Towards Emergence of Grid Cells by Deep Reinforcement Learning

Aqeel Labash, Daniel Majoral, Martin Valgur

January 2018

1 Introduction

The fields of neuroscience and artificial intelligence (AI) have strong synergies. Neuroscience provides new ideas and models for AI and AI feeds back hypotheses about how and why the brain might perform some tasks. An important discovery in neuroscience was grid cells [2]. Grid cells are a kind of neurons that seem to code position in several species. Recently, researchers find emergence of these cells in neural networks [1]. They trained their network in a supervised manner, but can grid cells emerge with reinforcement learning?

1.1 Plan

- Update Artificial Primate Environment Simulator (APES) to fit our task.
- Design a static environment with an agent. The agent can see only the nearest surrounding blocks. The agent is rewarded for eating a randomly scattered food item and then goes back to the starting point (home) with the shortest possible route.
- Build a neural network with LSTM to perform the task.
- If the model succeeds then check whether grid cells emerged or not.

2 Background

2.1 Deep reinforcement learning

Reinforcement learning (RL) is a branch of artificial intelligence that allows an agent to learn by trial and error while interacting in an environment. In particular, the agent must learn to select the best action in each specific state to maximize its accumulated reward [7]. The idea behind learning by interacting with an environment is inspired from how human and animal infants learn from the rich cause-effect or action-consequence structure of the world [7, 8, 6]. Therefore, RL is a biological plausible method of learning certain associations

and behavior. To deal with the large number of states presents in a typical environment, neural networks can be used as agents.

2.2 Deep Q-Networks

Deep Q-Networks (DQN) is the result of combining deep learning with the Q-learning algorithm [11, 10], which is one of the main algorithms in RL [7]. The idea is to use neural networks to approximate the Q-value function. The algorithm proceeds by saving transitions (state s_t, action a_t, reward r_t, and the new state s_{t+1}) to be used later to train a network parametrized by a vector θ. Mnih et al. [4] used two networks (a training network parametrized by θ and a target network parametrized by θ^- which is updated at every time step according to $(\theta^- = \theta \times \tau + \theta^- \times (1 - \tau))$. In this case the cost function to be minimized by the training network is the distance between its prediction and the following target value:

$$y_t = \begin{cases} r_t, & \text{if its terminal state} \\ r_t + \lambda \max_{a'} Q(s_{t+1}, a'; \theta^-), & \text{otherwise}\,. \end{cases} \tag{1}$$

2.3 Double Deep Q-learning

Double Deep Q-Network (DDQN) was propsed to avoid an overestimation bias in Q-values [3]. DDQN chooses the action using the trained network but estimates the value using a target network as shown in Equation 2:

$$y_t^{DDQN} \equiv r_t + \lambda Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta); \theta^-)\,. \tag{2}$$

2.4 Grid cells

Grid cells as defined by [5] "A grid cell is a place-modulated neuron whose multiple firing locations define a periodic triangular array covering the entire available surface of an open two-dimensional environment". They where first discovered by [2] and thought of as a micro-structure map in the brain. Figure 1 shows the activation of one grid cell neuron in a rat brain in different places in two-dimensional space.

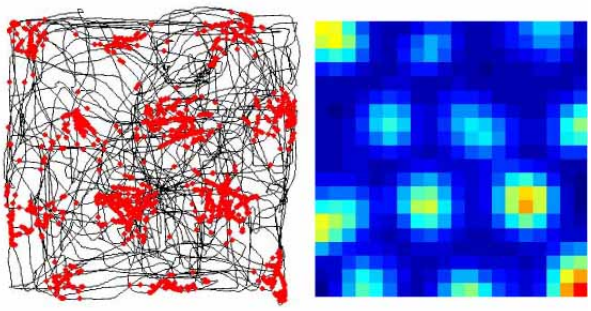

Figure 1: Left: Trajectory of one rat in a closed cage (black) with one of his grid cells spike locations superimposed (red). Right: Colour-coded map showing firing rate distribution for the same cell, from silent (blue) to peak rate(red). Figure taken without permission from Moser et al.[5].

3 Results

3.1 The task

In every episode the food spawns randomly in the upper half of the environment. The agent always starts from the same place, the bottom right corner. The agent receives a positive reward (0.5) for reaching the square where the food is (eating the food). If the food has been consumed when the agent reaches the starting position he receives another positive reward (1) and the episode finishes. The agent only sees his position and the eight squares around it. However the agent also receives as input his last action and whether the food has been eaten or not. The task is designed so the agent learns to estimate his location remembering his last actions. A good position estimation allows the agent to return home after eating the food in less steps, maximizing the reward.

3.2 Experiments

We start by studying the task's results in a small environment. Next we proceed to study the same task in a bigger environment. Finally we analyze the network activations and how they relate to the agent's position.

3.3 Small environment

First we tried with a small 7×7 environment (figure 2) and trained the network for 20k episodes. In the small environment the agent learned to locate the food and go back home in just a few steps. Figure 3 shows the progress of the agent during the training.

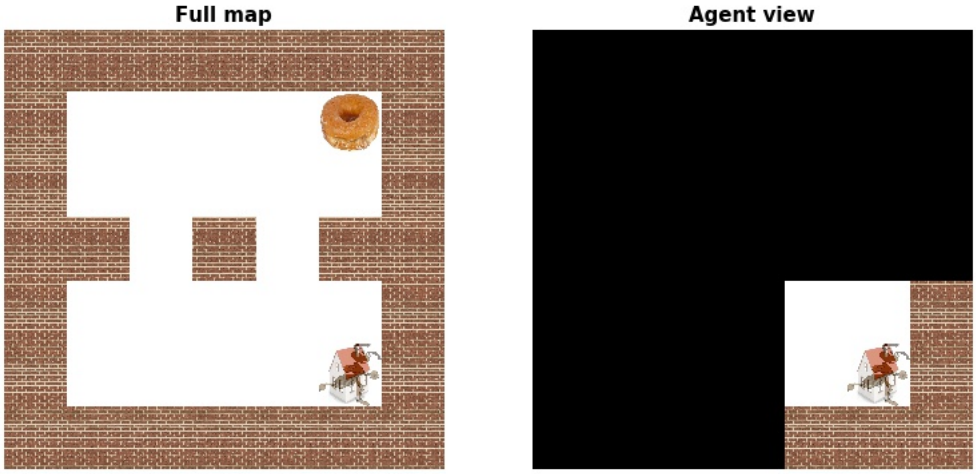

Figure 2: Small arena. Left: full map, right: only what network can see.

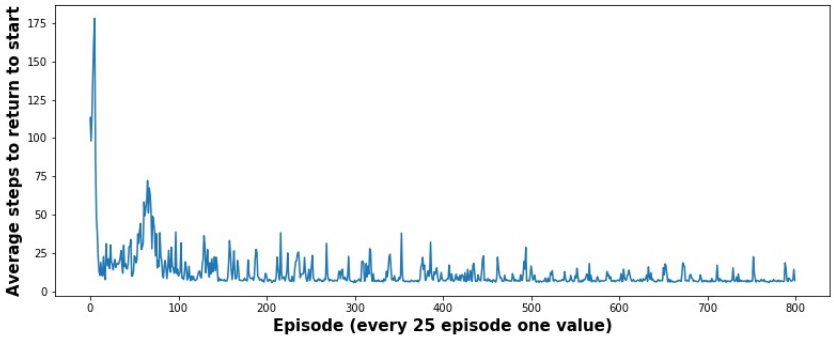

Figure 3: Steps taken by the agent to return home after eating food over the training in a 7×7 grid world.

3.4 Larger environment

To be able to visualize grid cells emergence we increased the size of the grid world to 11×11, see Figure 4. Although the number of states is larger, the agent also learns to perform well. Figure 3 shows the progress of the agent during the training. For more detail the reader can watch the agent performing the task over the training in https://youtu.be/ekIxRoz4lJI.

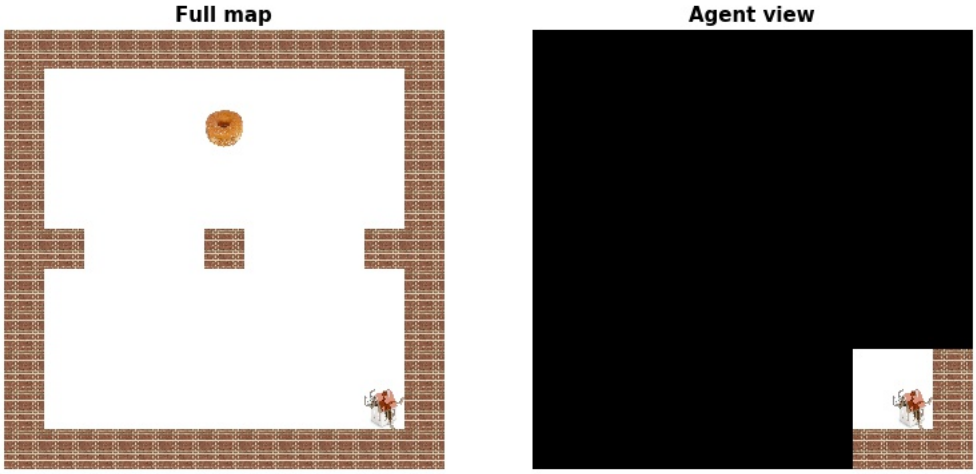

Figure 4: Large arena. Left: full map, right: only what network can see.

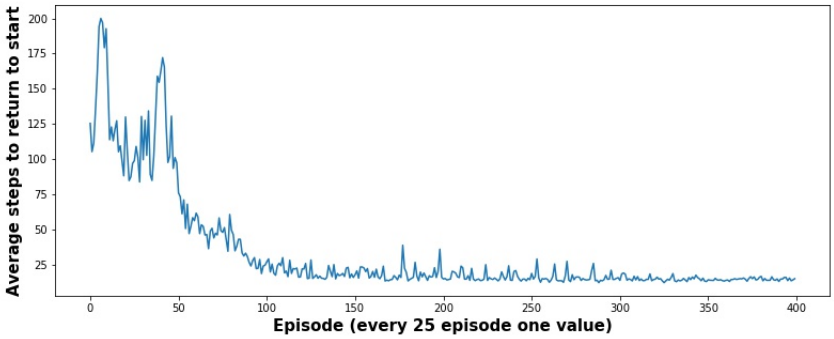

Figure 5: Steps taken by the agent to return home after eating the food over the training in a 11×11 grid world.

3.5 Activations

We have seen an agent has successfully learn the navigation task. Now we are interested in finding out how his neurons code for his position. For 10 episodes we let the agent perform a random walk while recording the LSTM layer activations at every step. We calculate the average activation for every neuron and for all the positions in the map (see Figure 6). We have inspected visually all the 290 hidden units. Around one third of the units do not show any dependence on location. Several neurons show a different activation near home than the rest of the map (See Figure 6 center right neuron). Some neurons show preference for the left or the right side or left side of the grid world (See Figure

6 bottom middle neuron). Others are more active at the center of the map (See Figure 6 top middle neuron). Although we find neurons similar to border cells or place cells, none of the cells can be qualified as a grid cell.

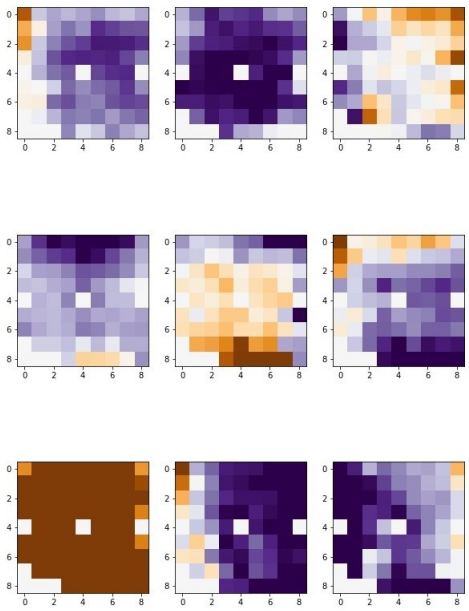

Figure 6: Selected examples of average activation for every square of hidden units from the LSTM cell in the 11X11 arena. In Purple positive values, brown negative ones

4 Discussion

We have trained an agent to perform a navigation task in two different environments: in a 7×7 grid world and in a 11×11 grid world. The network we have employed for the task is very simple, a dueling network with a convolutional layer and one LSTM. However, the agent learns to perform the task successfully. It is a significant accomplishment given that our agent at any given time has a very limited 3×3 vision of the environment. The successful navigation implies that the LSTM cell is able to integrate the actions of the agent, keeping track of the position. We have visually inspected the average activations of all the neurons of the LSTM layer for every location. However after it we conclude that no grid cells are present in our trained network.

In the future we plan to add different kinds of regularization to the network and increase the complexity of the environment. The final aim is to see if grid cells can emerge in deep reinforcement learning and to study under which conditions they do it.

5 Methods

All the simulations were performed in Python 3.6 and TensorFlow at the High Performance Computing Center of the University of Tartu.

5.1 Environment

We used Artificial Primate Environment Simulator `APES` to serve this project's aim. `APES` is a 2D grid world open source environment. It allows to shape the output (network input), reward, actions, and the spatial structure of the environment. We modified `APES`, allowing some elements to overlap others, otherwise the environment only allowed one element in each position. Particularly in the starting position the agent can overlap with Home.

5.2 Network

The network receives an image of size 3×3, which goes through a convolutional layer of 32 filters 3×3 with stride one and padding *same*. Then the output is flattened to generate a vector of size 288. This vector is concatenated with two new inputs. One contains whether the agent has eaten the food and has as possible values 1 (food eaten) and 0 (food not eaten). The other input contains the last action performed by the agent and has four possible values (0,1,2,3) which correspond to the four possible actions the agent can take. The resulting vector (size 290) is fed into an LSTM with 290 hidden units. Then, following the duelling network architecture [9], the output of the LSTM is divided to two streams: value and advantage. Finally, both streams are merged to generate the Q value (we use as advantage estimator the difference with the average advantage).

5.3 Training

The training of the network follows an epsilon-greedy policy. Before training there are 2000 steps (each step is one agent action) in which the agent acts completely randomly and no updates are made to the network. After this the probability of random action (ϵ) is fixed to one and decreased linearly until it reaches the value of 0.1 at the end of training. During this process the network is updated every five steps with eight episodes with full roll out. The replay memory stores up to 1000 episodes. When the maximum capacity is reached the oldest episodes are replaced. To calculate the Q values we employ Double Q-learning with τ of 0.001. The values for all parameters are provided in Table 1.

Table 1: **Parameters and applied values**

Training Parameters	
Parameter	**Value**
World size	7-11
γ	0.99
Episodes	20000
τ	0.001
ϵ	1 to 0.1
Batch size	8
Max steps episode	200
Pre-train steps	2000
Replay Memory size	1000 episodes
Update frequency	5 steps

6 Contributions

- Aqeel
  - Adapted APES environment to the task.
  - Corrected some bugs in main loop code
  - Worked on the report with Daniel.
  - Launched the experiments on the cluster.
  - Saved results of launched experiments.
  - Video generation.
  - Analyzing models (not activations).
- Daniel
  - Adapted the network to Gridcell task.
  - Writed the main loop code, with code from Aqeel and Axel tasks.
  - Built simulation to generate activation values.
  - Analyze the activations.
  - Writing report with Aqeel.
- Martin
  - Project discussion.
  - Watched the RL course by David Silver and studied relevant articles.
  - Experimented with increasing the grid world size.
  - Ran experiments on a local machine.
  - Writing report.

References

[1] Anonymous. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *International Conference on Learning Representations*, 2018.

[2] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

[3] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2094–2100. AAAI Press, 2016.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] E. Moser and M. Moser. Grid cells. *Scholarpedia*, 2(7):3394, 2007. revision #144456.

[6] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[7] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An.* A Bradford book. Bradford Book, 1998.

[8] Edward Lee Thorndike. *Animal intelligence: Experimental studies.* Macmillan, 1911.

[9] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.

[10] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[11] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards.* PhD thesis, University of Cambridge England, 1989.