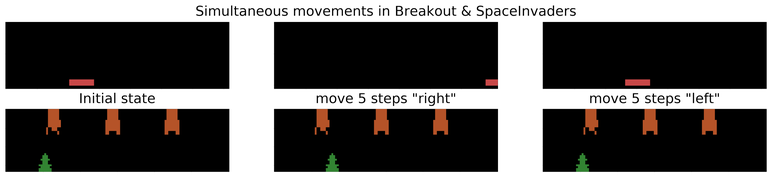
Twin DQN - Two tasks, one net

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

[1]Technical University of Munich

Our Idea

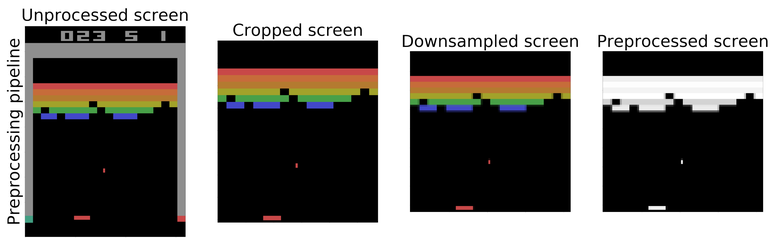
For this project we tried adapting already existing methods of reinforcement learning for Atari 2600 games to train a new architecture to play two Atari games simultaniously; only using the same action for both. Before coming up with a new architecture we first trained a Deep-Q-Learning neural network 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.



Check out our GitHub repository: **https://github.com/ms828/TwinDQN**

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.

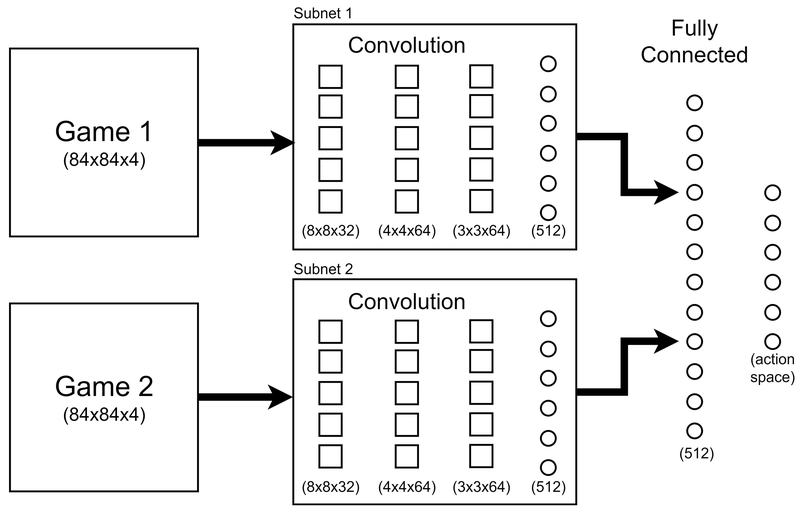


There is a frame skipping technique in the OpenAI framework, that was also modified. Instead of skipping randomly between two and four frames, the frame skipping rate was set to a constant value. During the training the rewards were clamped to -1 and 1 to gain comparability between different games. Additional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

Single Game

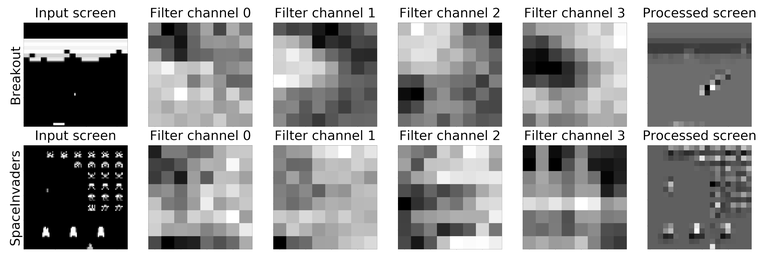
For a single game we tried to implement the architecture proposed in the paper "Human-level control through deep reinforcement learning"[1] which is the basis of our developed model for two games. For training we used a replay memory where we saved the frames of the game and the corresponding rewards, actions and the state of the game(running or done). Besides that we implemented a target network and the Adam solver instead of the RMSprop variant proposed in the paper to optimize our network. As loss function we used the smooth L1-loss and we clamped the rewards and the gradients between -1 and 1 to let the agent be able to learn multiple games with the same parameters.

Proposed Architecture



For our working 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. The actions for the game with the smaller action space were obtained by a mapping function, that adpats simliar actions. Each of the two preprocessed frame histories is fed into one of the subnets separately. The outcome of the two subnets is finally unified by the fully connected layers, that compute the Q-values for all possible actions.
With this version we already achieved our first goal which is being better than random play and it also achieves the best scores of all the architectures we tried out. As another model we implemented the same network we used for a single game, but we adapted the input by concatenating the frames of the two games. The other version we tried was stacking the screens in the channel dimension, instead of concatenating the frames along the width. These two approaches did not yield any good results.

Visualization

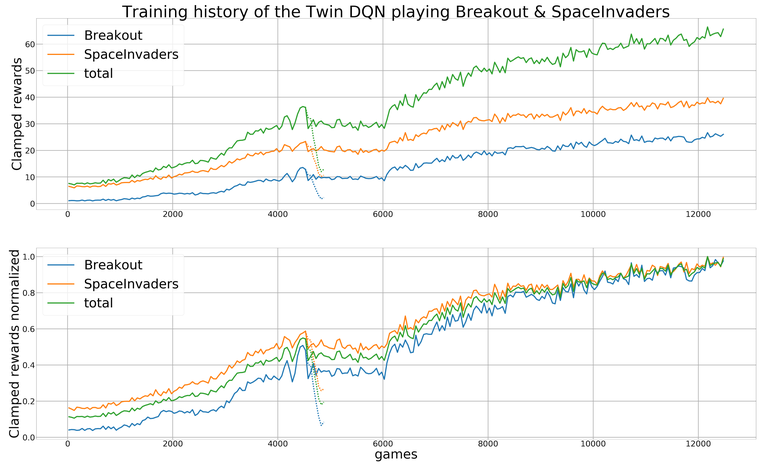


The above pictures show the convolution filters of the first layer of the subnets and screens processed by them. Those filters are used to determine for example, the velocity and the direction of motion of the ball in Breakout.

Results

Games	Total	Game 1	Game 2
Breakout + SpaceInvaders (Random)	(6.9, 40.3)	(0.9, 0.9)	(6.0, 39.4)
Breakout + SpaceInvaders (Twin DQN)	(67.8, 697.0)	(27.1, 34.5)	(40.6, 662.5)
Phoenix + SpaceInvaders (Random)	(24.9, 640.5)	(9.4, 159.1)	(15.4, 481.4)
Phoenix + SpaceInvaders (Twin DQN)	(57.2, 2579.5)	(29.2, 487.6)	(28.0, 2091.8)

The first element of the tuple is the clamped reward which was used for training, second one the actual, unclamped reward of the games. The random rewards were averaged over 20000 episodes, while we evaluated the Twin DQN on 1000 episodes with a frameskip of 4 and a probability of 0.01 of taking a random action instead of the action selected by the network.



[1]: Mnih et al. 2015, Human-level control through deep reinforcement learning