Deep Reinforcement Learning. Generalization in Video Games.

August Gammon Macholm
s165372
Technical University of Denmark

Natalia Agnieszka Karpowicz
s202400
Technical University of Denmark

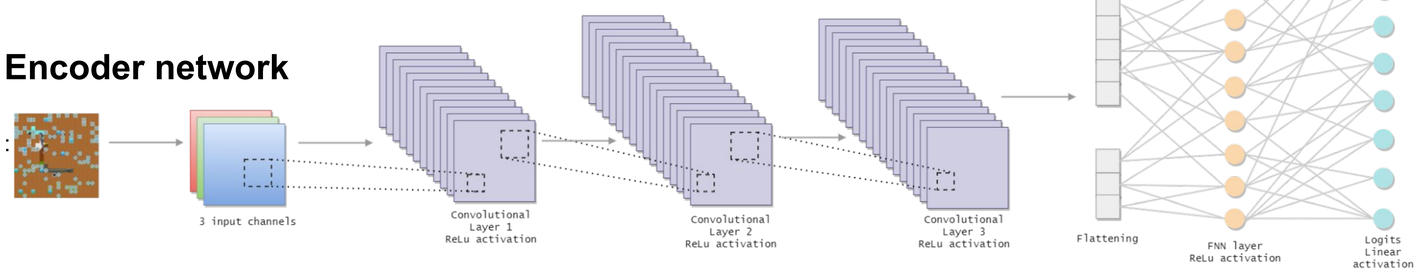
Introduction

The goal of the project is to teach the agent how to play multiple video games using PPO algorithm in the Procgen environment. The agent should be able run in many different environments by that simulate human behavior. People can adjust their thinking according to new situations but can machines do that? In this project we explored this problem to see if machines can learn to generalize.

Procgen Environment

Procgen Benchmark consists of 16 procedurally-generated environments (games). Each observation size is 64x64x3. The agent can choose between 15 possible actions at every step of the game. The Procgen Benchmark was especially designed to explore the problem of generalization in Deep Reinforcement Learning.

Encoder network



The diagram above shows the structure of the encoder's network. There are 3 input channels. The network consists of 3 convolutional layers.Data is then flattened and feedforward neural network layer is applied with 256 output features. The value and logits are calculated by applying the final FNN layers with one output for the value and 15 outputs for the logits.

Proximal Policy Optimization

PPO is a reinforcement learning algorithm, using the Actor-Critic method. The main advantages of PPO is that it simple to implement, data efficient, and robust. In PPO, we do gradient assent to optimize the following equation:

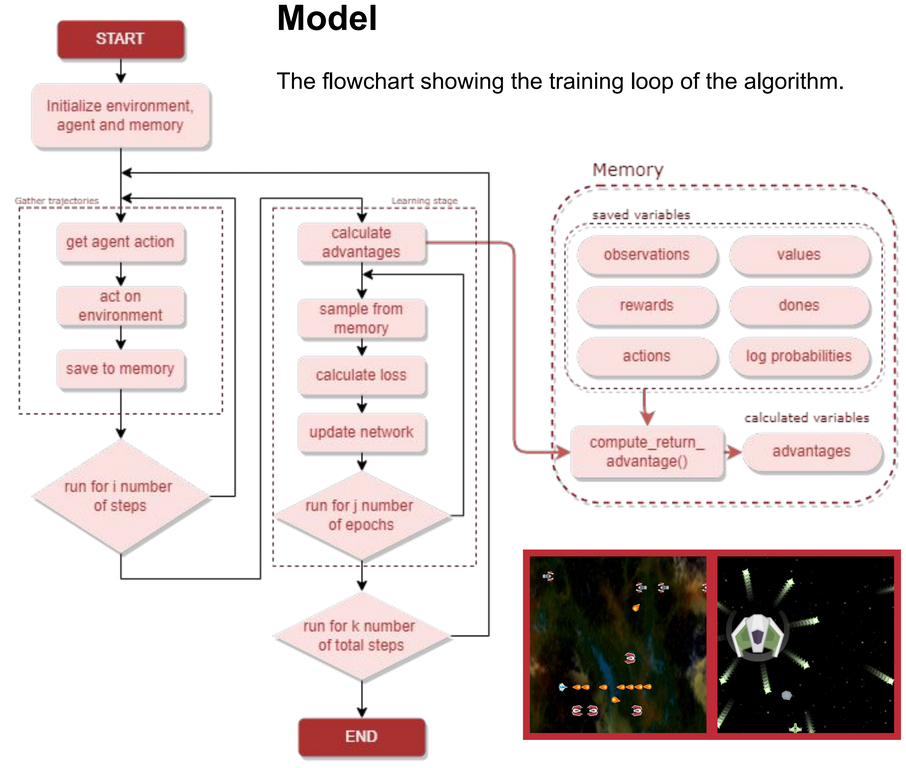
$$L_t^{CLIP+VF+S}(\theta) = \mathbb{E}[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

Where Lclip is our Clipped policy objective, Lvf is the squared error loss and s is an entropy bonus. The reason for optimizing the entropy, is to ensure exploration.

$$L_t^{CLIP}(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$
$$L_t^{VF}(\theta) = \frac{1}{2}\mathbb{E}[\max((V^{\pi_\theta}(s_t) - R_t)^2, (V_{CLIP}^{\pi_\theta}(s_t) - R_t)^2)]$$
$$V_{CLIP}^{\pi_\theta}(s_t) = clip(V^{\pi_\theta}(s_t) - V_t, -\epsilon, +\epsilon)]$$

Model

The flowchart showing the training loop of the algorithm.



Experiments

- Performing tests with the algorithm hyperparameters and network structure to improve the performance of the agent using single environment and by that see how hyperparameters influence the agent performance
- Test if the agent can play other games based on the network trained for single environment
- Comparing the results from training with one environment and with 2 environments to see how well the agent generalize

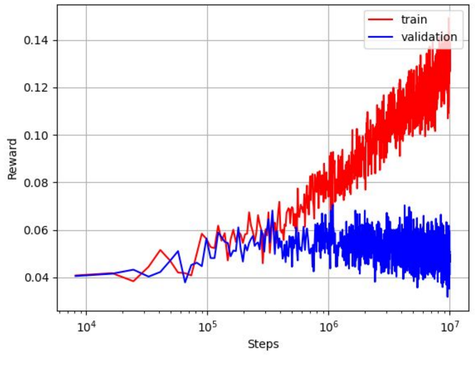
Results (summary)

The agent was trained on 2 games, 3 models were examined: using starpilot game, bossfight and training on both games. In all of the models, it was observed that the agent learned basic rules of the games, however, its performance could have been improved. Although, different hyperparameters were tested on starpilot model and the received result was satisfactory, they were not suitable for the bossfight model. When comparing, the first and third model, we can see that the agent learns slower while training on 2 games.

Results (graphs)

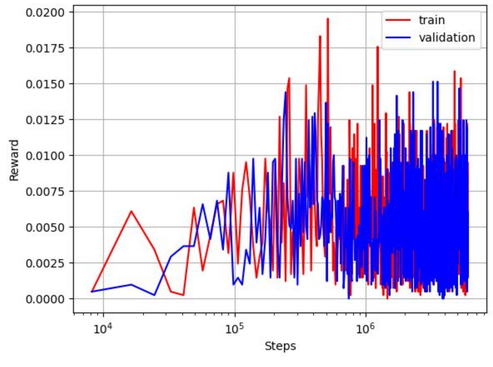
(StarPilot)

Performance of the model for StarPilot



(Bossfight)

Performance of the model for Bossfight



(StarPilot + BossFight)

Performance of the model for StarPilot and BossFight.