Deep Reinforcement Learning on a Budget: 3D Control and Reasoning Without a Supercomputer

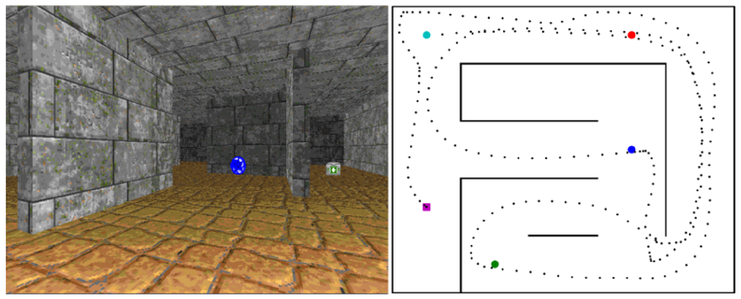
Edward Beeching, Jilles Dibangoye, Olivier Simonin and Christian Wolf

INSA Lyon, France

Abstract

We present a suite of tasks requiring complex reasoning and exploration in continuous, partially observable 3D environments. The objective is to provide challenging scenarios and a robust baseline agent architecture that can be trained on mid-range consumer hardware in under 24h. Our scenarios combine two key advantages: (i) they are based on a simple but highly efficient 3D environment (ViZDoom) which allows high speed simulation (12000fps); (ii) the scenarios provide the user with a range of difficulty settings, in order to identify the limitations of current state of the art algorithms and network architectures.

Scenarios and required reasoning



The ordered $K-$item scenario, where $K{=}4$. Left: The agent's view of the scenario. Right: a top-down view (unseen by the agent) of an episode where the agent collects the 4 items in the correct order.

Fig. 1: t

We have designed scenarios that aim to:

Be proxies for mobile robotics — the tasks represent simplified versions of real world tasks in mobile and service robotics, where an agent needs to act in an initially unknown environment, discovering key areas and objects. This also requires the tasks to be partially observable and to provide 3D egocentric observations.

Test spatial reasoning — the scenarios require the agent to explore the environment in an efficient manner and to learn spatial-temporal regularities and affordances. The agent needs to autonomously navigate, discover objects, eventually store their positions for later use if they are relevant, their possible interactions, the eventual relationships between the objects and the task at hand. Semantic mapping is a key feature in these scenarios.

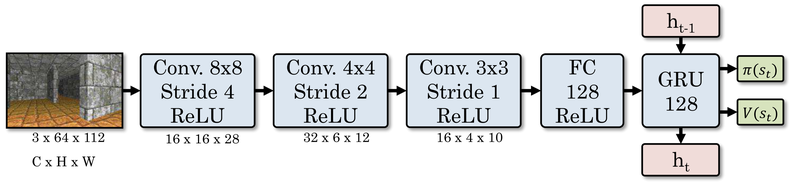
Discover semantics from interactions — while solutions exist for semantic mapping and semantic SLAM [1], we are interested in tasks where the semantics of objects and their affordances are not supervised, but defined through the task and thus learned from reward.

Generalize to unseen environments — related to being proxies for robotics, the trained policies should not encode knowledge about a specific environment configuration. The spatial reasoning capabilities described above should thus be learned such that an agent can make a difference between affordances, which generalize over configurations, and between spatial properties which need to be discovered in each instance/episode. We thus generate a large number of different scenarios in a variety of configurations and difficult settings and split them into different subsets, evaluating the performance on test data.

Agent Architecture

We apply a popular model-free policy gradient algorithm, Advantage Actor Critic (A2C) [3], implemented in the PyTorch [2] framework. A2C optimizes the expected discounted future return $R_t = \sum_{t=0}^{T} \gamma^t r_t$ over trajectories of states s_t, action a_t and reward r_t triplets collected during rollouts of a policy in the environment. The algorithm achieves this by optimizing a policy $\pi(.|s_t;\theta)$ and a value function $V^\pi(s_t;\theta)$. Both the policy and value function are represented by neural networks with shared parameters θ that are optimized by gradient descent.

The benchmark agent's network architecture is a 3 layer CNN similar to that used in [4] with 128 GRU to capture long term information, shown in Fig. 2.



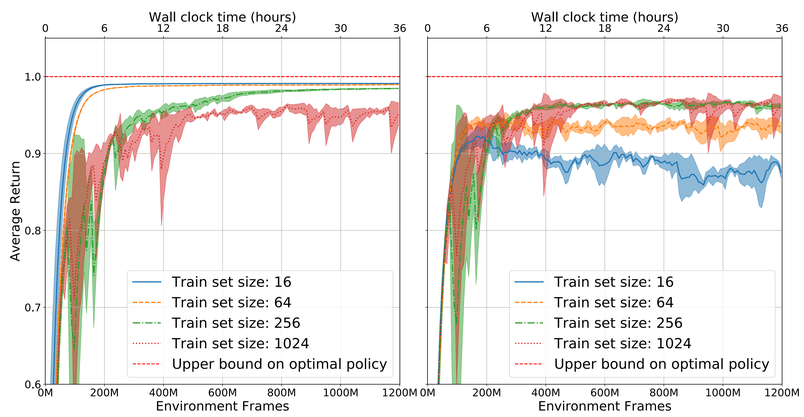
The benchmark agent's architecture

Fig. 2: t

Results

For each scenario there are a variety of possible difficulty settings. We have explored a range of simple to challenging configuration options and evaluated the agent's performance when trained for 200 M observations from the environment.

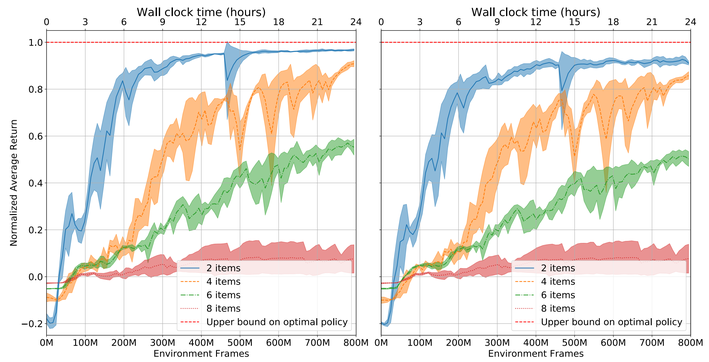
For each possible sub-configuration, 3 independent experiments were conducted with different seeds for the weight initialization in order to get an estimate of the stability of the training process. Results of the agent's mean performance on training and test datasets are shown with errors of one standard deviation across the 3 seeds. For each plot, we show a measure of the agent's performance such as average return or percentage of successful episodes on the y-axis and the number of environment frames on the x-axis. Figure 3 evaluates the generalization performance of the agent to unseen environment configurations, we vary the size of the training dataset and in order to quantify generalization performance.



Generalization to unseen maze configurations: the average return and std. of three separate experiments trained on the labyrinth scenario with training set sizes of 16, 64, 256 and 1024 mazes, evaluated on the training set (left) and a held out set of 64 mazes (right).

Fig. 3: t

Figure 4 shows results from the K-item task, experiments were conducted on 4 different scenarios of a fixed size of 5x5 with 2,4,6 and 8 items. Here, the more critical parameter is the number K of items to be retrieved in correct order. For the 2 and 4-item scenario configurations, we observe that the policies plateau near to the optimal policy, the benchmark agent struggles with the 6-item scenario and fails on the 8-item scenario.



Results from the "*Ordered k-item*" scenario for $k \in [2, 4, 6, 8]$, three experiments were undertaken for each k.

Fig. 4: t

Conclusions

This work presents four scenarios that require exploration and reasoning in 3D environments. The scenarios can be simulated at a rate that exceeds 12,000 FPS (frame skip=4) on average, including environment resets. Training can be done up to 9,200 FPS (frame skip=4), including forward and backward passes. Experiments have been conducted for each scenario for a wide variety of difficulty settings. We have shown robust agent performance across random weight initialization for a fixed set of hyper-parameters. Generalization performance has been analyzed on held out test data, which demonstrates the ability of the benchmarks to generalize to unseen environment configurations that are drawn from the same general distribution. We have highlighted limitations of a typical RL baseline agent and have identified suitable scenarios for future research.

Acknowledgements

This work was funded by grant Deepvision (ANR-15-CE23-0029, STPGP479356-15), a joint French/Canadian call by ANR & NSERC. We gratefully acknowledge support from the CNRS/IN2P3 Computing Center (Lyon - France) for providing computing and data-processing resources needed for this work.

References

[1] J. Civera et al. "Towards semantic SLAM using a monocular camera". In: *IROS*. 2011.

[2] Ilya Kostrikov. *PyTorch Implementations of Reinforcement Learning Algorithms*. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr. 2018.

[3] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In: (2016). URL: http://arxiv.org/abs/1602.01783.

[4] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* (2015).