve generalization.

III. Preliminaries

While generic RL algorithms deal with a single (PO)MDP, formulating the problem of generalization considers a distribution of (PO)MDP that generates similar but distinct instances of (PO)MDPs with the overall goal 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. If 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 $J(\pi_\theta) = \mathbb{E}_{p,\pi,\mathcal{T}_m}\big[\sum_{t=0}^{T} \gamma^t \mathcal{R}_m(s_t, a_t)\big]$. In our experiment, each game environment from our testbed corresponds to a distribution of POMDPs $p(m)$ while each procedurally generated level within that game corresponds to a sampled POMDP instance according to that distribution. The model is tested on the full distribution while learning from a limited number of levels ($n = 200$ in our case), thus providing an opportunity to evaluate how the model can generalize beyond the levels it encounters during training.

IV. Attention-based Partially Decoupled Actor-Critic

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

A. Partial Decoupling of Policy and Value function

Decoupling the policy and the value functions is crucial to overcome the problem of overfitting, which is the main drawback of a shared representation [8]. However, simply using two explicitly separated networks has serious inherent disadvantages because the policy function approximation depends on the gradient of the value function. [5] shows that the straightforward method of just using two separate networks for policy and value functions causes a performance decrease when compared to the shared network architecture. To address this issue, research that uses a separate network approach to optimize policy and value functions often also uses an auxiliary value [5] or advantage head [8] in the policy network (See the policy network at the middle in Figure 1). This auxiliary head provides a helpful gradient to the separate policy network to learn better task-relevant policy representation, whereas the separate value network optimizes the value function that plays the original role of the critic. Moreover, the two network models introduce additional hyperparameters such as update frequency of the policy network, update frequency of the value network, and coefficients for the advantage loss.

We leverage the hierarchical representation of image features to design our network. Generally, the low-level features include minor details such as lines, edges, dots, and curves, whereas the high-level features are composed of multiple low-level features. Based on this, we hypothesize that, although the high-level features responsible for accurate estimation of the policy and value functions may differ, the low-level features that constitute the high-level features are almost similar for both. Deep convolutional neural networks (CNN) can learn the feature representations hierarchically using a sequence of convolutional and pooling layers. Initial convolutional layers in a neural network learn the filters to capture low-level features while the later layers in the pipeline learn to identify larger objects and shapes. Therefore, we bifurcate the network earlier, at the convolutional layer level, instead of merely separating the policy and the value head as in the case of a fully shared network presented in [4]. This helps to decouple the high-level feature learning for policy and value on top of the same features learned by the shared network. Thus, our network comprises three parts: the part of the network shared between policy and value parameterized by θ, a second part dedicated to policy learning parameterized by ϕ_π, and a third part dedicated to value function approximation, which is parameterized by ϕ_v. The overall network is trained all together to optimize the following objective:

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

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 [25]:

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

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:

$$L_V(\theta, \phi_v) = \hat{\mathbb{E}}_t\Big[\big(V_{\theta,\phi_v}(s_t) - \hat{V}_t^{targ}\big)^2\Big] \tag{2}$$

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

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

B. Relevant Feature Learning using Attention

Attention is an effective means to learn high-quality and meaningful representation. To ensure more discriminative and relevant feature learning by the partially separated policy and value sub-networks, we propose to incorporate individual attention mechanisms within the corresponding blocks of the network. We apply attention modules similar to the Squeeze and Excitation (SE) network, which explicitly models the interdependencies between the channels of its convolutional

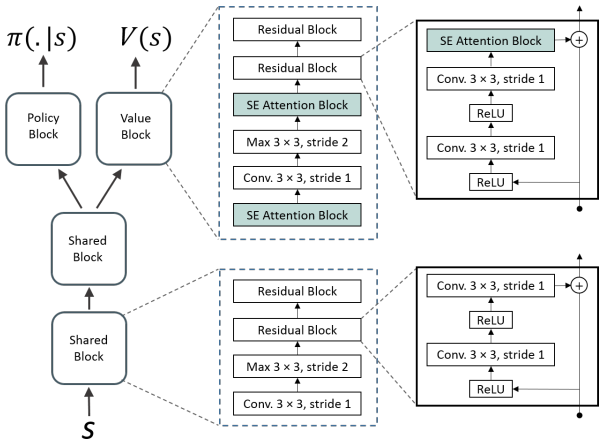

Fig. 2: Details of our proposed architecture. The separated blocks for the policy and value function (at the top) incorporate attention module (shown as shaded blocks) inside the residual blocks as well as one before the first convolutional layer and one after the max pooling layer. The shared blocks (at the bottom) are similar to the IMPALA CNN architecture [26]. They do not have an attention module.

features [27]. SE block leverages global information to put relative importance on more useful features than less useful ones. We use SE blocks only in the later layers (split value and policy sub-networks). Using a SE block in the later layers of a deep network enables distinct feature learning in a highly class-specific manner, while in the initial layers it learns in a class-agnostic manner. We argue that this characteristic of the SE block makes it a suitable choice for our task, where distinct features relevant to policy and value must be learned.

The placement of the attention block within the sub-networks is a crucial design choice. We identify multiple key positions in the base IMPALA-CNN architecture [26] and conduct extensive experiments to determine the proper positions for the attention block. Based on the empirical analysis, we incorporate one SE attention block in each Residual Block just before the residual connection as shown in Figure 2. We also add extra SE attention blocks outside of the Residual Block: one at the beginning of the sub-networks to initially prioritize the channels and another just after the max pooling layer for re-weighting the downsampled channels (See Section V-A and Figure 2 for details). We have found that the placement of the attention block either after the pooling layer or before the pooling layer makes a significant difference in terms of learning improved representation.

Following [27] to utilize global information beyond the local receptive field of filters, in our implementation, the squeeze operation in the attention block first encodes a channel descriptor $z \in R^C$ through global average pooling. Each element of z is defined as:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j), \tag{3}$$

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:

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \delta(W_1 z)), \tag{4}$$

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

$$\bar{x_c} = F_{scale}(u_c, s_c) = s_c u_c, \tag{5}$$

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 experiments, it is evident that the attention mechanism coupled with the contribution of Section IV-A helps in attaining the high level generalization.

V. 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 [4], which consists of 16 environments. We experiment on the easy difficulty setting for 25 million total timesteps as recommended in [4]. We train the model on 200 levels and test on the full distribution of the levels. Procgen environments are designed to have a discrete 15-dimensional action space. The agent is required to learn the optimal policy directly from image observations. The observation space is of the size 64×64×3. No frame stacking is used implying that the next state depends only on the present observation.

A. Network Architecture

Following previous works involving Procgen, we chose IMPALA's deeper residual CNN architecture as our backbone [4], [5], [8]. This model strikes a good balance between the achieved reward and the required computational power [4]. This particular IMPALA CNN architecture has 15 convolutional layers divided into three blocks [26]. Each block has a configuration similar to 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 an ReLU activation layer. APDAC shares the first two blocks (10 convolutional layers) of the IMPALA CNN, then branches out for the third block. Thus, APDAC employs five separate convolutional layers each for policy and value functions in order to learn features distinctly. Finally, the separated policy and value blocks incorporate one attention unit per residual block along with one at the very beginning of the separated

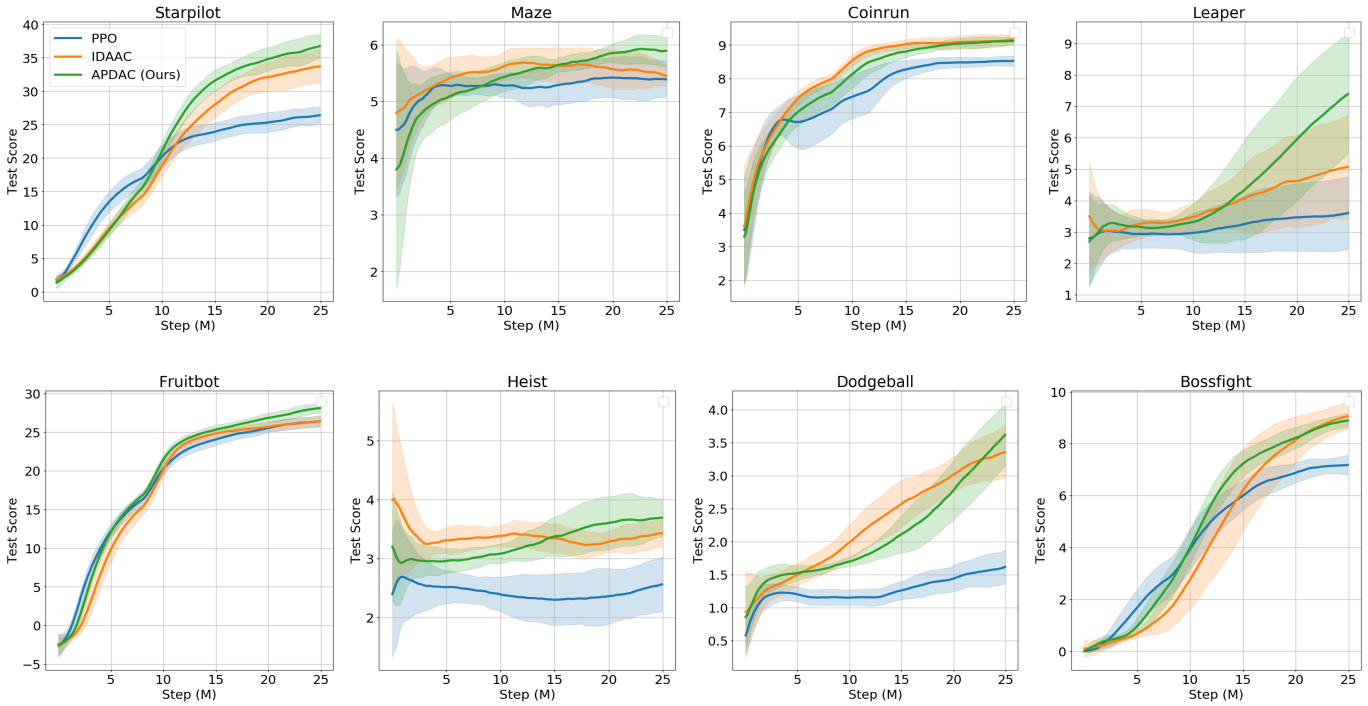

Fig. 3: Test performance of PPO [4], IDAAC [8], and our proposed APDAC over eight Procgen environments. Means and standard deviations are calculated over 10 trials, each with a different seed.

blocks and another just following the max pooling layer. Figure 2 shows the details of the architecture.

We extend the implementation of IDAAC [8] to experiment with our APDAC architecture. 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. When training PPO and IDAAC, whenever applicable, we followed the same hyperparameter setup from [8]. The only difference is that we reduced the number of mini batch sizes to minimize the required computational power. We use reduction ratio $r = 8$ for the SE attention block.

B. Generalization Performance on Test Distribution

In our experiments, we compared the performance of our network architecture, APDAC, with two representative state-of-the-art methods for all 16 Procgen environments. PPO serves as a representative of the models that use fully shared policy and value networks whereas IDAAC represents those with separate policy and value networks [25] [4] [8]. In addition, IDAAC uses a discriminator loss to decrease the dependency on irrelevant instance-specific aspects of the environment. Figure 3 shows the rolling mean test scores averaged over ten trials for each of the eight environments out of sixteen from the Procgen benchmark. The rolling standard deviations between trials are calculated as well, with confidence intervals bounding one standard deviation above and below each curve. Table I presents the scores on test levels after training for 25M environment steps for all sixteen environments. In these results, APDAC attains significant gains in performance compared to the standard shared network approach, PPO [4], [25]. Furthermore, APDAC performs better or competitively compared to the existing state-of-the-art, IDAAC, and does so with fewer convolutional layers and less computing time. In our particular instantiation, while two separate networks in IDAAC require 30 (15 for policy; 15 for value) convolutional layers, our approach requires only 20 (10 shared; 5 for policy; 5 for value). The number of parameters added for attention

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

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

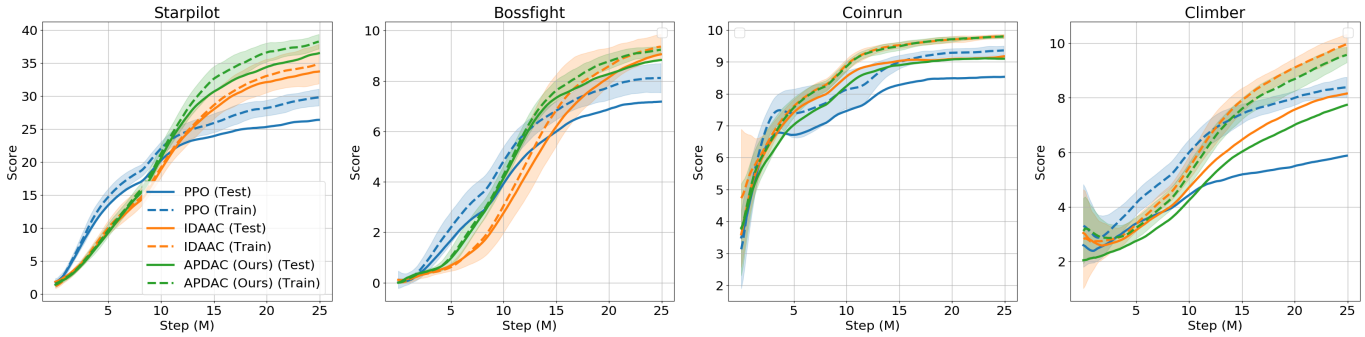

Fig. 4: Training and test performance of PPO [4], IDAAC [8], and proposed APDAC over four Procgen environments. Means and standard deviations are calculated over 10 trials, each with a different seed. The dotted lines refer to the training reward while the solid line refers to the test reward. In case of APDAC, the gaps between the dotted line and solid line (green) are similar to the fully decoupled approach, IDAAC (orange) and both are better than PPO (blue).

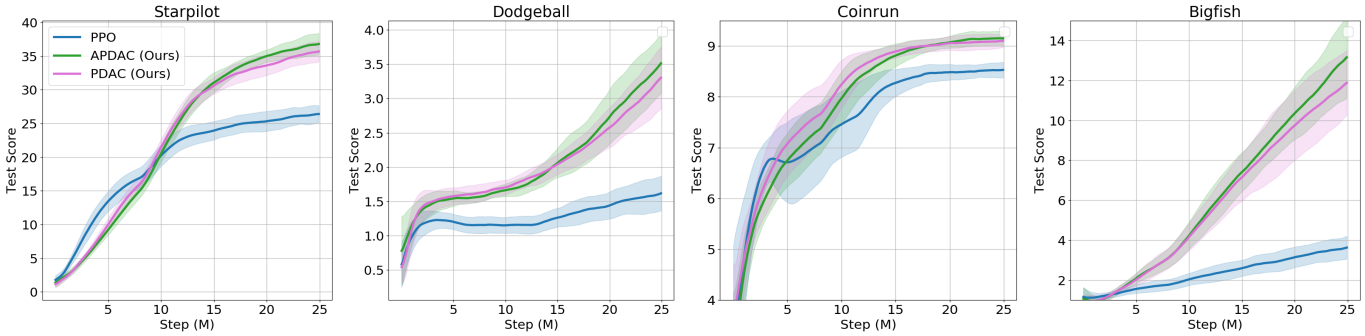

Fig. 5: Ablation study of the proposed method APDAC with the standard baseline PPO (no decoupling, no attention) and the ablated version PDAC (partially decoupled but without attention) on the test distribution for four Procgen environments. Means and standard deviations are calculated over 10 trials, each with a different seed.

TABLE II: Comparison of number of required convolutional layer and computational time for different approaches

Methods	No of Conv. Layer	GPU Hours
PPO [4]	15	3-4
IDAAC [8]	30	5-12
APDAC (Ours)	20	3-5

blocks is not significant. Table II shows this comparison. Also, APDAC requires a run time similar to the shared network approach (PPO) which is far less than the fully decoupled approach (IDAAC) in most cases. The reported hours are based on experiments on an Nvidia GeForce GTX 2080 GPU. Thus, we conclude that APDAC succeeds as an efficient hybrid architecture that improves upon its constituent network topologies.

C. Assessing the Generalization Gap

In addition to analyzing the performance on the test distribution, it is crucial to investigate the performance gap between training and test tasks or levels. This provides an indication of how well the model performs on unseen tasks compared to ones that belong to the training set. It is obvious from Figure 4 that APDAC is capable of reducing the train-test gap (solid vs. broken line) at the same level of a fully decoupled approach, IDAAC and far better than the fully shared approach, PPO.

D. Ablation

To evaluate the contributions of each proposed component, we further experiment with an ablated version of our proposed approach, which eliminates all attention blocks. Thus, this network includes only the partially separated policy and value representations and does not incorporate the contribution mentioned in Section IV-B. We denote this model as Partially Decoupled Actor-Critic (PDAC). To determine the difference in performance brought by this ablation, we compare the results of PPO, APDAC, and the ablation, PDAC, using the same experimental setup as before. Figure 5 shows the ablation results from four example environments. APDAC performs better than PDAC in most cases. Indeed, the comparison with PPO clearly shows that the main performance gain of APDAC comes from the partial separation of policy and value networks. Thus, sharing the initial part of the network does not harm performance and, in fact, reduces the required number

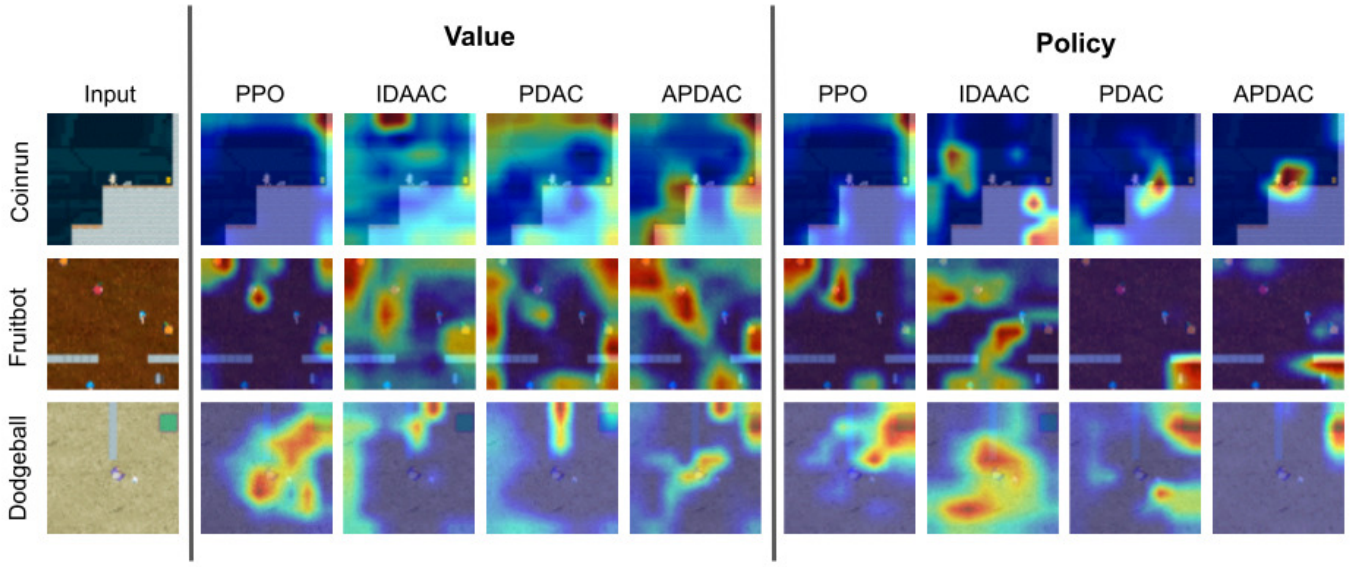

Fig. 6: Visualization of the value and policy representation for PPO [4], IDAAC [8], the ablation, PDAC, and the proposed approach, APDAC, on an input sample from three Procgen environments. The red regions correspond to the higher value of gradients while blue regions correspond to the lower value. For both policy and value, compared to other approaches, APDAC exclusively prioritizes informative spatial regions (red marked) and constructs asymmetrical representation.

of convolutional layers compared to an architecture with fully separate value and policy networks. The marginal performance gains between APDAC and the ablation are an incremental aspect of this work; however, attention is still a growing field of study in RL, and our research shows an opportunity to use attention to achieve generalization (Also, see Figure 6).

E. Visualization of Value and Policy using Grad-CAM

To understand the agent's ability to reason about the environment, we visualize where the agent is attending while making its decisions. This highlights the important spatial regions where exactly the agent is focusing on and helps us to understand in-depth about the key information that is influencing the agent's behavior. We generate activation maps for both value and policy using the Grad-CAM technique, which uses gradients back-propagated from the final layer [28]. Specifically, we set the last layer before the fully connected part as the target layer and set the final value and policy head outputs as the target concept. We visualize the policy in an action-indiscriminative way as we are more interested in learning the difference between policy and value activations rather than action-specific policy.

Figure 6 presents visualizations of the value and policy for PPO, IDAAC, proposed APDAC, and the ablation PDAC. The red regions correspond to the higher value of gradients while blue regions corresponds to the lower value. APDAC produces more fine-grained and explainable visualization compared to other approaches. Figure 6 shows that APDAC policy gradients are highly concentrated in specific spatial regions (see the red regions). A policy can be optimal by encoding the minimal task-relevant information from the observation [8]. The visualization reveals that APDAC policy attempts to focus only on the key information required to act immediately on the environment. Compared to the ablation PDAC, APDAC produces more clear gradients and consequently concise representation. We observe that APDAC value and policy gradients are highly asymmetrical. This confirms that an asymmetry truly exists between the policy and value function representations that needs to be addressed to achieve better generalization. PPO produces identical activation maps for policy and value, thus failing to encode distinct features for policy and value owing to its shared representation. While IDAAC representations are asymmetrical, it is evident that these are not plausible enough to reason about the action in the environment.

In Figure 6, the first row is from the game of Coinrun where the goal of the agent is to collect the coin at the far right of the level while avoiding the stationary saw obstacles, moving enemies, and chasms. In the sample input, the yellow coin is at the right, and the blueish saw obstacle is just in front of the agent. We can see the APDAC value function places more attention on the coin and the specific edges of the terrain. On the other hand, the policy put attention on the saw obstacle coming ahead as this needs to be bypassed.

In the game Fruitbot (the second row in Figure 6), a robot needs to navigate through the gaps in the walls to collect fruit along the path. It receives a positive reward for collecting fruit, however, receives a negative reward for a non-fruit object. The APDAC value column highlights all the fruits in red while the non-fruit objects are in the blue region. In contrast, the policy highlights the wall just in front of the robot as this wall must be avoided.

For Dodgeball (the third row in Figure 6), the player spawns

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

VI. Conclusion

In this work, we have simultaneously addressed the issue of limited generalization due to overfitting in a fully shared actor-critic network and the added complexity in the case of fully decoupled ones. Our solution, APDAC, differs from existing methods in the partial separation of its network and adding attention mechanisms to each split sub-network. Our empirical results demonstrate highly competitive performance compared to a fully decoupled state-of-the-art approach while reducing the number of required convolutional layers and the computational cost. APDAC learns compact policy representations that are robust to the variation of the unseen task episodes. APDAC appears as a promising way forward in the pursuit of generalization in deep RL on the grounds of performance, efficiency, and interpretability. Although attention greatly improves representation learning, the limited performance gap between APDAC and the ablation can be considered a limitation of the current work. A hopeful future direction is to investigate more beneficial structures for the attention mechanism.

Acknowledgments

We would like to thank Nazmun Akter Pia and Kevin Duong for their assistance in creating the visualizations and rendering.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[3] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

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

[6] 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

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

[8] 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.

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

[10] R. Raileanu and T. Rocktäschel, "Ride: Rewarding impact-driven exploration for procedurally-generated environments," *arXiv preprint arXiv:2002.12292*, 2020.

[11] 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.

[12] 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.

[13] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 1282–1289. [Online]. Available: https://proceedings.mlr.press/v97/cobbe19a.html

[14] 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*.

[15] M. Igl, G. Farquhar, J. Luketina, W. Boehmer, and S. Whiteson, "Transient non-stationarity and generalisation in deep reinforcement learning," in *International Conference on Learning Representations*, 2020.

[16] 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.

[17] B. Mazoure, A. M. Ahmed, R. D. Hjelm, A. Kolobov, and P. MacAlpine, "Cross-trajectory representation learning for zero-shot generalization in rl," in *International Conference on Learning Representations*, 2021.

[18] D. Hu, "An introductory survey on attention mechanisms in nlp problems," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2019, pp. 432–448.

[19] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3286–3295.

[21] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2961–2970.

[22] E. Barati and X. Chen, "An actor-critic-attention mechanism for deep reinforcement learning in multi-view environments," *arXiv preprint arXiv:1907.09466*, 2019.

[23] Y. Tang, D. Nguyen, and D. Ha, "Neuroevolution of self-interpretable agents," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 414–424.

[24] A. Manchin, E. Abbasnejad, and A. v. d. Hengel, "Reinforcement learning with attention that works: A self-supervised approach," in *International Conference on Neural Information Processing*. Springer, 2019, pp. 223–230.

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

[26] 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.

[27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[28] 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.