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

Anonymous authors
Paper under double-blind review

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. In BLINC, agents are incentivized to blind themselves as often as possible in the course of their navigation so as to more naturally lock on to aspects of long-term planning in the context of their motion. We find that BLINC consistently improves rewards scores by x% 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 that these algorithms are understanding. The inherent black-box nature of these methods make them hard to study.

In this work, we introduce qualitative and quantitative metrics to better understand how these networks appear to be learning and performing these navigation tasks. Our contributions are three-fold:

1. In the course of their navigation, we ask the question of how much visual information is actually required by state-of-the-art drl agents in the course of navigating previously explored environments. To guage this, we incrementally blind agents at training time so as to quanititavely assess the point at which navigation failiure occurs.

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.

Inattention blindness in humans Our experiments with blindfolding can be interpreted in relationship with attention affordance while driving. How much inattention can we afford while driving on a particular speed? While these questions haven't been answered in psychology, there have been studies **?** showing that cell phone users driver more slowly and follow farther from a paced car indicating that visual cues for navigation are dependent the distance travelled rather than the time travelled. But the very idea that cell phone users can drive supports the idea that not all visual information at all frames is not necessary for navigation.

While humans do not literally blindfold themselves while driving, they do suffer from inattentional blindness. Inattentional blindness is more common in familiar scenarios and experienced drivers then in unfamiliar scenarios **???**.

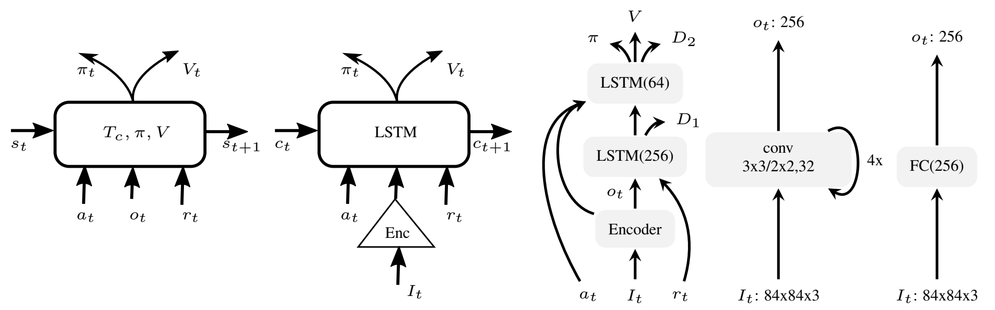

Figure 1: POMDP on the left, neural network implementation on the right. Nav A3C architecture on left and two possible Encoders: CNN encoder on center, FC encoder on right. We use CNN encoder for the 3D world while FC encoder for the gridworld.

Apart from the similarities, our experiments differ significantly from psychology experiments that explore the relationship between inattentional blindness and route-familiarity. While the psychology experiments measure change in reaction time to emergency events in dynamic environments, we measure the percent of time agent can afford blindfolding while navigating static familiar route.

3 BACKGROUND

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

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 $C(s_t, a_t) \rightarrow o_t$, 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

Our agents are trained on the same environments utilized in the Mirowski et al.paper. We additionally generate a variety of random mazes of dimensions 7x7, 9x9, 11x11, 13x13 and 15x15 and quantify performance on them as well.

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 in the original work, we provide our agents will the auxiliary loss signals of depth prediction and loop closure to improve convergence speed and stability. Our agents utilize a simpler, discretized motion model wherein at every point they must choose between the*forward*, *backward*, *rotate left* and *rotate right* actions.

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 environments on which they have been trained. Intuitively, we expect agents trained with some form of blinding to perform better long-term planning due to blindness-related unreliability in the expected future observations. We experiment with two kinds of blinding: curriculum and fine-tuning.

4.2.1 Curriculum Blinding

In currulum blinding, agents are incrementally blinded over training time while simulateneously being tasked with the learning to navigate task. The agent is tasked with learning a harder combined task in the hope that it more readily learns how its actions correspond to rewards gained in the long term. 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 pre-trained on the maps using the standard A3C approach. Agents are then ”fine-tuned” by being trained on the incrementally blind data where the blindness again increases from 0% to 100% over the course of training. These section of the experiments test whether the agents can transform their learned short term navigational plans to more long term ones and are also useful in determining where failiure occurs.

4.3 Blinding: Self-Supervised Blinding-Curriculum Training

In the previous experiments, blinding is forced upon the agent at training time. The agent at each instant is expected to formulate some sort of plan for future navigation due to the expected unreliabilities in future observations. Navigating blind, however, varies in difficulty greatly depending on the geometery of the section of the maze surrounding the agent at any point. For example, navigating blind down a straight corridor is a conceptually easier task than navigating a hallway with several turns in it.

Based on this intuition, we introduce the BLINC method. In BLINC, agents are incentivized via small rewards to blind themselves during navigation. This blinding is optional and is introduced via a doubling of the state space. At every instant, the agent must choose between the *forward*, *blind-forward*, *backward*, *blind-backward*, *rotate left*, *blind-rotate left*, *rotate right* and *blind-rotate right* actions. When blind actions are chosen, the agent recieves small positive reward values so as to encourage it perform this actions often. Similar to previous experiments, the blinding is introduced in incremental fashion so that the agent may first learn how blind-actions correspond to the non-blind ones in terms of action performed. We experiment with different reward structures to incentivize self-blinding as often as possible. In particular, we are interested in agents that choose to blind themselves contiguously over several frames so we may conclude that some form of long-term planning is occuring.

4.3.1 Constant Reward

In the constant-reward version of the experiment, agents are rewarded with a constant small reward for every blind action taken. Care is taken to ensure that the cumulative reward of performing ab lind action at every frame is less than that of find the goal in the maze once.

4.3.2 Contiguous-Linear Reward

In the contiguous-linear case, the first blind action is rewarded with a small position value. For every subsquent blind action performed immediately afterwards, the reward is linearly increased by a the number of blind frames till that point. The reward at the n^{th} contiguous blind frame is:

$$r_n = \frac{R}{2F} + \frac{R}{(2F)(2F+2)}$$

4.3.3 Contiguous-Quadratic Reward

Some contiguous reward structure that solves ALL our problems. WHERE ARE YOU!

5 Experiments

Static Goal, Static Spawn	Agent	Reward	Latency 1:> 1	Goals	% Blindfolding	Cont. Blind Frames
I-Maze	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Static 1	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Static 2	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Training-9x9-*	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Static Goal, Random Spawn	Agent	Reward	Latency 1:> 1	Goals	% Blindfolding	Cont. Blind Frames
I-Maze	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Static 1	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Static 2	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Training-9x9-*	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Random Goal, Random Spawn	Agent	Reward	Latency 1:> 1	Goals	% Blindfolding	Cont. Blind Frames
I-Maze	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Random 1	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Random 2	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					
Random-9x9-*	A3C					
	A3C-CB					
	A3C-FT					
	A3C-BLINC					

Table 1: We achieve comparable results when we test and train on the same experiments.

Static Goal, Static Spawn	Reward	Latency 1:> 1	Goals	Done
Planning-I-Maze				✗
Static 1				
Static 2				✗
Training-9x9-1				✗
-Training-9x9-120				
-Training-9x9-121				
-Training-9x9-122				
-Training-9x9-123				
-Training-9x9-124				
-Training-9x9-125				
-Training-9x9-126				
-Training-9x9-127				
-Training-9x9-128				
-Training-9x9-129				
Static Goal, Random Spawn	Reward	Latency 1:> 1	Goals	Done
Planning-I-Maze				
Static 1				
Static 2				
Training-9x9-1				
Random Goal, Static Spawn	Reward	Latency 1:> 1	Goals	Done
Planning-I-Maze				
Random 1				
Random 2				
Training-9x9-1				
Random Goal, Random Spawn	Reward	Latency 1:> 1	Goals	Done
I-Maze				
Random 1				✗
Random 2				✗
Training-9x9-1				✓

Table 2: We achieve comparable results when we test and train on the same experiments.

Random Mazes	Train Reward	Train Latency	Train Goals	Test Reward	Test Latency	Test Goals	Done
Training-9x9-10							✓
Training-9x9-100							✓
Training-9x9-500							✓
Training-9x9-1000							✓

Table 3: We achieve comparable results when we test and train on the same experiments.

5.1 Blinding: Curriculum and Finetuning

5.2 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