Combining Semantic Modeling and Deep Reinforcement Learning for Autonomous Agents in Minecraft

Andrew Melnik, Lennart Bramlage, Hendric Voss, Federico Rossetto, Helge Ritter
CITEC, Bielefeld University
33619 Bielefeld, Germany
andrew.melnik.papers@gmail.com

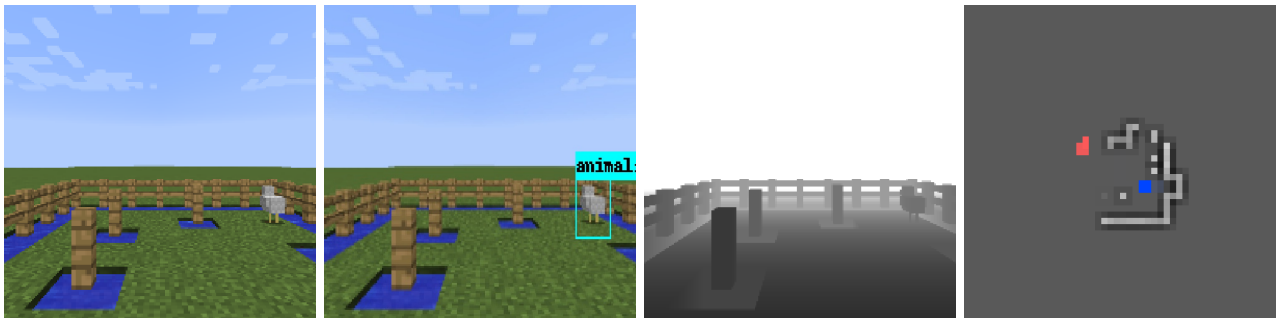

Figure 1: RGB observation, object detection, depth estimation, top-down map reconstruction.

1 Introduction

This work describes our extended solution [MARb] for the MARLÖ (Multi-Agent Reinforcement Learning in MalmÖ) Minecraft competition [MARa, PLHM+19, JHHM, GCH+19], and provides insights about how a semantic model of abstract representations of rich sensory data in an environment influence learning and performance in a task. Reinforcement learning can achieve state-of-the-art performance on a broad range of tasks [KMO+18, SM18], however, facing limitations in generalization and sample efficiency. Abstract latent representations and different levels of control can substantially increase sample efficiency [MFSR19]. Learning of a semantic model requires a structured representation of an environment. For that, a preprocessing pipeline can translate rich sensory information into useful abstractions and an object-based representation. A trained model can be utilized to provide navigational goals to the lower-level controller trained with Hindsight Experience Replay (HER) [AWR+17]. Universal value function approximators (UVFAs) and HER [AWR+17] allow efficient learning when a goal is provided.

2 Methods

To facilitate semantic inference we borrow from methods successfully employed in robotics and autonomous driving, namely U-Net [RPB15], SLAM, OctoMap [HWB+13] and an object detection module [Goo19]. This allows the agent to consider the locations and spatial relationships of all elements of the environment based solely on RGB-image data and the agent's actions. In the MARLÖ environment, any action is elementary and performed in exclusion of any other action (move one cell forward, turn left or right 90 degrees). We trained an open-source Keras implementation of U-Net [RPB15] on 841322 pairs of RGB and ground-truth depth images collected from the Minecraft simulator. We used Tensorflow Object Detection Library [Goo19] for tracking the second player, pet and exits in the challenge. Training data was collected in different game sessions by manually labeling 10000 frames. Combined with our action-based SLAM approach, continuous integration of new distance data from the depth estimator results in robust scene reconstructions as 3D Octomap [HWB+13] voxel volumes.

Table 1: Predefined concepts.

Entity	Relation	Entity
Object B	is attached (adjacent) to	Object A
Object B	is on the left side of	Object A
Object B	is on the right side of	Object A
Object B	is above	Object A
Object B	is below	Object A
Object B	is on the same vertical line as	Object A
Object B	is on the same horizontal line as	Object A

The Octomap is queried every step in the game environment to update the abstract top-down map representation (Fig. 1).

We consider a mixed reinforcement learning and semantic learning framework with a pre-processing pipeline. The pipeline translates RGB-images $s_t \in S$ into an abstract representation $z_t \in Z$ of a top-down map suitable for navigation (Fig.1), and tracks objects $o \in O$ with the object recognition module. We translate the abstract representations $z_t \in Z$ into a Boolean space of concepts $c_t \in C$ and determine causality of rewarding events in $C \in B^c$ (c is the number of concepts). A concept checks for a set of relationships of objects in the environment and provides a boolean test result. Along with the test result, each concept collects a relative position of the tested object. We predefined a set of object-related spatial concepts which are computed for each pair of objects in the "CatchTheMob" environment at each time step (Table 1). We used human-players demonstrations (20 episodes) to collect trajectories ($s_t \in S$), actions, and rewards. The objective is to leverage human-player demonstrations to acquire a semantic model of rewarding states that enables the agent to select navigational goals in the environment. When a set of rewarding concepts is selected, their superposition determines a distribution of positions related to the reward. We feed a sample from this distribution as a goal for the policy that controls the agent. The policy is trained with HER [AWR+17] to navigate the agent to a goal position in the top-down map ($z_t \in Z$).

3 Untangling causality

Here we provide experimental results which highlight the structure of the problem in the "CatchTheMob" environment [MARa]. We trained a DQN using the Rainbow implementation [HMVH+18] to catch the pet with two agents (black curve in Fig. 2) in the "CatchTheMob" environment [MARa] on the reconstructed top-down map representations (Fig. 1). The reward function returns +1 when the two agents are at adjacent positions to the pet but from two opposite sides. The rewarding condition of being at two opposite sides of the pet can be learned end-to-end or resolved by a higher-level planning model. To measure the potential benefit of the model we trained the same DQN [HMVH+18] in a simplified single-agent "CatchTheMob" environment. The reward function returns +1 when a single agent is at a position adjacent to the pet (blue curve in Fig. 2). We got more than 50 times faster convergence than in the two-agent case. The red curve shows training of a single player with

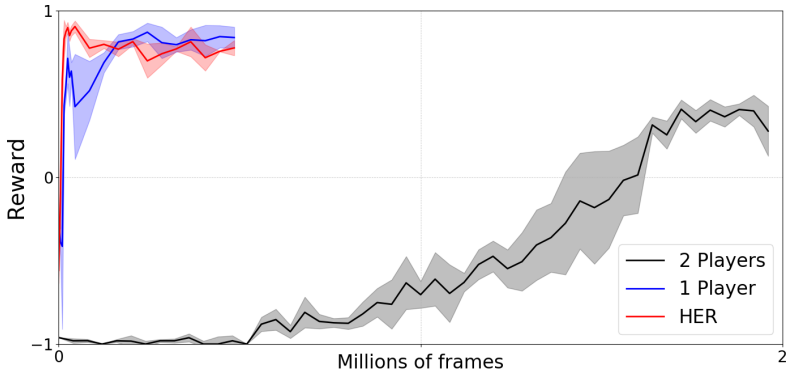

Figure 2: Convergence curves of agents trained on a 10x10 grid map. OY: cumulative reward (median of three seeds). The black curve shows training with two players, blue and red - with one player. In all experiments the episode runs until the reward +1 or the maximum of 50 steps is reached, the reward function gives -0.02 for each non-terminal action.

HER [AWR+17] to navigate to a given position. We got similar results to the single-agent case. These results highlight the benefit of higher-level planning through learning causality in the environment.

To learn the rewarding causality we select concepts activated at the rewarding events ($c_t \rightarrow c_{t+1}$, $r == 1$) as candidates for the necessary (but not sufficient) condition. Necessary concepts are encapsulated into a context and therefore not sufficient to fully determine the rewarding causality. To determine the sufficient set of concepts we optimize search by prioritizing concepts with a difference in rewarding / not rewarding samples and y a localization loss (1).

$$Loss = a\mathbf{P}(C_x) + b\mathbf{MSE}(D) \tag{1}$$

$\mathbf{P}(C)$ - reward prediction error of the selected set x of concepts C.
$\mathbf{MSE}(D)$ - mean square error of distribution of possible rewarding positions D. a, b - coefficients.

References

[AWR+17] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[GCH+19] William H Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, et al. The minerl competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:1904.10079*, 2019.

[Goo19] Google Brain. Tensorflow object detection api. `https://github.com/tensorflow/models/tree/master/research/object_detection`, 2019. Accessed: 2019-09-10.

[HMVH+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[HWB+13] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[JHHM] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell Microsoft. The Malmo Platform for Artificial Intelligence Experimentation *. Technical report.

[KMO+18] Łukasz Kidziński, Sharada Prasanna Mohanty, Carmichael F Ong, Zhewei Huang, Shuchang Zhou, Anton Pechenko, Adam Stelmaszczyk, Piotr Jarosik, Mikhail Pavlov, Sergey Kolesnikov, et al. Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 121–153. Springer, 2018.

[MARa] Crowdai marlo: Multi-agent reinforcement learning in minecraft. `https://www.crowdai.org/challenges/marlo-2018`.

[MARb] Github - marlo solution. `http://rebrand.ly/GitHub-MARLO`.

[MFSR19] Andrew Melnik, Sascha Fleer, Malte Schilling, and Helge Ritter. Modularization of end-to-end learning: Case study in arcade games. *arXiv preprint arXiv:1901.09895*, 2019.

[PLHM+19] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D. Gaina, and Daniel Ionita. The Multi-Agent Reinforcement Learning in Malm\"O (MARL\"O) Competition. jan 2019.

[RPB15] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[SM18] Malte Schilling and Andrew Melnik. An approach to hierarchical deep reinforcement learning for a decentralized walking control architecture. In *Biologically Inspired Cognitive Architectures Meeting*, pages 272–282. Springer, 2018.