Using RL to Sustain High-density Populations in Conway's Game of Life

(Dated: January 24, 2021)

Abstract: In this project we introduce a reinforcement learning agent in Conway's Game of Life with the objective to sustain long-living, high-density populations. We experiment by proposing a subdivided variant of the environment, alongside the standard tabular method problem definition. We find that the agent creates highly correlated, static structures of living cells. We observe fast learning rate for small grid sizes. Finally, we approach the task with a deep Q-network and observe similar results.

I. INTRODUCTION:

Cellular automata are discrete, local dynamical systems, which since their discovery have found application in computability theory, physics, and various areas. A two-dimensional cellular automaton is based on a grid of cells, each of which can be in a finite number states at a given point in time - for example ”live” or ”dead”, or ”1” or ”0”. For each cell, its state at the next time step depends on the states of its neighbours and some fixed rules. The initial configuration of the grid is given by the user.

The “Game of Life”, despite its name, is a zero-player game. It was created in 1970 by the mathematician John Horton Conway as a two-dimensional cellular automaton. Since then, it has inspired many simulations of physical and biological systems due to its analogies with the rise, fall and alterations of a population of living organisms. [1] In the original model of ”Game of Life” (GoL) cells can be either ”live” or ”dead” at a given time step. The game evolves according to the following simple rules:

- Any live cell with two or three live neighbours survives, otherwise it dies.
- Any dead cell with exactly three live neighbours becomes a live cell.

A wide variety of interesting spatial behavior can emerge in course of evolution from an arbitrary initial state in “Game of life”. The most popular structures are:

- ”Still lifes” - stable patterns, which do not change once they appear.
- ”Oscillators” - patterns with periodic behaviour.
- ”Spaceships” - self-localized patterns which translate across the grid.

Our goal in this project is to introduce a player into the game with the objectives to learn the rules of the automaton and preserve as much life as possible. This allows us to study the shift in the natural evolution of a system which obeys the rules of ”Game of Life”, if a small mutation takes place at each time step. A goal similar to ours has been pursued by [2] and [5].

II. METHODS:

Reinforcement learning in ”Game of Life”: We introduce a reinforcement learning (RL) agent which interacts with the game. The environment is a finite two-dimensional grid of size $n \times n$ with periodic boundary conditions on its sides, evolving by the rules of GoL. The state at a given point in time is the configuration of live and dead cells in the environment. The state space in our framework is $\#S = 2^{n\times n}$. We allow the agent to revive one cell in each generation before the natural evolution. Thus the action space for each state is $\#A = n^2$. If a live cell is selected we ignore the action. The reward is defined as the density of living cells in each generation after evolution:

$$R = \sum_{i=1}^{n^2} x_i \qquad \text{where} \quad x_i = \begin{cases} 1, & \text{if the } i^{th} \text{ cell is live} \\ 0, & \text{otherwise} \end{cases}$$

The game can end in two different ways: when there are no live cells, or when the number of generations has exceeded its maximum set value.

Algorithms: For our purposes we use Q-Learning [4], a reinforcement learning algorithm that can be applied to sequential tasks. It is an off-policy algorithm since the Q-Learning function uses a behaviour policy, which is exploratory in nature, to learn the policy that maximizes the expected value of the total reward starting from the current state. To learn the state-action value function $Q(S, A)$ we apply the following update rule iteratively at each time step:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_t + \gamma max_a Q(S_{t+1}, A') - Q(S_t, A_t)] \quad (1)$$

However, tabular methods are very limited with respect to the state space. For problems with larger state spaces Deep RL algorithms are more suitable. Deep Q-Learning trains a function approximator to learn the state-action value function. This approximator is a deep neural network, with parameters θ. Since the states are grids and the update rules are local i.e. 3×3 matrix is used to determine the state of the center cell, we can model GoL with a convolutional neural network.

Approaches: We use the following three approaches for our task:

1. **Q-learning on a small grid.** In this method the agent is allowed to act on any cell of the grid. In tabular methods we need to restrict the size of our grid to up to 20 cells total. To avoid this restriction we use a dynamic structure instead of 2D array for containing the action-value pairs. This optimization allows us to run the method on grids with up to to 50 cells.

2. **Q-learning on a large subdivided grid.** In order to circumvent the limitation from 1. we try to divide a larger grid into smaller regions of a size feasible for tabular Q-Learning. Now the agent also has to choose in which subregion to act. The action space expands by the number of subregions, but the state space and Q-table dimensions stay the same. We break the action down in two parts: epsilon-greedy selection of a subregion and epsilon-greedy selection of action as in 1. The downside of this method is that to the agent the subregions are not connected and their evolution is now non-deterministic.

3. **Deep Convolutional Q-Learning.** To be able to learn on all grid sizes, we use deep Q-learning. The action space is the same as in 1. Our network consists of one convolutional layer, followed by one or two hidden dense layers. We collect our transitions inside a buffer. For calculating the loss we use a target deep network.

Statistical analysis: All approaches are tested first for a fixed initial state and then for a random one. To evaluate the performance of the methods we graph the rewards as a function of game evolution. In order to understand the agent's behaviour we visualize the games.
We inspire from Schulman and Seiden [3] and implent their proposed variant of entropy for the Game of Life. A system that begins randomly populated will, under the propagation rules of the Conway game, develop structure and correlations. One way to measure the increasing order in the system is by using a quantity analogous to entropy in statistical mechanics. The general idea is to subdivide the system and measure the extent to which living cells do or do not cluster.
A $n \times n$ board is broken into subregions of size $j \times j$. Coarse-grained description of the state of the system is provided by the sequence $m_i \in [1, (n/j)^2]$, where m_i is the number of live cells in the i-th subregion. The entropy associated with this coarse-grained description is given by the logarithm of the total number of microscopic states that can be associated with the given sequence. Thus

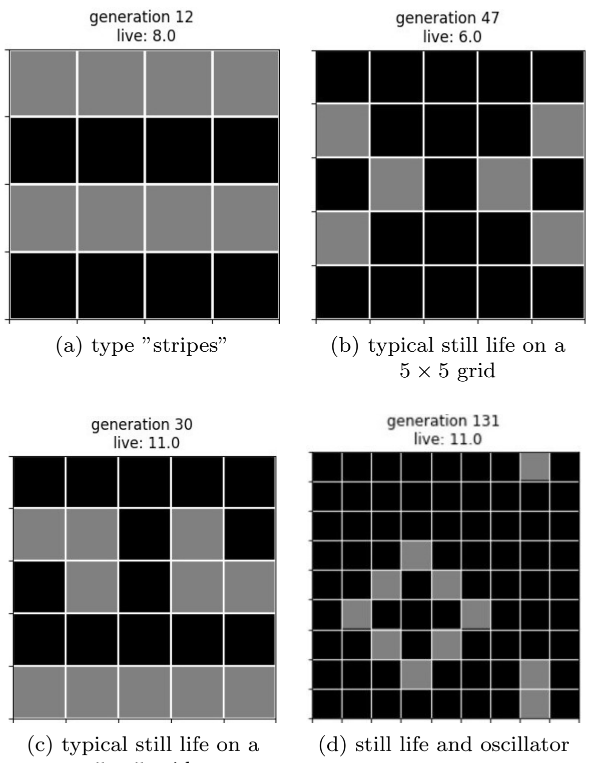
$$S = \frac{1}{n^2} log \prod_{i=1}^{(n/j)^2} \binom{j^2}{m_i} \tag{2}$$

Such defined, the entropy is strongly related to the density so we will normalize it by dividing by that of a random board with the same density. Normalized entropy is equal to one for boards with randomly placed live cells. The more ordered the structures on the board are, the lower the normalized entropy will be.

III. RESULTS:

We have implemented each of the above approaches and obtained the following results.

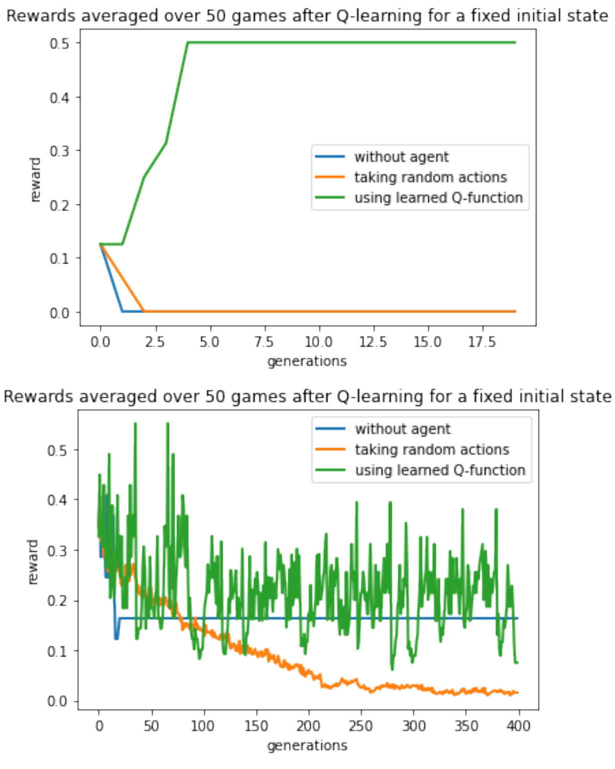
Q-learning on a small grid: We trained the model on square grids of size up to 7×7. For all grids we observe similar behaviour when training on fixed and on random initial states.
For grid sizes 4×4 the learning rate is fast. The optimal solution achieved from many different initial states is a still life of the type stripes (See fig. 1). For highly populated or underpopulated initial states dying off is inevitable in the first few steps.
For grid sizes 5×5 and 6×6 we continue to arrive at various still life forms. However, they are achieved in a different number of steps and have density of around 0.2. We also observe a few oscillators. There are still some initial states that die off but not as fast as without an agent.



(a) type "stripes" (b) typical still life on a 5×5 grid (c) typical still life on a 5×5 grid (d) still life and oscillator

FIG. 1: Still life

For 7×7 grids we rarely observe still life. The agent sustains the game for long periods of time. The



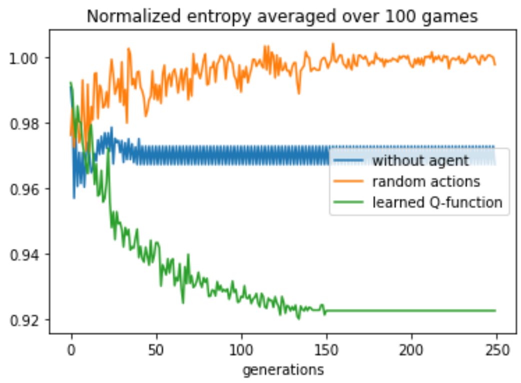


FIG. 2: Comparison of averaged rewards without an agent, with a random agent and using learned for a fixed initial state Q-function for a 4×4 grid (*top*), a 7×7 grid (*middle*). *Bottom:* Normalized entropy for a 6×6 grid.

evolution is non-periodic, alternating between high and low density states. (See fig. 2).
We compare the normalized entropy of states created by our agent with the entropy of the natural evolution and the random actions. Taking random actions completely erases the order that emerges. That is why the entropy with random actions quickly goes to one. Contrary to this behaviour, games with agent tend to have low entropy. Still lifes are by nature ordered structures. However, even when the agent can not lead the game to a still life it still creates structures of cells which are highly correlated. (See fig. 2, bottom)

Q-learning on a subdivided grid: We run the method for subregion sizes up to 8×8 . When trained on the whole state space the agent behaves worse than the random action evolution, similar to the natural evolution. Training with a fixed seed leads to a slightly better behaviour in the first few steps, progressing identically to random evolution.

Deep Convolutional Q-Learning: For a 4×4 grid and fixed seed the results are the same as those with tabular methods. The model converges fast to still life of type stripes. When training with a random initial state we still observe the same solution, with other forms

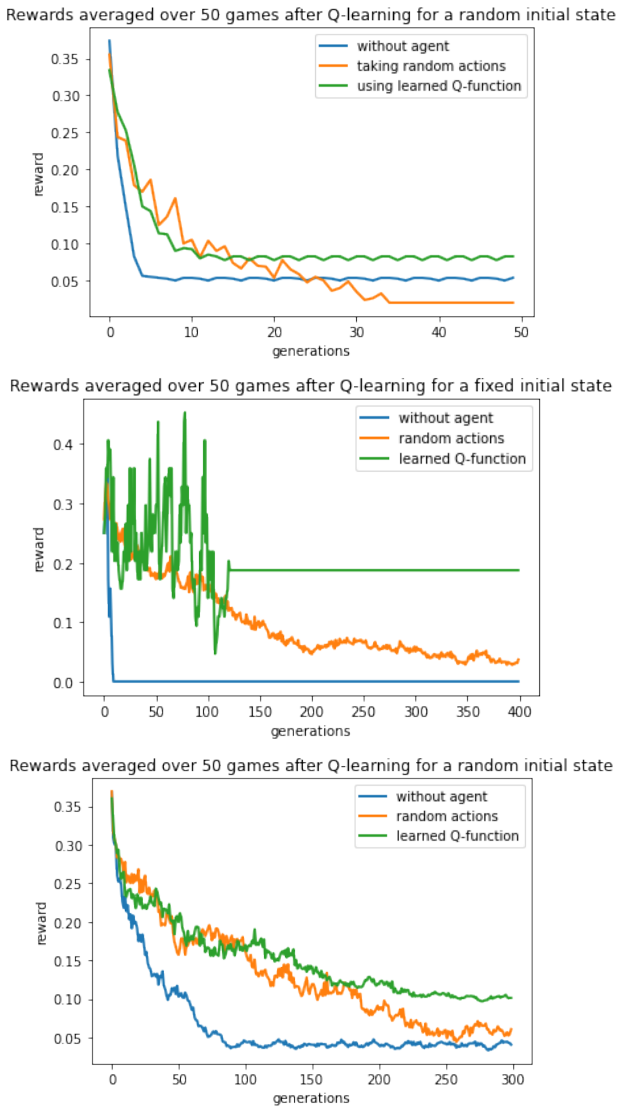


FIG. 3: Comparison of averaged rewards without an agent, with a random agent and using learned Q-function for a 4×4 grid (*top*), a 8×8 grid and fixed initial state (*middle*), and 8×8 grid and random initial state (*bottom*).

of still life occurring occasionally. However, many games die off quickly, which brings the average down and the overall performance is not as good as the corresponding tabular methods.
For larger grids with fixed seed learning of still life is achieved, but after more steps (fig. 3, middle). Learning is significantly slower without a fixed seed. After training for a feasible time, only slight improvement over random actions is registered (fig. 3, bottom).

IV. CONCLUSION:

In this project we propose a few strategies to introduce a RL agent in the Game of Life by Conway. They were able to produce different variants of still lifes and oscillators. We find that the standard tabular method is robust for this problem, but limited. Our modified approach does not achieve better results than the standard one, which is probably due to the introduced uncertainty. Our final method is deep Q-learning. When implementing it we frequently varied the hyper-parameters and the architecture of the network. We managed to obtain good results, but the learning was not very stable and we could not improve on the tabular methods. As to our knowledge it is yet to be proven that RL can find an optimal solution for arbitrarily large grids. Our studies may be improved by allowing the agent to change more than one cell per step.

Acknowledgements: We would like to acknowledge that the computational work reported on in this paper was partly performed on Google Colaboratory.

[1] Lorena Caballero, C Germinal, and H Sergio. Game of life: simple interactions ecology. *Mariana Benítez, Octavio Miramontes & Alfonso Valiente-Banuet. "Frontiers in Ecology, Evolution and Complexity."(Mexico: CopIt ArXives). TS0012EN*, 2014.

[2] Simon M Lucas, Alexander Dockhorn, Vanessa Volz, Chris Bamford, Raluca D Gaina, Ivan Bravi, Diego Perez-Liebana, Sanaz Mostaghim, and Rudolf Kruse. A local approach to forward model learning: Results on the game of life game. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

[3] LS Schulman and PE Seiden. Statistical mechanics of a dynamical system based on conway's game of life. *Journal of Statistical Physics*, 19(3):293–314, 1978.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 2017.

[5] Lucas Wilson. Conway's game of life controlled with reinforcement learning. *github*, 2018.