BLINC: QUERYING THE INTERNAL BELIEF STATES OF DEEP-REINFORCEMENT LEARNING METHODS

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

Deep reinforcement learning (DRL) algorithms have demonstrated strong progress in learning to reach a goal in challenging three-dimensional environments without requiring any explicit SLAM or path-planning in their navigation. While promising, the limitations and underlying pattern recognitions performed by these networks based approaches are not very well understood. In this work we introduce the BLINC training paradigm - an appendum to standard DRL models that can be used to (a) Fine-tune DRL algorithms and improve their performance (b) Afford new mechanics allowing for the querying of the internal states of these methods (more details as I figure it out). In BLINC, agents are incentivized to blind themselves as often as possible in the course of their navigation so as to more naturally pick up on aspects of long term planning. We find that BLINC consistently improves rewards scores by 5% across multiple environments and worlds. Our querying mechanics provide both qualitative and quantiative notions of the kind of beliefs learned and propagated by the underlying layers of these network based agents.

1 INTRODUCTION

Navigation remains a fundamental problems in mobile robotics and artificial intelligence **??**. The problem, traditionally called SLAM (Simulataneous Localization and Mapping), is classically addressed by separating the eventual task of navigation into *exploration* and *exploitation*. In the exploration phase, the environment is incrementally built and represented in some sort of *map* data-structure. In exploitation, this data structure is used for localization and path-planning to find an optimal path to a given destination based on desired optimality criterion. These classical methods, have been explored in depth and there have been many advances in this classical approach **?**, it remains a difficult challenge. *Either mention some examples or cite a paper that highlights the failiures of SLAM or both!*

More recently, end-to-end navigation methods—methods that attempt to solve the navigation problem without breaking it down into the separate parts of localization, mapping and path-planning—have gained traction. With the recent success of Deep Reinforcement Learning (**DRL**) **??**, these end-to-end navigation methods **?????** forego decisions about the details that are required in the intermediate step of map building.

Work by Mirowski et al. showcased agents that learned to navigate textureless environments to find desired goal locations trained on pure monocular vision - a feat that is still quite difficult for state-of-the-art monocular SLAM systems **?**. *Talk about the memory structures used - no need for any explicit path planning, slam or all that nonsense* The potential for simpler yet capable methods is rich on the surface.

Despite this potential and recent successes, state-of-the-art DRL based methods have been confronted with their own set of problems. In line with other Deep-Learning fallacies (*too negative?*), foremost among these is the difficulty in understanding the method limitations or the kind of patterns

[*] indicates equal contribution

that these algorithms are understanding. The inherent black-box nature of these methods make them hard to study.

In this work, we attempt to pull back the lid of how these networks appear to be in fact be performing this navigation. We phrase these queries within the context of exploration and exploitation as is traditional in the SLAM world. Our contributions are three-fold:

1. We succesively blind state-of-the art DRL agents in a curriculum fashion to gain an understanding of whether these agents can be forced to perform long-term planning in the execution of their learned navigation strategies.
2. In a bid to more easily teach agents to perform long-term planning, we introduce BLINC. BLINC, is a conceptually simple modification applicable to all DRL methods wherein agents are incentivized to blind themselves during navigation as often as possible. Extra incentives are provided when this blindness is contiguously performed over several frames. We showcase how agents trained via BLINC acheive better performance then current state-of-the-art methods.
3. We showcase BLINCs great strength in affording the ability to easily query the hidden states of the networks used by these models. We use this method to gain an understanding of each agent's explicit understanding of its surroundings at given points of time.

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.

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

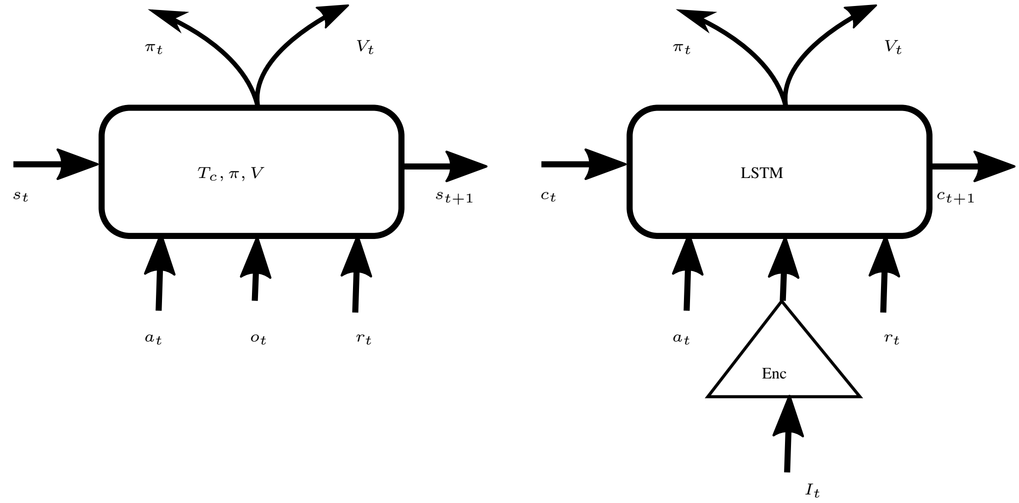

Figure 1: POMDP on the left, neural network implementation on the right.

$r_t \in \mathbb{R}$. 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 $C(s_t, a_t) \rightarrow o_t$, state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T(s_t, a_t) \rightarrow s_{t+1}$ and reward function $R(s_t, a_t) \rightarrow r_{t+1}$ 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 $C(s_t, a_t) \rightarrow o_t$ and $T(s_t, a_t) \rightarrow s_{t+1}$, a combined transition function $T_c(s_t, o_t, a_t, r_t; \theta_T) \rightarrow s_{t+1}$ 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 $\pi(a_{t+1}|s_t, o_t, a_t, r_t; \theta_\pi) \rightarrow \pi_t(a_{t+1}; \theta_\pi)$ and a value function $V(s_t, o_t, a_t, r_t; \theta_V) \rightarrow V_t(\theta_V)$ 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