Reinforcement learning on playing Space Invaders and Breakout simultaneously

Strobel Maximilian
max.strobel@tum.de

Werhahn Maximilian
maxi.werhahn@gmx.de

Kiener Martin
martin.kiener@tum.de

Seferis Emmanouil
ga62cac@tum.de

Project Proposal

1. Introduction

We want to train an intelligent agent, who is capable of playing Space Invaders and Breakout simultaneously with the same inputs. During a first step we want to implement a deep Q-network (DQN) with convolutional layers to get above the score of random play for either of the two games. In the next step we want to gain average human performance by optimizing hyperparameters. Afterwards we attempt to optimize the networks architecture to perform on an above human level. Finally we want to let the agent play both games at the same time with the restriction of only being able to send the same inputs to both games. This is why we chose Breakout and Space Invaders, since the valid inputs are almost the same.

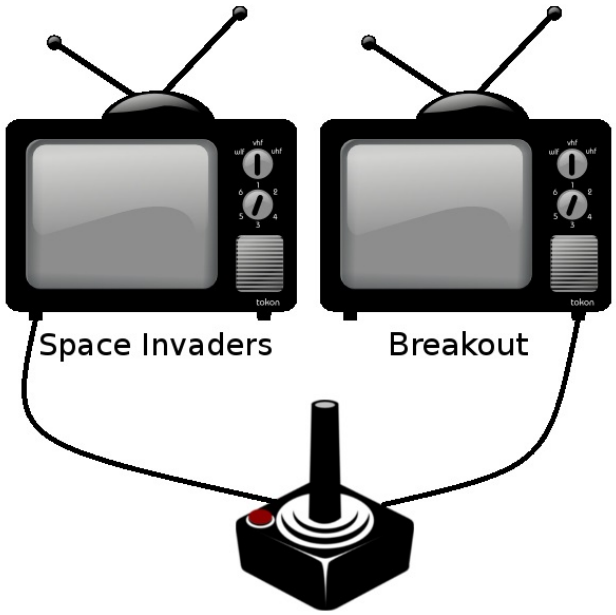

Figure 1. Space Invaders and Breakout played at once with identical inputs. The images of both screens are the inputs for the agent. Based on the inputs the agent takes an action, the input for both games, to increase the scores. The agent has to balance the benefits and drawbacks of its actions for both games.

1.1. Related Works

We were inspired by the paper "Human-level control through deep reinforcement learning" [**?**], which proposed an approach how to train convonlutional neural networks on classic Atari 2600 games. In the mentioned paper the authors were able to train an agent with skills classified higher than a professional human games tester. In the beginning we try to reproduce the results of this paper. After successfully implementing this architecture we combine two games together.

2. Dataset

For our project we decided to use OpenAI Gym as framework for the game. OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms implemented in Python. It allows us to gain information about the current state of the game in form of rewards and images. Instead of images it is also possible to use the memory of the current state of the game to directly access the positions of all entities in the game. It has also an API to control the agent in the selected environment.

Due to the fact that we use an reinforcement learning approach for our project, we do not need any labeled data and use the OpenAI Gym framework to gather inputs for our neural network. The input for the network is either current memory (states / positions / scores) of the game or images. The output on the other hand is an amount of actions taken (left / right / shoot). As score function we will use the rewards, that we can access from the Open AI Gym API.

3. Methodology

3.1. Architecture

As already mentioned in the introduction we try to implement the already existing architecture of a DQN. After the performance evaluation of a DQN, we try to optimize this architecture to improve the results by adding BatchNorm or MaxPool layers, since the original approach did only use Conv, ReLU and FC layers. Then two things could be done for playing two games at once; Either we train one network on both games simultaneously or we try to train two different networks, each for one of the games

and then have another way of selecting which input-action to take. This could be done by either taking the average or the argmax of the outputs of the networks.

As framework for our implementation we use PyTorch.

3.2. Training

At first we do not use pretrained networks, but train the network from scratch. During the training we might want to use a population based approach [**?**] for hyperparameter optimization. With population based training we try to accelerate the training and also gain performance benefits. For comparisions between different architectures we use cross validation.

If training takes too much time we could use pretrained networks for the combination of the two games and just train the network combining both outputs.

3.3. Ressource management

In our project we want to use the Google Cloud Platform and also the machines of the lab for a faster training on GPUs. For development and document sharing we considered to use GitHub.

3.4. Division of labour

The implementation of our project is subdivided in the tasks integration of OpenAI, network architecture, training / optimization methods and evaluation / visualization of the outcomes.

We record our progress continuously and work on the drafts for the poster and the paper in teams of two. The final work on the poster and the paper we will do with the entire team.

4. Outcome

We try to build an agent which is capable of playing Breakout and Space Invaders at once with one shared input and achieves a decent score. We evaluate our own network performance and compare it with the single game results from the paper and other existing implementations, which are also provided by OpenAI Gym. The resulting network will probably perform worse than current state-of-the-art networks, since it depends on two games at the same time.