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

Anonymous Author(s)
Affiliation
Address
email

Abstract

Existence of policy-value representation asymmetry negatively affects the generalization capability of the traditional actor-critic architecture that uses a shared representation of policy and value. Fully separated networks for policy and value avoid overfitting by addressing this representation asymmetry. However, two separate networks introduce high computational overhead. Previous work has also shown that partial separation can achieve the same level of generalization in most tasks while reducing this computational overhead. Thus, the questions arise *Do we really need two separate networks? Is there any particular scenario where only full separation works?* In this work, we attempt to analyze the generalization performance compared to the extent of decoupling. We compare four different degrees of subnetwork separation, namely: fully shared; early separated, lately separated, and fully separated on the RL generalization benchmark Procgen, a suite of 16 procedurally-generated environments. We show that unless the environment has a distinct or explicit source of value estimation, partial separation can easily capture the necessary policy-value representation asymmetry and achieve better generalization performance in unseen scenarios.

1 Introduction

Deep reinforcement learning algorithms are an impressive contribution of the deep learning era because they can learn different control tasks directly from interaction with an environment. However, a large number of training samples is necessary to optimally train the models. Thus, deep RL algorithms suffer from poor generalization performance when applying the learned policy to an unseen scenario. Raileanu and Fergus [2021] showed that policy-value representation asymmetry is an underlying cause for poor generalization performance in shared actor-critic architecture. Value estimation in a particular state depends on instance-specific features; however, learning the policy only depends on task-specific features. Raileanu et al. hypothesize that ignoring this asymmetry, while learning a joint representation through a shared network for both policy and value, limits the agent's capacity to learn a generalizable policy. Combining the two representation guides the policy to be unnecessarily biased to the value features. Thus the learned policy overfits to the training instances and performs badly in terms of generalization.

One possible solution is to use two different networks for policy and value that will capture distinct features [Cobbe et al., 2021][Raileanu and Fergus, 2021]. However, the use of two separate network creates a significant bottleneck regarding computation time. Also, the dependency of policy function on the value gradients requires extra precaution to design the network. Nafi et al. [2021] propose a workaround that compromises between the two by partially separating the policy and value network. This approach attains competitive performance compared to the fully separated counterpart while

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

requiring less computational time. This architecture also eliminates the need for an additional value (advantage) head to propagate the value gradient to the policy parameter.

All the abovementioned approaches were evaluated on the demanding generalization benchmark Procgen [Cobbe et al., 2020]. Procgen offers sixteen different game environments that provide access to the unlimited number of levels of a particular game with diverse background and game attributes such as agent position, shape and color of the game assets, enemy spawn time. The game level target remains same throughout all levels, and the learned agent needs to perform well in the whole distribution of levels while learning from a limited number of levels.

We observe that while all these approaches offer a solution to the specific problem they identified, no solution is perfect for all sixteen game considering both aspects - generalization performance and required computation time. Full separation of policy and value introduce very high computational overhead. In our experiment, on a single core GPU, for most of the games, we notice 5 times higher running time for the fully separated approach, IDAAC [Raileanu and Fergus, 2021], compared to the fully shared baseline PPO [Cobbe et al., 2020] or partially separated approach APDAC [Nafi et al., 2021]. On the other hand, the partial separation achieves competitive performance in most games, but fails in some cases.

Thus, the question remains open what approach to use if we encounter a new environment. Unfortunately, there is still no guideline for deciding how much decoupling is enough for policy and value to achieve a reasonable performance, by looking into the environment properties, while keeping the computational burden low. In most cases, substantial compromise in generalization performance is not acceptable just because of computational overhead. This also lead to the question "Can separating the network early (having most of the layer separated) improve the overall performance?"

In this work, we investigate the effect of different extent of decoupling on the agent's performance. Our analysis primarily leverage the existing literature, however, devises purposeful insight. We evaluate four architecture with different degree of separation - fully shared (no separation), early separated, lately separated, and fully separated. We attempt to delve into the learned representation for policy and value in all these settings to identify whether and how the separation helps. We conclude that full separation is a must when there is a clear distinction between the source of value and policy features. Otherwise, partial late separation suffice, possibly be the best option, however, too early separation may decrease the performance, in addition to higher computation time.

2 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 Farebrother et al. [2018], Packer et al. [2018], Cobbe et al. [2019a] which has led to an emphasis on developing intelligent agents that avoid overfitting and can generalize well to unseen data Rajeswaran et al. [2017], Justesen et al. [2018], Grigsby and Qi [2020], Lyle et al. [2022]. Some of the methods that have been proposed include regularization techniques like dropout Igl et al. [2019], batch normalization Cobbe et al. [2019a], Hu et al. [2021], and data augmentation Cobbe et al. [2019b], Wang et al. [2020], Raileanu et al. [2020], Yarats et al. [2020], Zhang and Guo [2021]. A feature-swapping regularization technique to avoid observational overfitting is proposed in Bertoin and Rachelson [2022] whereas Igl et al. [2020], Lyle et al. [2022] use policy distillation to improve generalization. Another aspect of study has been the network architecture such as the work of Raileanu and Fergus [2021] who use decoupled policy and value networks to improve generalization and show that sharing policy and value functions leads to overfitting. In this context, while PPO Schulman et al. [2017] proposed a fully shared actor-critic architecture, Nafi et al. [2021] proposed the use of partially decoupled actor and critic networks that reduce the overall parameter count while performing comparably to the fully decoupled architecture. Cobbe et al. [2021] introduce a phase-wise training whereas Zhang et al. [2020], Agarwal et al. [2021a] use bisimulation metrics to study similarity between states to learn task-relevant representations. Some studies suggest a link between interference and generalization in temporal difference learning and show that TD methods lead to under-generalizing parametersBengio et al. [2020]. Mazoure et al. [2020] propose to predict the future states by maximizing the mutual information between its internal representation of successive time steps. Jia et al. [2022] introduced generalist-specialist training framework, while Paischer et al. [2022] propose the use of language models with history compression which enables memory to store abstractions of the observations to allow for generalization. One

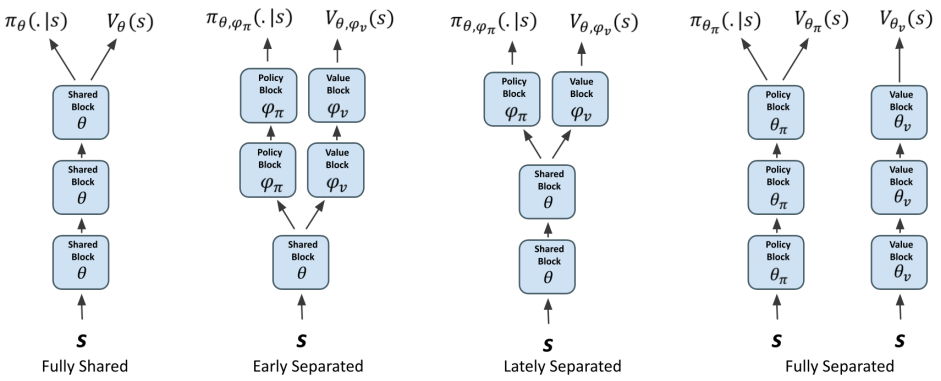

Figure 1: Architectures with different extent of decoupling for policy and value networks

recent work identifies the value network as being much more prone to overfitting and propose a Delayed-Critic Policy Gradient (DCPG) method which trains the value function less frequently and with more training data compared to the policy Moon et al. [2022].

3 Methodology

To analyze the sensitivity to decoupling of policy and value, we create four different scenarios (1) there is no separation of the policy and value network other than the policy and value heads at the end, (2) the network is separated early in the layers (3) the network is separated in the more downstream layers (4) solely two separate network. Figure 1 presents the four architectures. We use the large IMPALAN-CNN architecture as the base network as used by previous state-of-the-art works [Cobbe et al., 2021] [Raileanu and Fergus, 2021]. Also, the Procgen benchmark authors present their results using this version of IMPALA-CNN architecture [Cobbe et al., 2020]. This deeper IMPALA CNN architecture has three blocks each having a configuration of Conv Layer - Pooling Layer - Residual Block - Residual Block, where each residual block in turns has two Conv layers. Thus the whole architecture includes 15 convolutional layers in total [Espeholt et al., 2018].In the rest of this section, we describe the details of the four architectures we used and how we implemented those using the IMPALA-CNN backbone. We also discuss their loss function and the optimization process.

3.1 Fully Shared

By *fully shared*, we denote the traditional architecture that shares the network for the actor and the critic (policy and value). Specifically, we share the IMPALA-CNN base network that means all the convolutional layers for policy and value. However, we just separate the final fully connected layers that represents the policy and value head. This is very similar to many existing standard Proximal Policy Optimization (PPO) [Schulman et al., 2017] implementation [Cobbe et al., 2020] [Raileanu and Fergus, 2021]. Given the network paramet 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, $S_\pi(\theta, \phi_\pi)$ is the entropy bonus for exploration, and α_v and α_s are the corresponding coefficients. Generally, Trust Region Policy Optimization (TRPO) method maximizes a surrogate objective function defined as:

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

where $r_t(\theta) = \frac{\pi_{(\theta)}(a_t|s_t)}{\pi_{(\theta)_{old}}(a_t|s_t)}$ is the probability ratio between the new policy and the old policy, and $\hat{A}_t$ refers to the advantage estimate at timestep t. To avoid excessively large policy updates, PPO clips the value of $r_t(\theta)$ between the intervals of $[1-\epsilon, 1+\epsilon]$ and takes the minimum between the

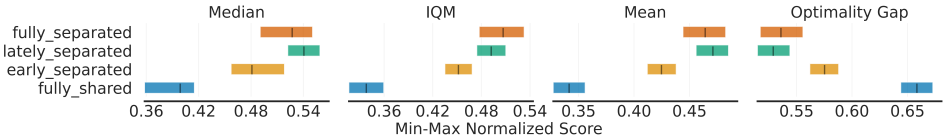

Figure 2: Min-max normalized score for all four variants: fully shared, early separated, lately separated, and fully separated across all the 16 environments in Procgen benchmark

original value of $r_t(\theta)$ and the clipped one. Thus the final clipped surrogate objective function looks as follows:

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

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

3.2 Early Separated

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 the part of the network shared between policy and value is parameterized by θ, the separated policy network is parameterized by ϕ_π, and the separated value network is parameterized by ϕ_v. Then, we optimize the objective jointly for all parameters using the PPO loss:

$$J_{ES}(\theta, \phi_\pi, \phi_v) = J_\pi(\theta, \phi_\pi) - \alpha_v L_V(\theta, \phi_v) + \alpha_s S_\pi(\theta, \phi_\pi) \tag{2}$$

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 the entropy bonus.

3.3 Lately Separated

In this variant, we share the first two block of the IMPALA-CNN architecture and use two separate third block for policy and value. That means first 10 convolutional layers are shared while there are two different set of 5 convolutional layers for policy and value. As before, 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, we optimize the objective jointly using the same PPO loss:

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

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 the entropy bonus. This is similar to the partial separation proposed in [Nafi et al., 2021]. The difference between the early separated and this one is that the number of the shared parameters represented by θ is higher in the lately separated.

3.4 Fully Separated

As the name suggests, we use two fully separate network for policy and value. This is exactly similar to the one proposed by Raileanu and Fergus [2021] that fully disentangle the policy and value representation. We also keep the extra advantage head in the policy network as without any sort of value or advantage gradient, isolating the policy network, the policy optimization process completely fails [Cobbe et al., 2021]. Since there are two different networks, the optimization takes place in two phases. First it optimizes the policy network and then optimize the value network. The policy network parameterized by θ_π is optimized for:

$$J_{FS}(\theta_\pi) = J_\pi(\theta_\pi) + \alpha_s S_\pi(\theta_\pi) - \alpha_A L_{A_\pi}(\theta_\pi) \tag{4}$$

Here $L_{A_\pi}(\theta_\pi)$ is the advantage loss coming from the additional advantage head used to support the policy network [Raileanu and Fergus, 2021]. On the other hand, the value network, parameterized by ϕ_v, optimizes the value loss defined as follows:

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

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

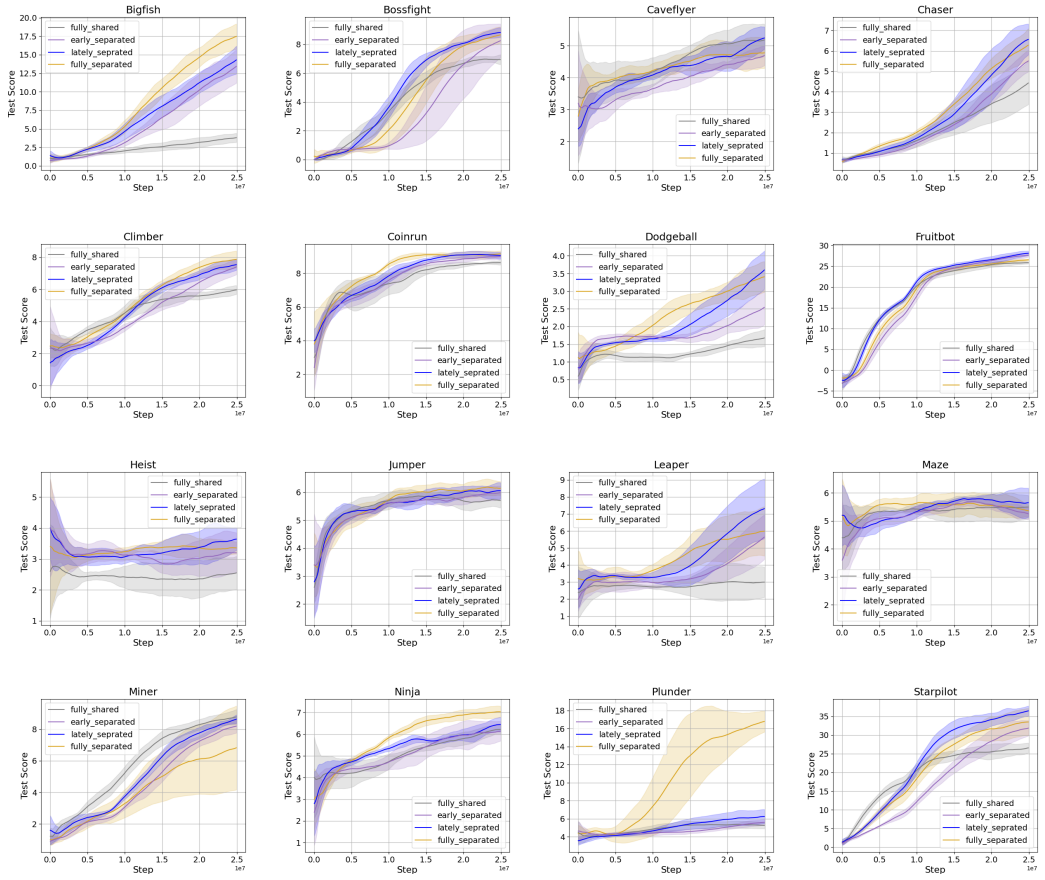

Figure 3: Testing performance for fully shared, early separated, lately separated, and fully separated architecture for all the 16 environment of Procgen benchmark. Mean and standard deviation are calculated over 5 trials, each with a different seed.

4 Results and Discussions

We conduct our experiments against all the 16 game environments available in the Procgen benchmark [Cobbe et al., 2020]. The highly diverse environments provide us the opportunity to analyze and deduce claims regarding when to deploy which one. Figure 2 shows the min-max normalized score across all the 16 environments as introduced by Agarwal et al. [2021b]. We can see the mean score for the lately separated model outperforms all other approaches while sharing a significant portion of the network. At the same time, the early separated model having a large portion of the network separated for the policy and value still performs worse than the late separation. However, the general assumption is early separation would work best because this should better capture the policy-value representation asymmetry through its larger separated part. We argue that this deviation is because of the higher isolation of the policy network. As the separated part increases in the early separation, the downstream policy network suffers from the value gradient more and as suggested by Cobbe et al. [2021] policy network without enough value gradient may collapse. This may be overcome by introducing an extra value or advantage head even in the partially separated policy network, however, this will again increase the computational overhead reducing the benefit of the partial separation.

In the next phase, we look at the empirical results for all individual games to evaluate how different degrees of separation affects them. Figure **??** presents results across the entire Procgen benchmark. We observe that, in all the games except *Plunder*, lately separated model achieves competitive reward against the fully separated model. As partially separated models requires significantly less time than the fully separated approach, we recommended to use that in general. However, we inspect the issue of complete failure in the game of *Plunder*. Our investigation reveals that unlike other games,

Plunder includes an on-screen timer that slowly counts down. When timer runs out, the episode ends at that moment. Thus this on-screen timer acts as a life bar for the agents which has no relation to the policy. The policy need to learn to avoid hitting friendly ships and destroying the enemy pirate ships by firing cannonballs. However, the life bar is an important source of value estimation of the state. Thus we conclude when explicit source of value estimation is present in the input state, any one should consider learning representation for policy and value through two fully separate network irrespective of the time complexity to gain generalization improvement.

5 Conclusions

In this work, we attempted to infer meaningful insight from the existing works through carefully designed ablation studies regarding partial and full decoupling of policy and value. We presented comparative analysis of the models that use different level of separation for the policy and value function in procedurally generated environments. Our work clearly identifies when to leverage two fully separate network even though that entails high computational complexity. The contribution thus focuses on practical criteria that will help DRL users to decide what type of network to deploy when encountering a new environment to achieve better generalization.

References

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 8787–8798. PMLR, 2021.

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

Nasik Muhammad Nafi, Creighton Glasscock, and William Hsu. Attention-based partial decoupling of policy and value for generalization in reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 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.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

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 2019a. URL `https://proceedings.mlr.press/v97/cobbe19a.html`.

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

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.

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`.

Clare Lyle, Mark Rowland, Will Dabney, Marta Kwiatkowska, and Yarin Gal. Learning dynamics and generalization in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 14560–14581. PMLR, 17–23 Jul 2022.

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*, 2019.

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.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019b.

Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *Advances in Neural Information Processing Systems*, 33: 7968–7978, 2020.

Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862*, 2020.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

Hanping Zhang and Yuhong Guo. Generalization of reinforcement learning with policy-aware adversarial data augmentation. *arXiv preprint arXiv:2106.15587*, 2021.

David Bertoin and Emmanuel Rachelson. Local feature swapping for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2022.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

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

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021a.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In *International Conference on Machine Learning*, pages 767–777. PMLR, 2020.

Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *Advances in Neural Information Processing Systems*, 33:3686–3698, 2020.

Zhiwei Jia, Xuanlin Li, Zhan Ling, Shuang Liu, Yiran Wu, and Hao Su. Improving policy optimization with generalist-specialist learning. In *International Conference on Machine Learning*, pages 10104–10119. PMLR, 2022.

Fabian Paischer, Thomas Adler, Vihang Patil, Angela Bitto-Nemling, Markus Holzleitner, Sebastian Lehner, Hamid Eghbal-Zadeh, and Sepp Hochreiter. History compression via language models in reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17156–17185. PMLR, 17–23 Jul 2022.

Seungyong Moon, JunYeong Lee, and Hyun Oh Song. Rethinking value function learning for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

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.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021b.