DTU Compute
Department of Applied Mathematics and Computer Science

Online Reinforcement Learning with Augmented Data on the OpenAI ProcGen Benchmark.

Jacob Bahnsen Schmidt (s184346) & Kasper Schou Telkamp (s170397).

1 Introduction

Reinforcement learning (RL) is a field within deep learning that focuses on the interaction between an agent and the environment in terms of states (s), actions (a) and rewards. By interacting with the environment, the agent learns a policy (π_θ) that defines the agent's way of behaving at a given time. In this project, we focus on generalization in RL using the coinrun environment from the Procgen Benchmark [1]. We use this benchmark to investigate the ability to generalize using different model architectures and data augmentations.

2 Proximal Policy Optimization (PPO) [4]

Supervised learning makes great use of gradient decent. This is not possible in RL as this field uses a range of different reward systems instead of having a cost function. For this reason the field of RL uses PPO, which is a concept used in RL to optimize a policy. PPO uses a clipped likelihood function (Eq. 1) to describe the optimal policy.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \quad (1)$$

Where $r_t(\theta)$ describes the probability ratio between the new and old policy, $\hat{A}_t$ is the expected advantage over time and ϵ is a hyperparameter used to limit the clipped $r_t(\theta)$. With this design, the change in probability ratio is ignored when it would improve the objective function, and it is included when it makes the objective function worse. In Fig. 1 a single time step of the clipped likelihood function is visualized.

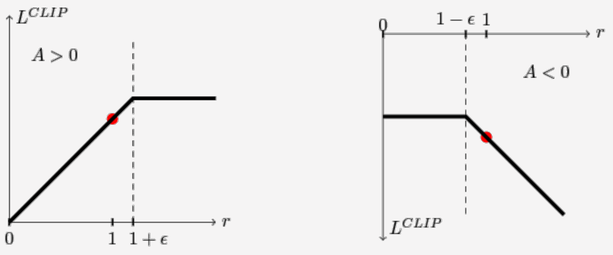

Figure 1: L^{CLIP} is visualized as a function of the probability ratio, for positive advantages (1a) and negative advantages (1b).

The clipped likelihood function is combined with the mean square error of the value function and an entropy term to yield the following objective function (Eq. 2), which is maximized each iteration.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right] \quad (2)$$

where c_1 and c_2 are hyperparameters used to scale the value function and the entropy term, respectively.

3 Model

In this project three types of models are considered, a baseline model using a Nature CNN architecture, and two models using the IMPALA CNN architecture [2]. The model architectures are sketched in Fig. 2.

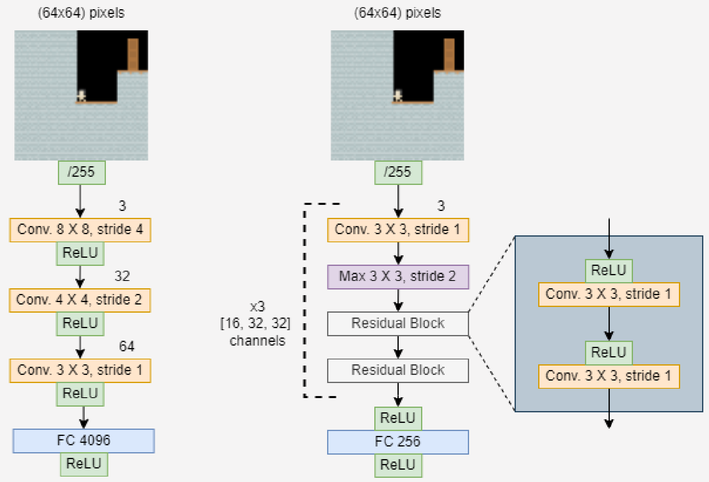

Figure 2: Model architectures. Left: Small architecture inspired by Nature CNN. Right: Large architecture inspired by IMPALA CNN.

Inspired by [3] the two IMPALA CNN models receives different inputs. One of the models receives a normal pixel-based input, whereas the other receives a pixel-based input that have been altered by a random convolution to add noise to the input. This is done to study the effect on generalization when using data augmentations. An image of the pixel-based input and the input altered by a random convolution is visualized in Fig. 3b.

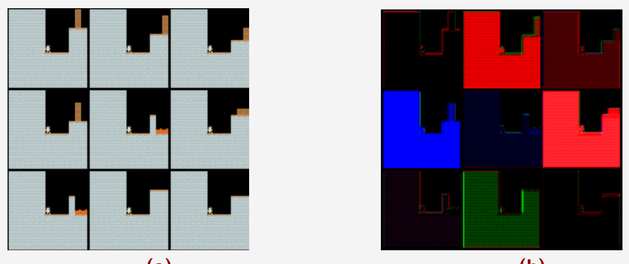

Figure 3: Pixel based input (3a) and the input altered by a random convolution (3b) for 9 different observation states.

4 Results

In Fig. 4 the training and validation reward for the models is visualized. All the models are trained using an environment with 200 levels and a black background and evaluated in an environment with unlimited levels and a random background.

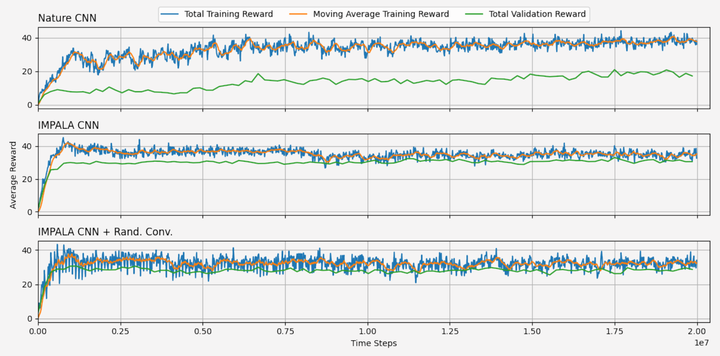

Figure 4: Total training and validation rewards for the Nature CNN, IMPALA CNN and IMPALA CNN with a random convolution.

5 Conclusion

While the Nature CNN model is able to fit reasonably well to the 200 levels it is trained on, the IMPALA CNN models are converging in fewer timesteps. Additionally, the IMPALA CNN models are also able to generalize to unseen levels and backgrounds, which is not the case for the Nature CNN model. This suggest that the model architecture is very important in generalization. Moreover, a random convolution of the input was tested and while it adds some noise to the input, an increase in the performance of the model was not observed in this project. An explanation for this can be that we simply did not train for enough timesteps.

References

[1] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning, 2020. https://arxiv.org/pdf/1912.01588.pdf.

[2] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. https://arxiv.org/pdf/1802.01561.pdf.

[3] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data, 2020. https://arxiv.org/pdf/2004.14990.pdf.

[4] J. Schulman, O. Klimov, F. Wolski, P Dhariwal, and A Radford. Proximal policy optimization algorithms, 2017. https://arxiv.org/pdf/1707.06347.pdf.

DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science