Reinforcement Learning Generalization via Partially Decoupled Policy-Value Networks

Kristof Pusztai
Rodrigo Palmaka
Ameya Chander

May 2021

1 Introduction

Recent works in reinforcement learning have had a strong focus on generalization and regularization of existing methods. These methods usually build off the PPO2 policy gradient algorithm developed by OpenAI and range from observation augmentation to policy-value function architecture alterations [1]. In our project, we decided to explore both of these frameworks, seeing how each regularization technique affects generalization, as well as trying the two together.

2 Literature Review

For observation augmentation, our starting point was studying Reinforcement with Augmented Data (RAD) [2]. This paper seems to take inspiration from CNN methods of generalization and applies it to RL generalization - these include cutouts, color-jitter, among others. Another generalization method of interest is known as MixReg which was inspired by the success of mixup in supervised learning [3]. MixReg works by by convexly combining two observations randomly sampled from the collected training batch, and then trains the RL agent on them with their interpolated supervision signal[3]. In this way, the diversity of the training data is greatly diversified, aiding in combating overfitting and granting better generalization to unseen levels.

Regarding architectures for learning policy and value functions, our focus was studying the Invariant Decoupled Advantage Actor-Critic (IDAAC) method [4]. This work claims that the current approach of shared weights between policy and value functions can cause overfitting. It proposes training separate networks for the two functions and achieves a new state-of-the-art at the time of its publishing (Feb. 2021).

3 Background

Our work builds off the PPO2 algorithm, using existing implementations to test our architectural and regularization modifications. Specifically, we use the Stable Baselines library for our project [5]. This library is a fork of Open AI's Baselines library, a popular package for working with models such as PPO. Our choice of using Stable Baselines was almost entirely a result of its support for ipynb/Google Colab notebooks - which allowed us to train our model in GPU environments.

4 Methodology

For policy-value function architecture augmentation, we implement a novel architecture which was inspired by the IDAAC model.
While IDAAC suggested completely decoupling the two networks and adding a surrogate loss, we instead propose a partially decoupled set of networks[4]. We propose keeping the ImpalaCNN [6] feature extraction layers shared and add a partial decoupling where the last layers of the two network are independent of each other. The intuition behind this novel proposition has to do with the fact that if networks are fully decoupled, the model may not train well in some cases as seen in past studies[7]. Hence, methods which completely decouple networks end up having to add some sort of surrogate loss in order to retain a relationship between the two networks. However, by only partially decoupling the networks, we inherently keep a shared relationship between the networks but also allow them to diverge in their final layers. Hence, the partially decoupled networks still share some representation among each other whilst allowing the networks to capture different details. These details might be necessary to generalize to unseen levels, as proposed by the decoupling paper[4]. To take our experimentation further, we also test against MixReg alone as well as MixReg with partial decoupling. Our training was standardized so that each model was trained to achieve a mean reward of 20 on the training environments.

5 Challenges

The main challenge we faced during this project was untangling the complexity and lack of documentation of the baseline libraries. We chose to go with Stable Baselines for its superior documentation and Google Colab support compared to the original OpenAI Baselines implementation. Despite this, we still ran into some trouble implementing custom policies such as ImpalaCNN which was crucial to get PPO2 to work with the FruitBot environment. Additionally, implementing MixReg required extensive understanding of Stable Baselines inner workings as well as the original OpenAI Baselines. Ultimately, we had to create a custom Stable Baselines policy to implement the ImpalaCNN, as well as a custom Stable Baselines model which implemented MixReg. The whole process required extensive trial and error since the differences between Stable Baselines and the original library are significant. To compound onto this, Google Colab limits GPU usage times so we were forced to switch among different google accounts to circumvent this. We were also forced to create model checkpoints so we could continue training when Google inevitably kicked us off a GPU instance after reaching the time/compute limit.

6 Results

For testing we used 500 randomly generated levels via the OpenAI procgen gym environment generator for Stable-Baselines. These levels were unseen during training and so should reflect the generalization performance of each trained agent. In Figure 1 we see that the MixReg+Decoupled agent performed the worst in all cases except for when trained on 250 unique environments, where it did the best. This is quite a surprise as the performance of this model drops back down to being the worst in the 500 unique level environment. We ran this models training multiple times to make sure there was no error in training. Despite this surprise, it is clear that combining MixReg and Partial Decoupling generally will result in worse performance. It's also interesting to note that the Baseline PPO2 Model with Partial Decoupling performs better then the standalone Baseline PPO2 Model, but slightly worse then then MixReg in all the unique training environments.

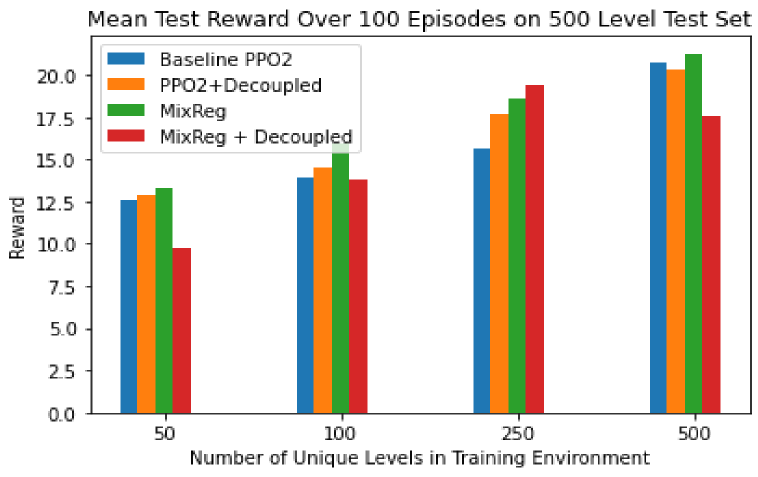


Figure 1: Test evaluations for each unique training environment

Additionally, we investigated the number of training steps required for each model to achieve a mean reward of 20 on the training environment. Figure 2 shows that both our regularization technique and MixReg seem to decrease training time for the 50 and 100 level training environments, but then increase train time for the 250 and 500 level environments.

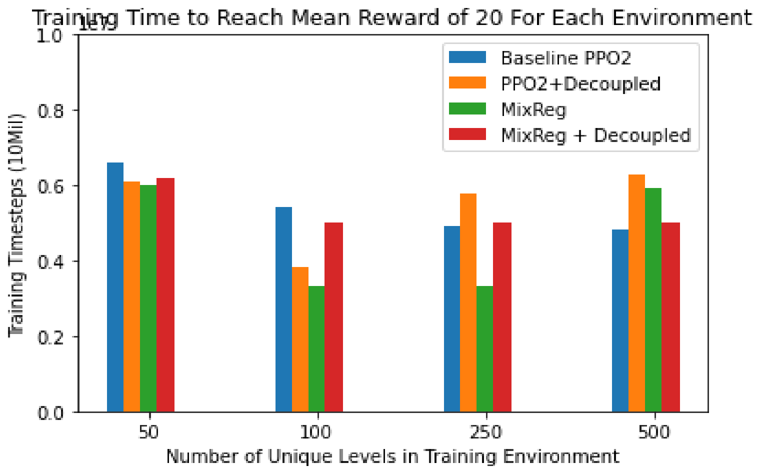


Figure 2: Training Timesteps Graph

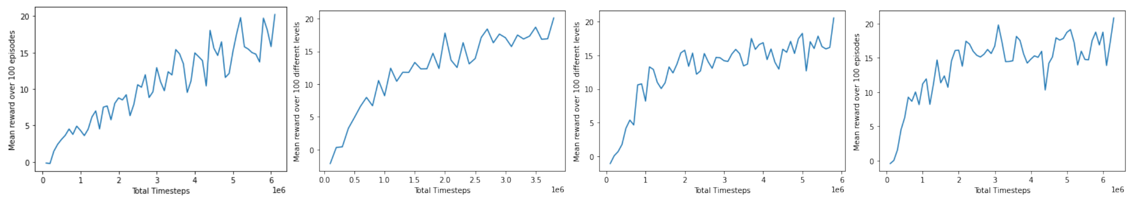
7 Conclusion

As we expected, our Partial Decoupling regularization technique shows an increase in generalization compared to the Baseline PPO2 (seen in Figure 1). However, our method does not generalize better then MixReg alone. As for combining MixReg and our method, we see that MixReg's data augmentation does not interact well with the effects of Partial Decoupling. This is contrary to our original belief that combining these two regularization techniques would lead to even better results. One explanation for this could be that combining these two regularization techniques led to a very high bias in our model which would explain ineffective generalization to unseen levels.

8 Team Contributions

- Kristof Pusztai (33%): Came up with alteration to IDAAC to create partial decoupling. Helped implement ImpalaCNN and MixReg for Stable-Baselines. Wrote and ran training code for 50 unique level environment agents. Wrote and ran training code for PPO2+Decoupled agents.
- Rodrigo Palmaka (33%): Contributed to implementing ImpalaCNN and MixReg for Stable-Baselines. Wrote and ran training code for 50 and 100 unique level environment agents.
- Ameya Chander (33%): Contributed significantly to write-up, generated final figures and evaluation rewards. Wrote and ran training code for 500 unique level environment agents.

9 Appendix



(a) 50 Level Training (b) 100 Level Training (c) 250 Level Training (d) 500 Level Training

Figure 3: Training Curves for partially decoupled (PPO2+Decoupled) Agents

References

[1] "John Schulman et al. “Proximal Policy Optimization Algorithms”. In: (2017). `"https://arxiv.org/pdf/1707.06347.pdf"`. DOI: `1707.06347`.

[2] Michael Laskin et al. *Reinforcement Learning with Augmented Data*. 2020. arXiv: `2004.14990 [cs.LG]`.

[3] Kaixin Wang et al. “Improving Generalization in Reinforcement Learning with Mixture Regularization”. In: (2020). `"https://arxiv.org/pdf/2010.10814.pdf"`.

[4] Roberta Raileanu and Rob Fergus. “Decoupling Value and Policy for Generalization in Reinforcement Learning”. In: (2021). `"https://arxiv.org/pdf/2102.10330.pdf"`.

[5] Ashley Hill et al. *Stable Baselines GitHub*. URL: `https://github.com/DLR-RM/stable-baselines3`.

[6] Lasse Espeholt et al. *IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures*. 2018. arXiv: `1802.01561 [cs.LG]`.

[7] Karl Cobbe et al. “Phasic Policy Gradient”. In: (2020). `"https://arxiv.org/pdf/2009.04416.pdf"`.