Technical University of Denmark

Deep Reinforcement Learning Generalization in Video Games

Nicolae Ghidirimschi, Adrian Macarenco, Alan Mansour

Introduction

Reinforcement learning (RL) is a machine learning paradigm used heavily in developing smart agents that are able to excel at various video games, in particular ones with non-deterministic environments, where it might take a significant amount of time before the consequences of an action can be observed. It has sparked much interest in the research community, and thanks to multiple advancements and improved techniques from researchers across the world, it can nowadays be tuned and used as a powerful tool to train smart agents in the most diverse of games and environments. In 2018, `OpenAI` has used RL to train an agent that could play against an entire team for the multiplayer online game *Dota 2*. After a few months of training, it has been able to beat the back-then world champion team in a live exhibition match. Despite its major success, this achievement is hardly replicable in practice not only due to long training time, but also due to the computational power required. More concretely, a total of 128000 CPUs and 256 GPUs were used to train this network, a significant requirement even for a big tech company.

Procgen is a framework developed by the same `OpenAI` team, which is able to procedurally-generate 16 dynamic and non-deterministic minimalist environments. Their high diversity within and across environments, coupled with their simplicity and resemblance to more complex real games (such as *Dota* and *Starcraft*) offers an experimentally convenient sandbox. Using this, various RL algorithms and techniques can be applied, observed and evaluated without the time and computational power limitations inherent to more complex games.

Project goal

The goal of this project is to research the state-of-the-art RL algorithms and techniques and apply them to train a model that is able to generalize beyond the environment in which it is trained. More concretely, we aim for a model that trained on easy levels of the *Starpilot* game (*see Fig. 1*) from the **Procgen** benchmark, is able to a certain degree generalize to more complex environments of the same game, such as ones with harder difficulty or a background image.

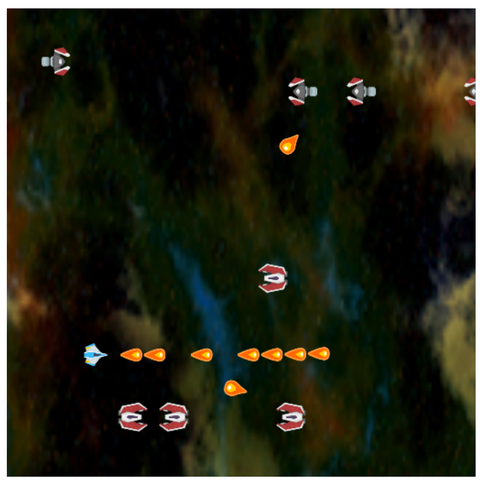
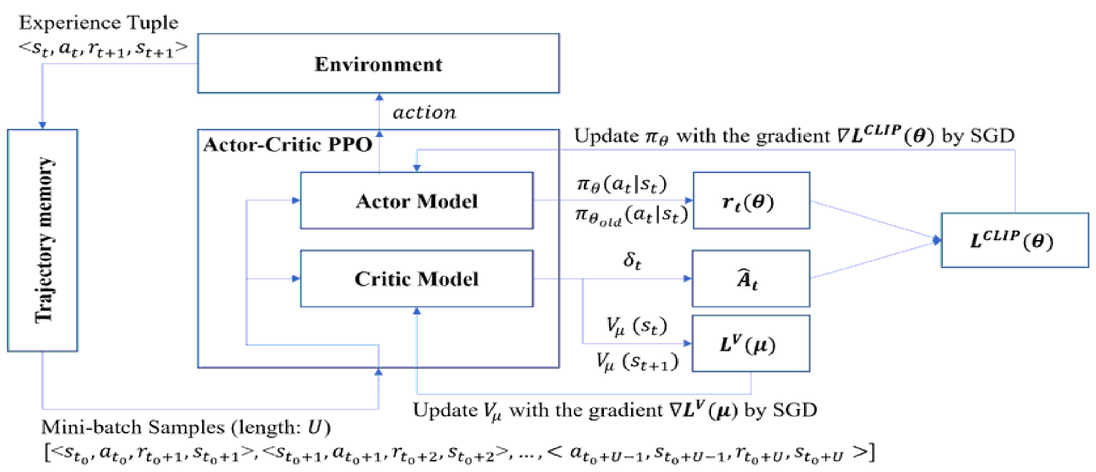


Fig. 1: A frame from the Starpilot game

Actor-Critic method

We implemented and used **Proximal Policy Optimization** as our Actor-Critic method. An overview of it is given in the following figure:

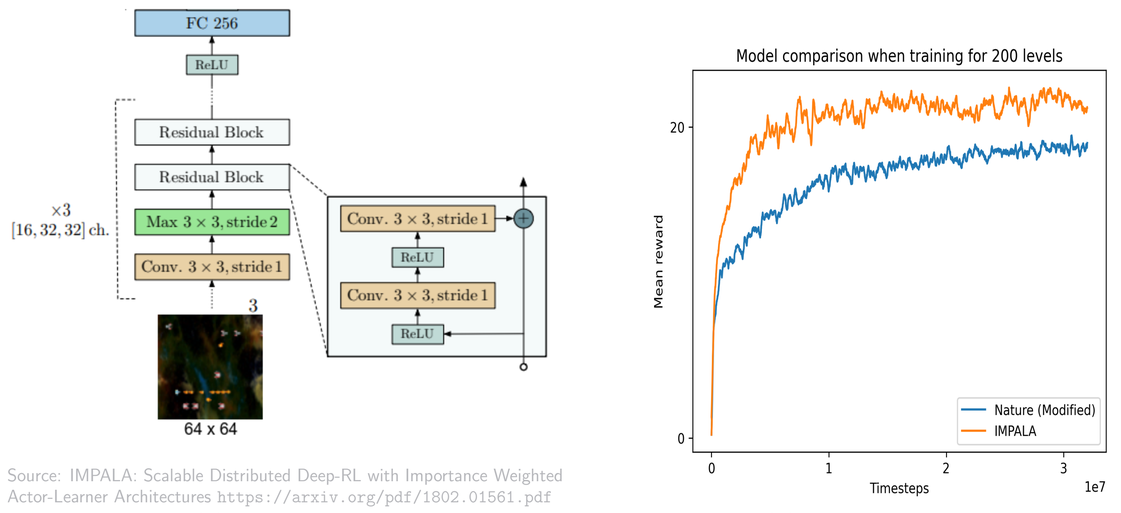


Source: Federated Reinforcement Learning for Training Control Policies on Multiple IoT Devices - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-actor-critic-proximal-policy-optimization-Actor-Critic-PPO-algorithm-process_fig3_339651408 [accessed 5 Dec, 2021]

Fig. 2: Proximal Policy Optimization (PPO) algorithm process

Model

We experimented with two different models for the neural network used to compute the Value function in the PPO: `Impala` and `Nature` Convolutional Neural Network models. We found that a reduced `Impala` CNN outperforms `Nature`. The final model, as well as the comparison of its performance against the other model is illustrated in *Figure 3*.



Source: IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures `https://arxiv.org/pdf/1802.01561.pdf`

(a) Model structure *(b)* Model performance

Fig. 3: Model implementation and performance relative to `Nature` CNN

Improvements and training

We used **data augmentation** to improve the performance of our model. In particular, we trained our network on a grayscaled version of each frame. We also experimented with more complex techniques such as automatic data augmentation using UCB-DrAC method, which added some improvements. The performance of two networks with and without data augmentation (only with grayscaling) is presented in *Figure 4*.

Our choice of **hyperparameters** followed the examples and guidelines given in the suggested papers. We experimented with the values given in the papers and changed them when we noticed improvements over the paper baselines, although these changes are minor. When training on our local machines, we had to scale down some hyperparameters in order to fit in the RAM constraints.

Due to the limitations of our machine, we ultimately trained our final model on an HPC cluster. The training took on average 12 hours. For more complex models we could not go beyond 24 hours which is an HPC constraint for the running time of any process. To ensure we have the best model at the end, we overrode a local copy every time we obtained a better mean reward.

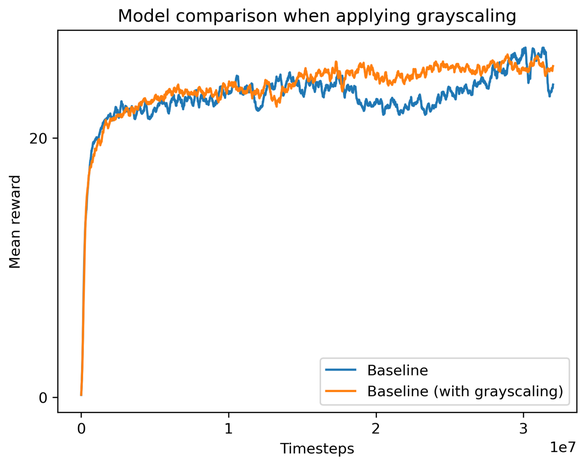


Fig. 4: Performance with Data Augmentation

Conclusions

In conclusion, we were able to develop an agent that is not only able to play at a nearly-human level on the non-deterministic environment on which it was trained, but also generalize to harder and relatively more complex environments (such as ones having a background or new types of enemies and obstacles). Given the hardware, as well as the time limitations (this project was developed in a second half of a 5 ECTS course) we are satisfied with our current results. This said, we consider there exists much space for improvement and express our further interest in the topic of RL and its applications in video games.