NAVIGATING UNSEEN ENVIRONMENTS USING DEEP REINFORCEMENT LEARNING

Shurjo Banerjee*, Vikas Dhiman*, Brent Griffin & Jason J. Corso
The Electrical Engineering and Computer Science Deparment
The University of Michigan
Ann Arbor, MI 48109, USA
{shurjo,dhiman,griffb,jjcorso}@umich.edu

ABSTRACT

In recent years, agents trained via Deep Reinforcement Learning based methods have shown a lot of promise in learning to navigate to goal locations in complicated three-dimensional worlds. Of paritcular interest has been the fact that these agents have to shown to reliably perform this navigation without the need for any explicit SLAM or path-planning. In their current form, however, these agents are not well understood enough to serve as a viable replacement to traditional methods. Of particular concern has been the literature's standard approach of utilizing the same environments for training and testing in the evaluation of these agents. In this work we (a) Reimplement state-of-the-art DRL agents and quantify their performance across thousands of seen and unseen environments possessing large amounts of variety (b) We introduce simple modifications to standard architectures to improve the performance of these agents on previously unseen worlds. Our findings showcase that state-of-art methods are able to generlized learned behaviors but retain no memory of previously visited areas. Our modified agents are found to outperform them drastically via the explicit use of frame-action look up tables within these networks.

1 INTRODUCTION

Deep Reinforcement Learning (**DRL**) has gained a wide amount of interest in the artificial intelligence and computer vision communities in the last few years. From Mnih. et. al's seminal paper of learning to play Atari games from pure visual input (Mnih. et. al, 2013) and the much publicized defeat of Go's world champions by Deepmind's Alpha-Go agent (Silver, Huang and Maddison, 2016) to the recent defeat of the DOTA2 world champion by OpenAI, the successes of DRL have been wide, varied and promising.

Recent work by Mirowski et. al. has shown the potential applicability of DRL methods to robotic navigation. Agents, trained on pure monocular vision, have been demonstrated to learn simple navigation tasks in complicated three dimensional worlds without any requirements for explicit SLAM or path-planning.

What makes these DRL navigation systems (**DRLNAV**) particularly attractive is the property of **environment invariance**. The exact same methods work on a variety of worlds ranging from simple 2D mazes to complicated 3D physics engines without requiring any form of architectural modifications.

In their current form, however, DRLNAV agents are yet to showcase the generalized performance to serve as a true viable alternative to classical methods. As with all deep network related work, they require enormously large amounts of training data. Oftentimes, the agents are trained and validated upon the **same environments** over hundreds of millions of iterations. With no precautions taken against overfitting, the millions of parameters availed by deep learning architectures, and the usage of the same maps for training and testing, it is important to ascertain that DRL agents aren't simply

*indicates equal contribution

memorizing environments in their entirety instead of truly learning navigation based explorative skills.

For game-like environments such as Atari-Breakout or GO with a fixed set of rules, exploring and over-fitting to the entirety of the state space allows for agents that are able to defeat any form of competitor. In the context of navigation, however, agents must be able to generalize learned navigational directives to unseen worlds.

Our contributions in this work are four-fold:

1. **DRL and Generalized Navigation**
   We analyze the ability of current state-of-the-art DRL agents at navigating unseen environments. Our experiments are carried out across both fully observable and partially observable environments so as to quantify the strengths and weaknesses of these agents across different domains. For our simplest environments, we carry out this analysis against the backdrop of path-planning so as to have an additional comparison metric against explicitly created classical non-learning based methods.
2. **Blindfolded-Curriculum Training**
   We introduce modifications to the DRL architecture for agent training across different environments which we dub Blindfolded Curriculum Training (BCT). We showcase how BCT achieves *improved* performance over traditional methods and leads to agents more suited for long term planning.
3. **Implicit Map Querying**
   We showcase BCTs greatest strength - mainly the ability to query an agent's implicit understanding of its surroundings at any point in time. We quantify this ability as measure of the agent's ability to internalize local map structures and call it **implicit mapping**.

2 Related Work

Localization and mapping Robotic localization and mapping for navigation as a problem since the beginning of mobile robotics and sensing. Smith and Cheeseman **?** introduced the idea of propagating spatial uncertainty for robot localization while mapping and Elfes popularized Occupancy Grids **?** for mapping. In the last three decades, the field has exploded with variation of algorithms for different sensors like cameras, laser scanners, sonars, depth sensors, variation in level of detail like topological maps **?** for low level of detail to occupancy grid maps for high detail and variation in environment types like highly textured or non-textured.

All these approaches require huge amount of hand-tuning and design for adapting to different environments and sensor types. The level of detail of maps also needs to be decided before hand irrespective of the application and hence is not optimized for the application at hand.

Deep reinforcement learning Deep reinforcement learning (DRL) came back to the limelight **?** Check whether this citation should be here with Mnih et al. **??** demonstrating that their algorithms outperform humans on Atari games. Subsequently, the DRL algorithms have been extended **?** and applied to various games **?**, simulated platforms **?**, real world robots **?** and more recently to robotic navigation **??**.

The exploration into robotic navigation using deep reinforcement learning is a nascent topic, it has potential to disrupt the fields of simultaneous localization and mapping and path planning. Also, **?** train and test on the same maps which limits our understanding of the generality of the method. In fact, it is very common to train and test on the same environments in reinforcement learning based navigation works **??** with the only variation being in location of goal and starting point. In contrast, **?** do test on random maps but the only decision that the agent has to make is avoid a goal of particular color and seek other color rather than remembering the path to the goal. On similar lines, **?** test their method on unseen maps in VizDoom environment but only vary the maps by unseen texture. In this work, we take the study of these methods significantly farther with a thorough investigation of whether DRL-based agents remember enough information to obviate mapping algorithms or the need to be augmented with mapping algorithms.

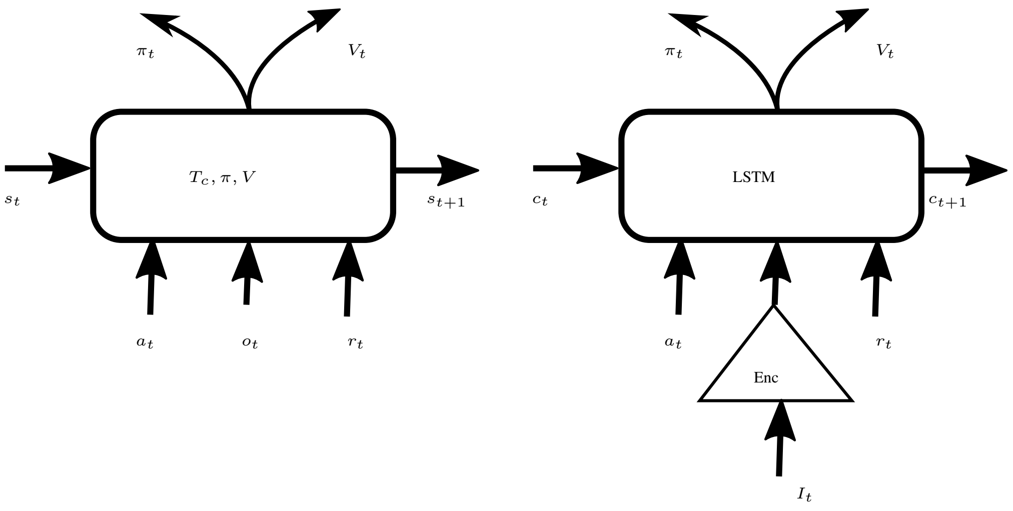

Figure 1: POMDP on the left, neural network implementation on the right.

3 Background

Our experimental setup is inspired by Mirowski et al. work **?**. We summarize the technical setup here for completeness. We recommend **???**

The problem of navigation is formulated as interaction between environment and agent. At time time t the agent takes an action $a_t \in \mathcal{A}$ and observes observation $o_t \in \mathcal{O}$ along with a real reward $r_t \in \mathbb{R}$. We assume the environment to be Partially Observable Markov Decision Process (POMDP). In a POMDP the state of the environment $s_t \in \mathcal{S}$ is assumed to be the only information that is propagated over time and both o_t and r_t are assumed to be independent of previous states given current state and last action. Formally, a POMDP is a six tuple $(\mathcal{O}, C, \mathcal{S}, \mathcal{A}, T, R)$ that is observation space $\mathcal{O}$, observation function $C(s_t, a_t) \to o_t$, state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T(s_t, a_t) \to s_{t+1}$ and reward function $R(s_t, a_t) \to r_{t+1}$ respectively. For our problem setup, the observation space $\mathcal{O}$ is the space of encoded feature vector that can be generated from input image or combination of other inputs, action space $\mathcal{A}$ contains four actions: rotate left, rotate right, moved forward and move backward and reward function R is defined for each experiment so that the reaching the goal leads to high reward with auxilary reward to encourage certain kind of behavior.

For Deep reinforement learning the state space $\mathcal{S}$ is not hand tuned, instead it is modeled as semantically meaningless *hidden state* of a fixed size float vector. Also, instead of modeling observation function $C(s_t, a_t) \to o_t$ and $T(s_t, a_t) \to s_{t+1}$, a combined transition function $T_c(s_t, o_t, a_t, r_t; \theta_T) \to s_{t+1}$ is modeled such that it estimates next state s_{t+1} directly considering previous observation as well as reward into account. For policy-based DRL a policy function $\pi(a_{t+1}|s_t, o_t, a_t, r_t; \theta_\pi) \to \pi_t(a_{t+1}; \theta_\pi)$ and a value function $V(s_t, o_t, a_t, r_t; \theta_V) \to V_t(\theta_V)$ are also modeled. All three functions T_c, π_t, V_t share most of the parameters in a way such that $\theta_T \subseteq \theta_\pi \cap \theta_V$

Our objective is to estimate unknown weights $\theta = \theta_T \cup \theta_\pi \cup \theta_V$ that maximizes the expected future reward, $R_t = \sum_{k=t}^{t_{end}-t} \gamma^{k-t} r_k$, where γ is the discount factor,

$$\theta^* = \arg\max_\theta \mathbb{E}[R_t]\,. \tag{1}$$

Asynchronous Advantage Actor-Critic In this paper we use policy-based method called Asynchronous Advantage Actor-Critic (A3C) **?** that allows weight updates to happen asynchronously in a multi-threaded environment. It works by keeping a "shared and slowly changing copy of target network" that is updated every few iterations by accumulated gradients in each of the threads. The gradients are never applied to the local copy of the weights, but the local copy of weights is periodically synced from the shared copy of target weights. The gradient for weight update is proportional to the product of *advantage*, $R_t - V_t(\theta_V)$, and *characteristic eligibility*, $\nabla_{\theta_\pi} \ln \pi_t(a_{t+1}; \theta_\pi)$ **?**, updating the weights according to the following update equations

$$\theta_\pi \leftarrow \theta_\pi + \sum_{t \in \text{episode}} \alpha_\pi \nabla_{\theta_\pi} \ln \pi_t(a_{t+1}; \theta_\pi)(R_t - V_t(\theta_V)) \tag{2}$$

$$\theta_V \leftarrow \theta_V + \sum_{t \in \text{episode}} \alpha_V \frac{\partial (R_t - V_t(\theta_V))^2}{\partial \theta_V} \,. \tag{3}$$

For more details of the A3C algorithm please refer to **?**.

4 APPROACH

4.1 BASELINE MODELS

We utilize the model presented in Mirowski et al.as our baseline model. The agent's egocentric view is fed in to a deep network that in turn converts it to an action to take within the environment. Due to the POMDP nature of the problem, memory is provided to the agent via a set of stacked LSTMs so that it can learn to assign contextual importances to past actions and observations. As mentioned in the paper, we provide our agents will the auxiliary loss signals of depth prediction and loop closure to improve convergence speeds. Our agents are trained on the same environments utilized in the original paper and display comparable results. We additionally train on our agents on newer random environments of different dimensions coupled with several more wall texture varieties so as to provide more quantitative evaluations of the ability of these agents in more diverse environments.

4.2 BLINDING

We incrementally blind these agents so as to guage the amount of information that is actually required by agents to perform navigation in the contexts of these environments. Intuitively, we expect agents thus trained to perform better long-term planning due to the potential unreliability of future observations at any given point. We experiment with two main types of blinding.

4.2.1 CURRICULUM BLINDING

In currulum blinding, agents are simultaneously asked to learn to navigate in conjunction with receiving increasingly "blind" data. The agent is tasked with learning a harder combined task in the hope that it more readily learns how its actions can affect long term rewards gained instead of relying on receiving the corresponding frames at every single point of time. Blinding is linearly increased from 0 to 100 over the course of each trial.

4.2.2 FINE-TUNING

In the fine-tuning approach, baseline agents are first trained on the map using the standard A3C approach. Agents are then "fine-tuned" by being fed increasingly blind data where the blindness again climbs linearly over the training period. The hope is that the agent learns to augment its already learned navigation strategies with improve versions of looking in to the future due to the blindness etc etc.

4.3 BLINDING: SELF-SUPERVISED CURRICULUM TRAINING

In both sets of blindfolding experiments described thus far, the blinding is forced upon the robotduring training time. Every maze, however, possesses different degrees of difficulty in their different parts. For example, navigating a single corridor blind is a much easier task than that of navigating cross roads.

Based on this idea, we introduce BLINC, wherein agents are incentivized via small additional reward signals to

5 Experiments

5.1 Baseline Models

PART	**DESCRIPTION**
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

Table 1: Baseline experiments.

5.2 Blinding: Curriculum and Finetuning

5.3 Blinding: Self-Supervised Curriculum Training

6 Analysis

7 Conclusion

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

References

References