Hex Game – Group P

Our Goal: Solving the Hex Problem with Reinforcement Learning Techniques

## What is Hex?

Ein Bild, das Screenshot, Muster, Grafiken, Kunst enthält.

KI-generierte Inhalte können fehlerhaft sein.

<https://en.wikipedia.org/wiki/Hex_(board_game)>

Hex is a game that is played by two players. The goal is to create a connected line from the one side to the other. The opponent that achieves that goal wins the game.  
Usually the game is played on a 11x11 rhombus board. Other board formations like 3x3, 5x5, 7x7 will be covered in this paper.

## Ideas

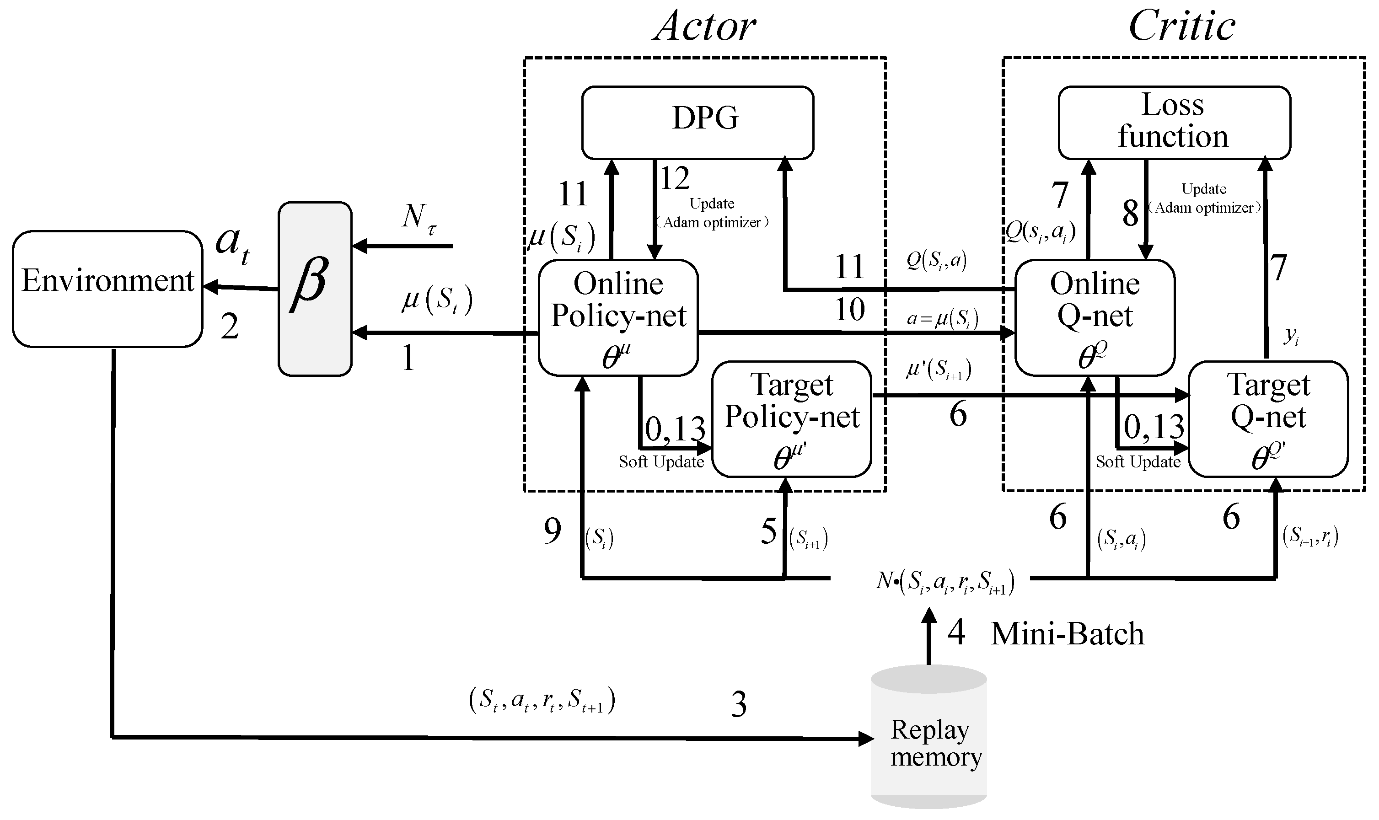
We brainstormed some ideas as a group. Following Algorithms were written down:

* DDPG
* DQN/Double DQN
* Actor Critic
* Advantage Actor Critic (A2C)
* A3C
* Genetic Algorithms xd

We rooted for DDPG for a long time and it was our final decision over the rest.

## First Steps

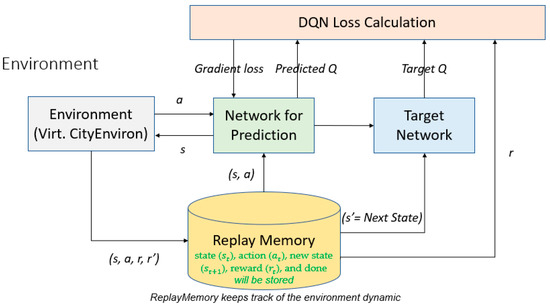
After we all understood the baseline-agents and also how the class HexPosition works. We got to work. The DDPG-Algorithm isn’t the simplest one out there. (see following figure)



<https://www.mdpi.com/2076-3417/12/19/9703>

**Terminator agent ?!**

So, we tried to start “small” with a Double DQN with Replay Buffer. Someone could say it is a DDPG without the critic, and a discrete action space.



<https://www.mdpi.com/2076-3417/12/7/3220>

We had problems with convergence of the dqn

Tried to finetune the Policy net architecture. From MLP to CNN with MLP head

We also played around with soft-update, learning rate, epoch time and reward calculations.

The Problem the Loss was going down but the win rate against random agent for example decreased

Ein Bild, das Text, Diagramm, Reihe, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.  
DQN Agent on a 5x5 field against random agent

After playing around with other hyperparameters we started getting very frustrated. Our next logical step was to add a critic. We hoped that this could increase the convergence and we would get better performance (higher win-rate)

We also saw a very nice paper that tried a policy critic model to achieve beating the hex game.

@INPROCEEDINGS{8408296,

author={Lu, Mengxuan and Li, Xuejun},

booktitle={2018 Chinese Control And Decision Conference (CCDC)},

title={Deep reinforcement learning policy in Hex game system},

year={2018},

volume={},

number={},

pages={6623-6626},

keywords={Games;Neural networks;Training;Machine learning;Computer architecture;Learning (artificial intelligence);Law;Computer Game;Hex Game;Deep Reinforcement Learning;Actor-Critic;A3C},

doi={10.1109/CCDC.2018.8408296}}

After we added the critic network we sadly didn’t converge either. We used a form of A2C (Advantage Actor-Critic). Note that we just added Adventage and Critic to our DQN-Agent. That caused some problem (we discuss later).

We tried to play around with entropy, advantage normalization, gradient clipping, and also feature engineering.

We splited the board (state) into 3 channels:

1. Player tokens
2. Opponent tokens
3. Legal turns

We also created a new model that was similar to the Alpha zero net

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

* Shared Backbone:
  + CNN with 3 residual Blocks (Conv2D (3→128) + BatchNorm + ReLU -> 3 x Residual Block)
  + Policy, Value Head

After that we still didn’t converge at all!

After the meeting where we discussed in groups we got pointed out for using a replay buffer

That was problematic because A2C assumes on-policy learning, by going off-policy we hurt convergence.

After removing the replaybuffer and rewrite our train function. We finally saw some sort of convergence. But it was very little. We saw a win rate of 60-70% against the random agent on a 5x5

We also noticed that a big batch size seems to stabilize the gradient. In the end we trained with a batch size of 1024 moves and 2048 games played for each epoch. That worked pretty well.

The biggest breakthrough we had was adding shaped rewards. We used Dijkstra pathfinding algorithm that compares the shortest path from state and next\_state, if it got lower we add a little extra reward. That was a very big thing!

After adding the the shaped reward and removing the replay buffer we finally saw some numbers. We beat random agent against random\_agent 80/100 times on a 7x7 board.

We didn’t stop there tho. We added a enemy blender (multiple baseline agents <list of baseline agents and description>), played around with the model size and tweaked some hyperparameters like games played per epoch. Also we added a learning rate scheduler to help with converging (We assume that it does not really helped much) because the win rate still oscillates quite a lot in the end even though the learning rate was at 1e-6.

In the end we also had the problem that our model only can play white (means only one side of the board). To fix that we just added a hyperparameter that switched the feature preparation. We also had to change how the self\_play\_data\_generation worked.

The big limitation we faced with the split approach is that playing against itself was not possible, because for each training cycle it can only play one side. That would have been our next big step to make it multi sided. We sadly ran out of time to do that.

In the end we achieved a very robust model. It beated random\_agent 95% of the time on a 7x7!

We also played in human against the model and it was quite stupid but still we could witness some strategy, which made us very proud.

In the end we sadly lost very early in the ternament. If we would have added the selfplay and a lot more training time I think we could have won our cohort :-)