DL4CV Project

Strobel Maximilian[1], Werhahn Maximilian[1], Kiener Martin[1], and Seferis Emmanouil[1]

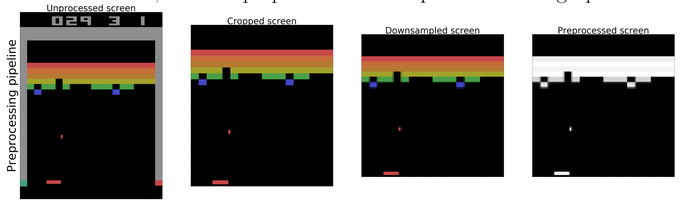
[1]Technical University of Munich

Our Idea

For this project we tried adapting already existing methods of reinforcement learning for Atari 2600 games to train a new architecture to play two Atari games simultaniously only using the same action for both. Before coming up with a new architecture we first trained a Deep-Q-Learning neural network on multiple single games. For the game Breakout we achieved above human-performance with this trained model which was our first goal of this project. After finishing this task we tried various new architectures to reach our second goal; to perform better than random play on two Atari games concurrently with the same action sequence.

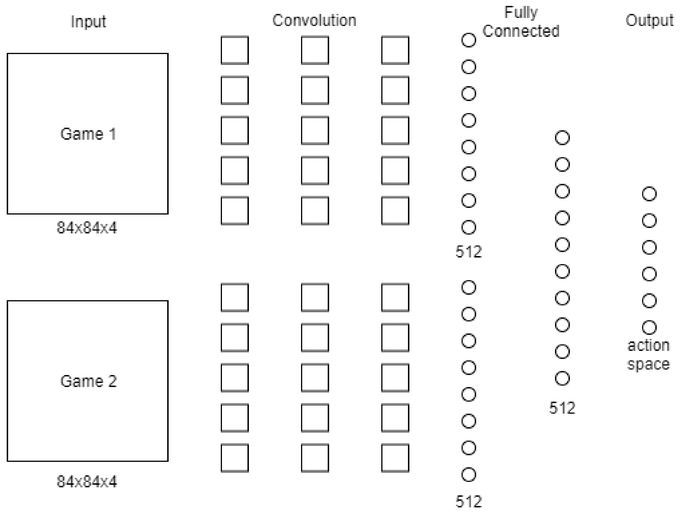
Preprocessing & Environment

The environment for the learning agents was OpenAI gym. This toolkit for developing and comparing reinforcement learning agents was used as an interface to Atari 2600 games like Breakout or SpaceInvaders, that the agents tried to learn. The interface provides informations about the state of the games as well as the current screen. Due to an internal flickering of the games (e.g. shoots are only on odd frames visible), the current screen is obtained by a maximum operation on two successive frames. Before the screen was fed into the neural network with the screen, it was preprocessed as depicted in the graphic below.



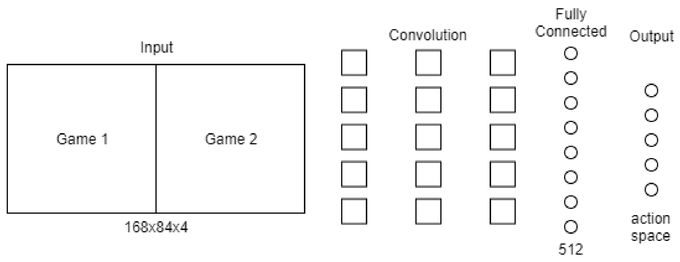
In OpenAI gym there exists a frame skipping technique, that was also modified. Instead of skipping randomly between two and four frames, the frame skipping rate was set to a constant value. During the training the rewards were clamped to -1 and 1 to gain comparability between different games. Additional the loss of one live was punished with a reward of -1, whatever the real reward was.

First Architecture



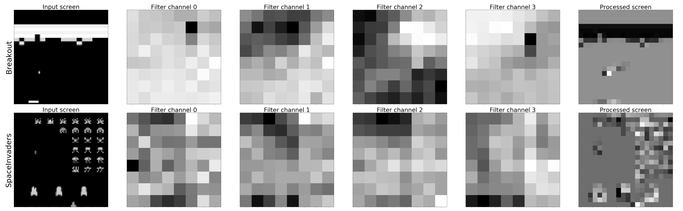
For the first architecture we duplicated the already existing DQN model, removed the output layers of both subnets and added two additional fully connected layers of size 512 and the maximum of the action spaces of the games to map the input screens of both games to one single action. Each of the two preprocessed frame histories is fed into one of the subnets separately. The resulting output action is then again mapped onto a second action which is used for the other game. With this version we already achieved our first goal which is being better than random play.

Second Architecture

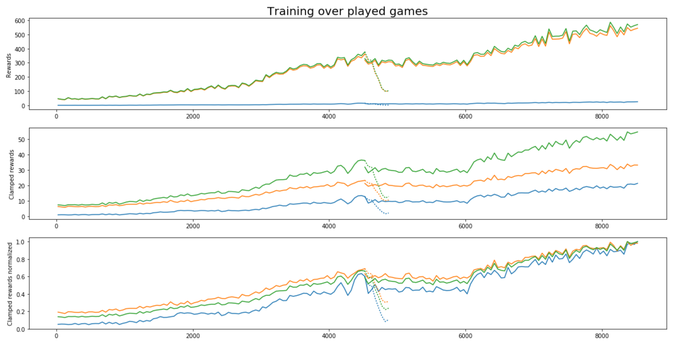


Another architecture we used to train both games at the same time was the typical DQN model. As input the frames of both games were concatenated into a single tensor. The dimensions of the torch tensor as input for the DQN is [1,4,168,84]. Furthermore the rewards of both environments are combined. The results for this architecture were way worse than for the first architecture. (Possible reasons?) (More details about the rewards etc.?)

Visualization



Results



Games	Random Play	Double DQN	Single DQN
Breakout + SpaceInvaders	(6.9, 40.3)	(,)	(,)
Phoenix + SpaceInvaders	(24.9, 640.5)	(46.6, 1877.8)	(,)

The first element of the tuple is the clamped reward which was used for training, second one the actual, unclamped reward of both games. The random rewards were averaged over 20000 episodes while the rewards for