Exercise 3: CNN for maze

The network we used to solve the task consisted of two times two Convolutional layers with a MaxPool at the end. In the first two convolutional layer we used 32 3x3 filter with padding to the same size and a pool size of 2. The other convolutional layers had the same size and padding but used 64 filters. All of them had a ReLu activation function and there were 25% Dropouts after the pooling layers. Afterwards there were four dense layers with 128 units each, also activated by ReLu. The output layer used was a Softmax layer.
In almost all cases the agent performed just as well as the A* data and took the shortest path, only in a few cases it didn't find the way and started to oscillate.

Experiment 1 - Adding more history

Increasing the history doesn't change the performance of the network a lot as it's already pretty good. However, it increases the performance and shortens the agent's path further (e.g. eliminates small detours).

Experiment 2 - Changing the target location

If we change the target more than one or two steps the agent can't find the target at all. If we only move it one step though, the agent can reach it in most cases. That's, of course, because we train the agent on the specific path to the target from which it won't deviate much.

Experiment 3 - Changing the map

Changing the map can make a huge difference or none at all, depending on if the change blocks the A* path the agent wants to take. In that case, it usually doesn't generalize enough to find a new way to the target.

Methods for generalization

We tried less training, less complex networks (less hidden units) and also less history, but in all cases the general performance wasn't good and the generalization didn't improve as we thought. We think the best way to build a generalizing network would be to use Reinforcement Learning, as there we don't just learn the path but an adaptable method to find the target.