FoLaR: Foggy Latent Representations for Reinforcement Learning with Partial Observability

Hardik Meisheri
TCS Research
Mumbai, India
hardik.meisheri@tcs.com

Harshad Khadilkar
TCS Research
Mumbai, India
harshad.khadilkar@tcs.com

***Abstract*—We propose a novel methodology for improving the rate and consistency of reinforcement learning in partially observable (*foggy*) environments, under the broader umbrella of robust latent representations. The present work addresses partially observable environments, which violate the canonical Markov assumptions. We propose adaptations for *any* on-policy model-free deep reinforcement learning algorithm, in order to improve training in partially observable situations: (i) recurrent layers for including information from previous observations, (ii) predicting the step reward and the next latent representation as auxiliary outputs from the same latent space as used for inferring the action, and (iii) modification of the loss function to penalise errors in the two auxiliary outputs, in addition to the reward-based gradients used for policy training. We show that the proposed changes substantially improve learning in several environments over vanilla Proximal Policy Optimisation (PPO) and other baselines in literature, especially in known challenging environments with hard exploration.**

***Index Terms*—reinforcement learning, partially observable settings, robust latent representation**

I. Introduction

Reinforcement learning (RL) in its canonical form is designed to find optimal policies for Markov decision processes (MDPs). This set of algorithms has produced remarkable successes in fully observable settings, both deterministic [1], [2] and stochastic [3]. However, it is well-known that the learning degrades substantially in partially observable settings, at least partly due to the violation of Markov assumptions. One of the key reasons of poor performance of deep reinforcement learning (Deep RL) in such situations is overfitting to noise during training, which itself is a result of poor knowledge of the environment. Recent work [4]–[8] has shown that more robust latent representations within the Deep RL agent can lead to substantial improvements. The goal of this work is to adapt this idea for partially observable environments, and to demonstrate its effectiveness compared to standard RL algorithms. We call the concept *FoLaR* or Foggy Latent Representations, because the partial observability is similar to operating in a fog.

Most DRL techniques assume that the state is fully observable at each timestep, though they are frequently used in scenarios where this assumption is violated. In such instances, information available during the decision-making process is neither perfect nor complete. In this paper, we deal with partially observable environments where observations are a subset of the true state at that timestep. Partial observability results in a generalised version of MDP known as POMDP [9], [10]. DRL is known to suffer from sample inefficiency, and this issue is further exacerbated in POMDP settings. One way to tackle this challenge is to learn robust latent representations, using a combination of value, policy, and other signals. This idea has been used in robotics and control domains to reconstruct either the next state or current state or both [6], [7]. However, the latent representations are typically learnt in advance using techniques such as variational autoencoders (VAE). Learning latent representations decoupled from policy and reward signal often leads to the encoding of information that is irrelevant to the task at hand [8], [11], though such models do have the advantage of being generalizable to similar environments with zero or few shot learning [12].

Some recent studies have focussed on predicting the latent representations, rather than the entire state or observation, and have shown improvements in policy learning and sample efficiency. However, their application has been primarily to physical systems (such as the DeepMind Control Suite) or to fully observable instances with hard exploration (such as Montezuma's Revenge) [8], [11], [13], [14]. By contrast, this paper investigates the efficacy of predicting future latent representations (in addition to usual outputs such as value and policy) in POMDPs. Combined with noisy rewards and partial and imperfect information, prediction of the next state is at best approximate and noisy or foggy. We present our algorithm which augments the loss function of any off the shelf algorithm and trains it end-to-end (learning representations and policy improvements at the same time), achieving better sample efficiency or in some cases solving tasks which are not solvable with vanilla algorithms. We present our findings in an empirical manner and discuss some intriguing observations. We hypothesize that augmenting latent representations with predictive loss and learning end-to-end generates better policy and sample efficiency as opposed to decoupled learning of representations and policy. Our method operates entirely in the latent space and does not rely on reconstructing raw states; this helps in learning to encode information that is relevant to the task. The key contributions of this paper are,

- Proposing a training paradigm and loss function to learn robust latent representations in POMDP settings contex-

tualized on latent representations of belief state,
- Ability to modify any off-the-shelf RL algorithm to improve the sample efficiency and exploration characteristics, and
- Extensive evaluation on two partially observable environments with varying scales.

In the rest of the paper, we review prior work in building latent representations and its nuances Section II. We describe our methodology for implementing FoLaR in Section III. Section IV provides detailed overview of experiments conducted and different environments considered. Empirical results and ablation studies are presented in Section V and finally we conclude our work in Section VII along with some possible future directions.

II. Related Work

The idea of learning representations effectively for generating better policy and in turn solving tasks in a much more efficient way has been studied from various perspectives. Recently, it was proposed that the ideal representations should allow prediction in value function space via any linear prediction map [15], [16]. In general, there is a common consensus that learning more robust representations of states (in case of POMDP belief states) results in better policies and stable value functions. One option is to learn better representations by imposing a constraint of reconstructing the entire state or observation from the latent representations, forcing the model to encode all the information into dense representations [6], [14], [17]. While this idea does prove to be effective, it can force the model to encode information which irrelevant to the task at hand [11].

Another paradigm which has received significant interest is to decouple the representations learning from policy improvement [4], [18], [19]. These algorithms iterate between policy updates and representation learning, to obtain continuous improvement. We hypothesize that although decoupling the representations learning works better in fully observable environments, they do not perform well in partially observed environments (where information is not complete and is often noisy). Since the observations only partially describe the state, the same latent representations may correspond to very different true states, leading to noisy policy updates. We postulate that co-learning these two tasks will lead to more robust updates to both the latent representations and the policy.

Recently, the notion of making latent representations reflect the state (dis)similarity has been proposed. Earlier, this was achieved by adding a loss based on some similarity metric [20] in the input space and latent space. One such metric is bisimulation and is recently being used to learn better representations without reconstruction [8], [11]. However, the metric presented is computationally intensive to compute and difficult to adapt to stochastic and/or partially observable environments, as the notion of choosing the same actions in similar states is unclear when the states are only partially observed [21].

In [22], authors present the equivalence of trace between the belief state in POMDPs which is similar to the equivalence of states in MDPs. In particular, two belief states are said to be belief trajectory equivalent if, for any choice of future actions, they generate the same distribution on future observations and rewards. Such belief trajectory equivalence is related to predictive state representation (PSR) [23]–[28] which are a compression of the past history which is sufficient to predict the future observations. The compression of past state information can also be viewed as an aggregation of partially observed states into belief state [29]. We, therefore, contextualized the prediction of next state representations on the history of belief state representations.

Self Imitation Learning (SIL) [5] proposes to learn latent representations better by leveraging past experiences to solve harder exploration problems. They augment the learning algorithm by forcing the model to learn its past actions when the expected reward is greater than the value approximated by the agent in an actor-critic framework. They achieve the state of the art results in Atari games and more specifically Montezuma's Revenge. We have used SIL as one of the baselines for our experiments.

III. Methodology

A Markov decision process (MDP) can be modelled as $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ denotes the decision or action space, $\mathcal{T}$ represents transition probabilities from one combination of state and action to the next, $\mathcal{R}$ denotes the rewards, and γ is the discount factor for future rewards [30], [31]. This formulation can be extended to partially observable Markov decision processes (POMDPs) with the inclusion of belief state and noisy information. These are typically modelled as a controlled state and an agent which makes noisy corrupted observations of the state [9], [32].

Actor-Critic extensions to policy gradient methods have been very successful in solving complex and hard exploration tasks in the past [1], [2], [33]–[35], since having a value estimate of state $V(s)$ helps in reducing the variance in the policy updates. In general, policy update in the policy gradient framework is defined as,

$$\Delta\theta \propto \nabla_\theta log(\pi_\theta(a_t/s_t))A(s_t, a_t), \tag{1}$$

where π_θ represents the policy parameterized by θ and $A(s_t, a_t)$ represents the advantage. This function is defined by $A(s_t, a_t) = Q^\pi(a_t, s_t) - V^\pi(s_t)$. An extension introduced in [1] uses entropy regularization in gradient updates to improve exploration and avoid local convergence. The gradient update is modified as follows,

$$\Delta\theta \propto \nabla_\theta log(\pi_\theta(a_t/s_t))A(s_t, a_t) + \beta\nabla_\theta H(\pi_\theta(s_t)), \tag{2}$$

where β is a hyperparameter known as entropy coefficient and H is the entropy. $A(s_t, a_t)$ is estimated using Generalized Advantage Estimates (GAE) [36]. In this paper, instead of estimating GAE from a small number of environment steps per trajectory, we estimate from full rollouts and use whole trajectory to make a single gradient update. This reduces the variance in the gradients, a crucial requirement for the sparse reward environments with terminal rewards that we consider.

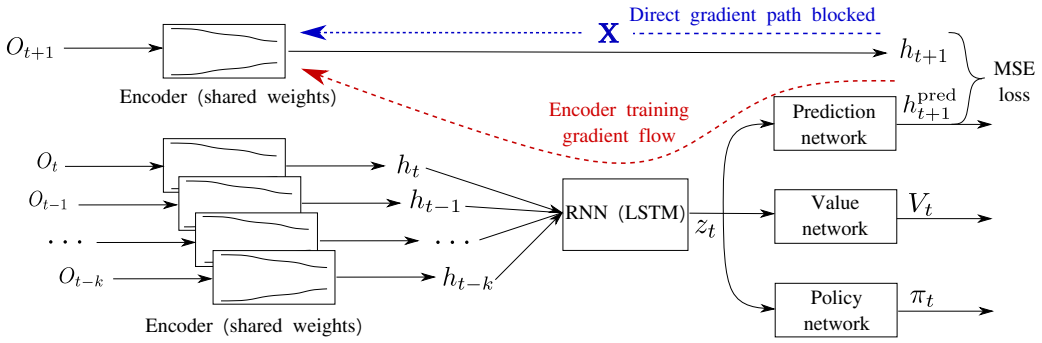


Fig. 1: System Architecture

As described in [29], the belief state in POMDPs (where a partial observed state is denoted by O_t) can be estimated from current and previous observations $O_t, O_{t-1}, \ldots, O_{t-k}$ using recurrent memory, where k is the length of the time window into the past (illustrated in Fig. 1). As shown in the figure, we first generate an encoded version $h_t, h_{t-1}, \ldots, h_{t-k}$ of these observations, which are then passed to the recurrent portion of the network. While the encoder could be of any architecture in general, we use CNN layers in this work. Similarly, the recurrent portion of the network could be anything in general, but we use an LSTM (long short-term memory) [37] in the present work to generate latent representations z_t. This representation is then used by three downstream networks: (i) value network, (ii) policy network, and (iii) prediction network.

Both policy and value networks use the same latent representation z_t as the input, leading to fewer trainable parameters and potentially richer representations. The task of the prediction network in the proposed architecture is to predict the encoded representation of the next observation, and this is trained using the loss relation,

$$L_t^{NL} = minimum(\eta \times mse(h_{t+1}^{pred}, h_{t+1}), \epsilon) \qquad (3)$$

where h_{t+1} is the encoded latent representation of next state (available after one time step), η is a hyperparameter defining the weight to be associated with this loss, and ϵ defines the clip limit. We clip the loss to reduce the variance induced into the latent representations, by providing a pessimistic lower bound on the change in the latent representations before and after one training batch. In addition, the gradients are not allowed to flow directly to encoder from this loss (see Fig. 1). Allowing gradients to directly flow from loss to encoder hampers the training and leads to exploding/vanishing gradient problem, as we show in Section V. This is in agreement with [13]. Our proposition is that forcing z_t to support three tasks (value, policy, and next-observation) results in richer, more robust latent representations. The specific environments and training procedure are explained in the next section.

IV. Experimentation

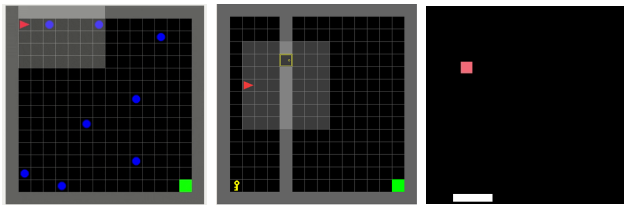
A. Environments

We evaluate the effectiveness of our approach on two types of environments: partially observable (Dynamic Obstacles and DoorKey from the Minigrid suite [38]) and fully observable (Catcher from pygame suite [39]).

1) MiniGrid: It features a series of highly structured environments of increasing difficulty and partial observability. Each environment features a task in a 2D gridworld setting and shares a discrete 7-dimensional action space (Forward, Turn Left, Turn Right, Open Door/Box, Pick Up Item, Drop Item, Done). We use a 3-channel partially observed integer state consisting of a 7×7 grid size. The 3 channels represent the object, its color and its status (for example, is a door open or closed). The environment has multiple entities (agent, walls, lava, boxes, doors, keys); objects can be picked up, dropped, and moved around by the agent; doors can be unlocked by keys of the same color (which might be hidden inside boxes). The agents cannot see past walls and closed doors. The objective is to reach a goal state in the quickest amount of time (which is captured by assigning to the goal state a reward which decays over time). We evaluate our approach on two Minigrid environments: Dynamic Obstacles and Doorkey.

Dynamic Obstacles has randomly moving obstacles as shown in Fig. 2. The agent receives a positive reward if it reaches the goal, a negative reward if it comes in contact with any of the obstacles (in addition to episode termination) and zero reward otherwise (timeout). The reward decreases the longer it takes to reach the goal state. **DoorKey** is a hard exploration task, where an agent has to learn to pick up a key, open a door, and move to another room to reach the goal state (see Fig. 2). The terminal rewards are similar to the Dynamic Obstacles environment.

2) Catcher: It is a relatively simple environment, where the agent (paddle) has to catch falling tiles (illustrated in Fig. 2). The action space is of size 3 (left, right, do nothing), and the agent receives +1 reward for catching each tile and -1 if it misses a tile. The episode ends when the agent misses 3 tiles. We consider for our experiments a grid size of 32×32. This environment is fully observable, but the state is more complex as we use raw pixel inputs rather than structured features. The



(a) (b) (c)

Fig. 2: Sample grids for (a) Dynamic Obstacles , (b) Doorkey and (c) Catcher. (a-b) The agent is denoted by a red triangle, the goal is a green square, and the moving obstacles are blue circles. Solid grey cells are walls. The agent can only observe information in the light grey cells. The Doorkey environment also features a key and a door in the wall. c) The red "brick" is generated at a random point on the top edge and falls in every time step, and must be "caught" by the white line.

state is a 3-channel RGB image of size 32×32. We choose the more challenging input type to better understand the capability of condensing information into a dense state.

B. Training Details

We test two instances of each environment, one of small grid size and another of large grid size. In addition, we report our results over two settings of entropy values. We keep entropy coefficient (β) of 0.01 in the first case and we exponentially decay from 0.2 to 0.01 in the latter case. We have presented all our results with PPO algorithm [33] as the base RL algorithms.

For all our experiments, we show the mean and 95% confidence interval (shaded region) of the average return across 10 different seeds for smaller grid size and 5 random seeds for larger grid size (due to computational constraints). All plots show a trailing average of 1000 episodes. We train all variants in all environments with 16 parallel actors, a learning rate of 0.001, a discount factor γ of 0.99 and a generalised advantage estimator λ_{GAE} of 0.97.

In addition to results with PPO, we also show results with A2C for Dynamic Obstacles with a smaller grid size. This quantitatively substantiates our claim of being compatible with any off-the-shelf algorithm. We validate our approach against two baselines: vanilla PPO (where $\eta = 0.0$, no effect of L_t^{NL}) and Self imitation learning [5]. We have chosen values of η as $0.1, 0.5, 1.0$ and 10.0, with 10.0 being the extreme case which can give us insights about effects of large η value. A study with more granular values of η is done in Section VI.

V. Results and Discussion

In this section, we present the results of experiments conducted on the 3 environments: Doorkey, Dynamic Obstacles, and Catcher. We start by explaining intuitively what effect our proposed architecture has on learning.

A. Dynamic Obstacles

A consolidated set of results is shown in Table I for all Minigrid environments, including Dynamic Obstacles and Doorkey. We first describe results in Dynamic Obstacles. Training plots are generated with moving average of 1000 episodes.

TABLE I: Training results for all Minigrid environments with PPO+FoLaR. Two baselines are included: $\eta = 0$ is vanilla PPO, and we also show results for self-imitation learning (SIL). Mean and best values are over the last 1000 episodes.

DoorKey								
Algo.	6x6 (small)				8x8 (large)			
	$\beta = 0.01$		$\beta = 0.2 \rightarrow 0.01$		$\beta = 0.01$		$\beta = 0.2 \rightarrow 0.01$	
	mean	best	mean	best	mean	best	mean	best
η =0	0.929	0.955	0.646	0.955	0.001	0.002	0.001	0.002
η =0.1	0.934	0.958	0.935	0.955	**0.571**	**0.966**	0.382	**0.963**
η =0.5	0.935	0.955	0.935	0.956	0.001	0.007	0.016	0.082
η =1	0.935	0.957	0.931	0.955	0.384	**0.966**	**0.571**	**0.963**
η =10	0.936	0.958	0.660	0.952	0.001	0.003	0.001	0.004
SIL	0.934	0.957	0.659	0.953	0.001	0.002	0.001	0.005
Dynamic Obstacles								
Algo.	6x6 (small)				16x16 (large)			
	$\beta = 0.01$		$\beta = 0.2 \rightarrow 0.01$		$\beta = 0.01$		$\beta = 0.2 \rightarrow 0.01$	
	mean	best	mean	best	mean	best	mean	best
η =0	-0.003	0.008	0.834	**0.950**	-0.005	0.000	0.158	0.619
η =0.1	**0.894**	**0.949**	**0.931**	**0.950**	**0.679**	**0.949**	**0.879**	**0.955**
η =0.5	0.528	**0.949**	0.927	0.949	0.529	**0.949**	0.720	**0.955**
η =1	0.261	0.948	0.836	**0.950**	0.356	0.948	0.758	0.948
η =10	0.077	0.863	0.738	**0.950**	-0.164	0.000	-0.002	0.000
SIL	-0.005	0.000	0.836	**0.950**	-0.025	0.000	0.057	0.617

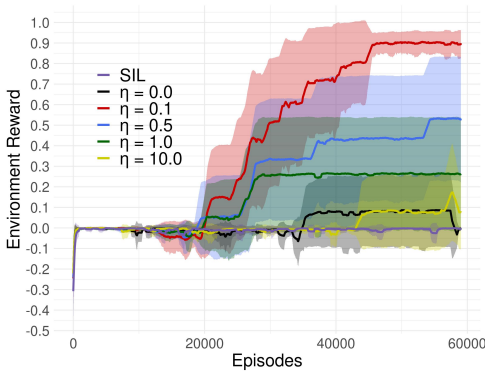


Fig. 3: Training results with PPO+FoLaR for Dynamic Obstacles with 6x6 grid size. β is kept constant at 0.01.

1) 6×6: Fig. 3 shows the training for the lower entropy setting (constant $\beta = 0.01$) on the 6×6 grid. We can clearly observe that $\eta = 0.1$ outperforms both baselines, not just in the expected rewards but also in rate of convergence. In addition, we can conclude that apart from $\eta = 10.0$ which is an extreme value for η, all other values perform better than the baselines. It is also important to note that in a dynamic partially observable environment and without explicit entropy encouragement, SIL is unable to learn even for small grid size.

This issue can be resolved by decaying β from a starting value of 0.2 to a final value of 0.01, with results shown in Fig. 4. There is still a small difference in the final expected reward as shown in Table I, but all the algorithms converge to ≈ 0.95. Lower values of η (0.1 and 0.5) perform slightly better than rest of the algorithms. The results suggest that having a higher entropy coefficient allows for for better exploration even in vanilla PPO, reducing the impact of η for this relatively simple case. We shall see greater distinction in subsequent discussion.

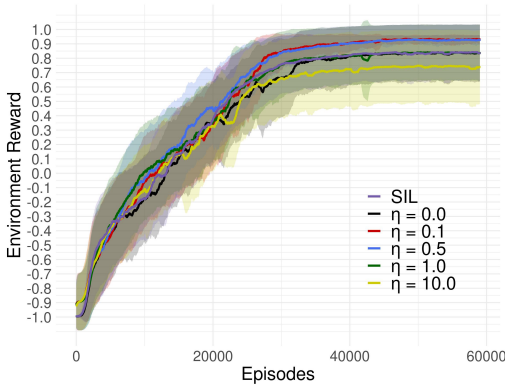


Fig. 4: Training with PPO+FoLaR for Dynamic Obstacles with 6x6 gird size. β is exponentially decayed from 0.2 by a factor of 0.999 per episode, clipping at 0.01.

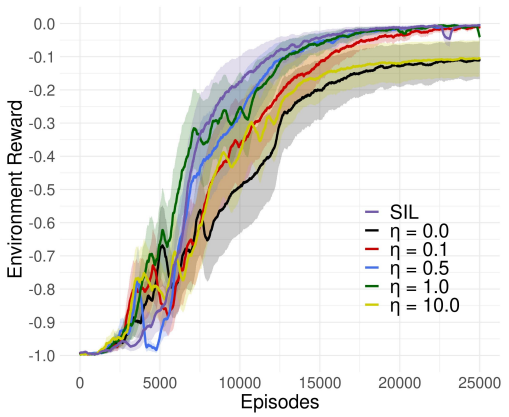


Fig. 5: Training with A2C+FoLaR for Dynamic Obstacles with 6x6 gird size. β is exponentially decayed from 0.2 by a factor of 0.999 per episode, clipping at 0.01.

Finally, we also demonstrate the versatility of the approach by augmenting the prediction network to advantage actor critic (A2C) with entropy maximisation. The results are shown in Fig. 5. We observe that all values of η apart from $\eta = 10$ outperform vanilla A2C. Note also that none of these curves are able to reach positive rewards, in contrast to Fig. 4. A reward of 0 indicates the ability to avoid obstacles (relatively simple task), but without reaching the goal.

2) 16×16*:* Fig. 6 shows the training curve for 16×16 grid size with fixed low entropy coefficient. We note that none of SIL, vanilla PPO or the high-η version of our algorithm are able to learn to reach the goal in 80000 episodes, and only acquire the skill of avoiding obstacles. Among the rest, $\eta = 0.1$ has the best performance, followed by $\eta = 0.5$.

While Fig. 4 failed to differentiate between the algorithms in a 6×6 grid with higher entropy, we see a marked difference for the 16×16 grid as shown in Fig. 7. Curves corresponding to $\eta = 0.1$, 0.5, and 1 are much better as compared to the baselines. Overall, we can observe consistent trend in

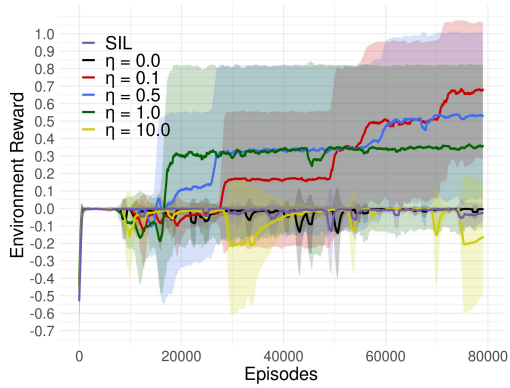


Fig. 6: Training with PPO+FoLaR for Dynamic Obstacles with 16x16 grid size. β is kept constant at 0.01.

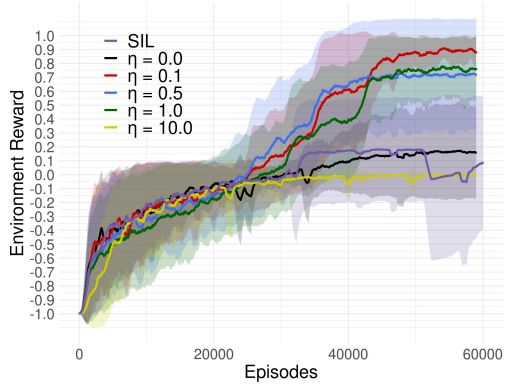


Fig. 7: Training with PPO+FoLaR for Dynamic Obstacles with 16x16 grid size. β is decayed from 0.2 by a factor of 0.999 per episode, clipping at 0.01.

Dynamic Obstacles, with $\eta = 0.1$ doing the best, followed by $\eta = 0.5$ and $\eta = 1$. Dynamic obstacles is a highly dynamic environment and having higher weight to the predictive loss L_t^{NL} leads to poorer performance. The probable cause is that the agent refers to remain in one place, which has more predictable observations. We shall find a different trend in the more predictable Catcher environment, as shown later.

B. Doorkey

1) 6×6*:* Fig. 8 shows the training graph for 6×6 size with entropy coefficient kept constant at 0.01. We observe that $\eta = 1$ and $\eta = 0.5$ perform best followed by $\eta = 0.1$ and $\eta = 0$ and SIL, and $\eta = 10$ performs the worst. The differences in learning curves are relatively minor for this small environment; this is probably because the 6×6 size is nearly the same as the observable window. In fact, Fig. 9 shows that the performance for $\eta = 10$, $\eta = 0$, and SIL degrades in the higher entropy coefficient training. FoLaR with any value apart from $\eta = 10$ is unaffected.

2) 8×8*:* On the other hand, there is a remarkable difference in results for the 8×8 environment as shown in Fig. 10. We

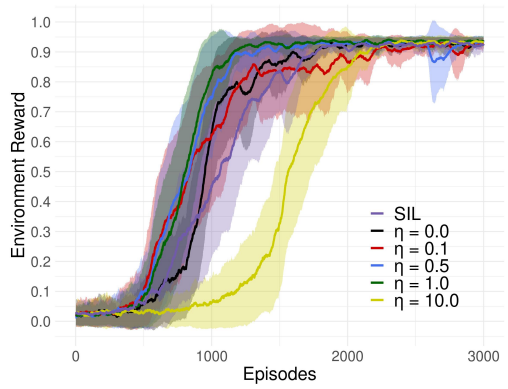


Fig. 8: Training with PPO+FoLaR for Doorkey with 6x6 grid size. β is kept constant 0.01.

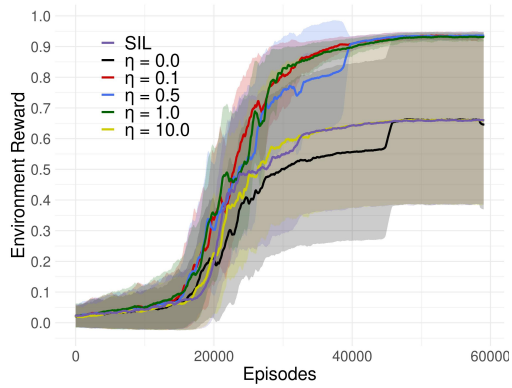


Fig. 9: Training with PPO+FoLaR for Doorkey with 6x6 grid size. β is decayed from 0.2 by a factor of 0.999 per episode, clipping at 0.01.

see that only FoLaR with $\eta = 0.1$ and $\eta = 1$ are able to solve this hard exploration task. The same trend is visible in Fig. 11, where β is exponentially decayed. We note that 8×8 DoorKey is typically not solvable from scratch by vanilla RL algorithms; the fact that FoLaR is able to do so is therefore encouraging.

C. Catcher

Fig. 12 and Fig. 13 show the results with the low and high (decaying) entropy scenarios as before, for the 32×32 Catcher environment with an 800 step timeout and normalised rewards. The variance for the low-entropy training case is very large. The high decaying β case indicates that FoLaR with the three higher η values performs well, but the variance is still high.

D. Hypothesis: What is η doing?

Intuitively, the losses in the value and policy networks in Fig. 1 directly target improved rewards, but this is not true of the prediction network. This fact raises the question of what effect the prediction loss (and its magnitude as defined by η) has on learning, if any. As we observe in

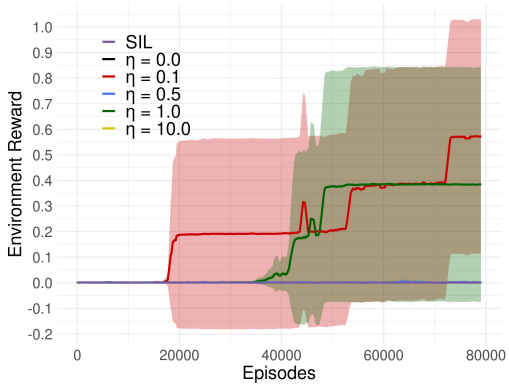


Fig. 10: Training with PPO+FoLaR for Doorkey with 8x8 grid size. β is kept constant 0.01.

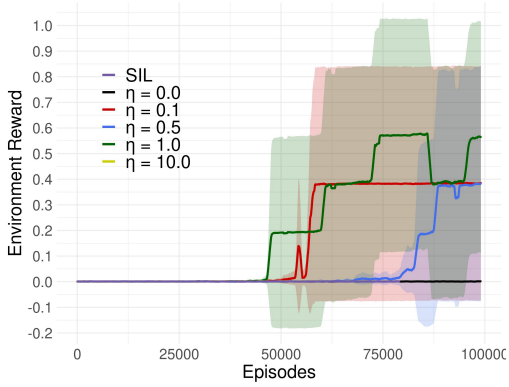


Fig. 11: Training with PPO+FoLaR for Doorkey with 8x8 grid size. β is decayed from 0.2 by a factor of 0.999 per episode, clipping at 0.01.

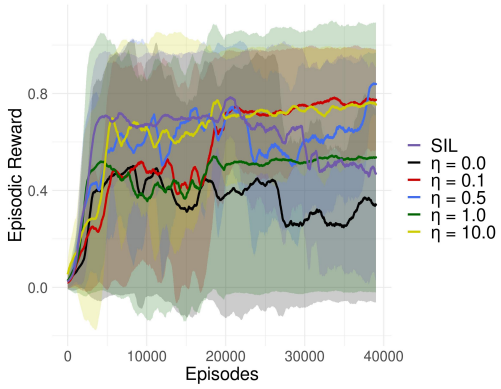


Fig. 12: Training with PPO+FoLaR for Catcher with 32x32 grid size. β is kept constant 0.01.

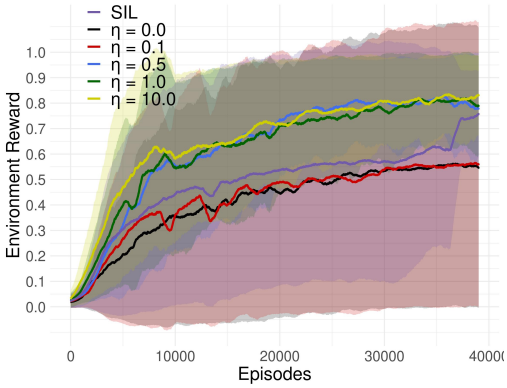


Fig. 13: Training with PPO+FoLaR for Catcher with 32x32 grid size. β is decayed from 0.2 by a factor of 0.999 per episode, clipping at 0.01.

this section, η has a definite positive effect on the success of learning; lower values of η are better in more dynamic environments, while higher values of η are better in more static or predictable environments. We therefore conclude that the prediction loss complements the value and policy losses by focussing exploration in regions with both high rewards and better predictability. In this regard, it has a more subtle effect than pure exploration (or higher entropy); while the latter prefers novelty for its own sake, prediction loss specifically latches onto high-reward, more predictable trajectories. More dynamic environments do not provide such trajectories easily, and high η in such cases leads to overemphasis on predictability. However, a small prediction loss nevertheless significantly helps focus exploration in more promising regions. In more predictable environments, it is easier to find predictable high-reward trajectories, and we can afford to work with a higher η. We also believe that use of additional loss term smoothens out the objective function landscape which is similar to how higher entropy achieves better results in hard exploration tasks as stated in [40], however, it does warrant more detailed experiments which we treat it as a one of direction for future work.

VI. Ablation Study

A. Effect of gradient propagation

As mentioned in Section III, we do not allow gradient propagation directly from L_t^{NL} to the encoder. For counterfactual analysis, we train the 6×6 DoorKey environment while allowing gradients to flow directly. The results are shown in Fig. 14. We can observe that for all the values of η, there is degradation in performance. As the value of η increases (larger gradients), degradation becomes much more prominent. This is also in line with the findings in [13].

B. Effect of η value

In order to understand the effect of η on expected rewards, we trained FoLaR on 6×6 Dynamic Obstacles with η ranging from 0.1 to 1. Fig. 15 shows the results. For each value of η it was trained for $60,000$ episodes same as figure 3. We can observe that low values of η provide better average rewards and also lower variance among the different random seeds. High values of η prove to be detrimental to the learning and overwhelm the policy and value loss during training, which reinforces our earlier hypothesis.

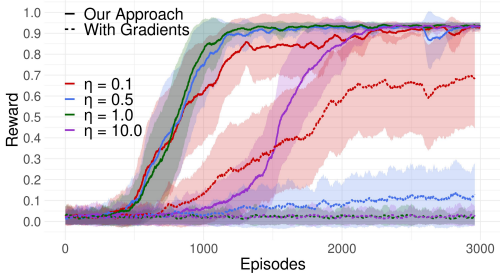


Fig. 14: Effect of propagating gradients to encoder from L_t^{NL} on DoorKey 6x6.

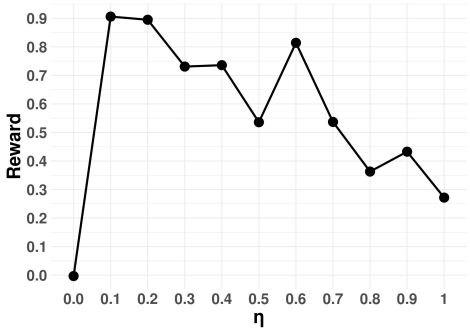


Fig. 15: Mean of last 1000 episodes of training in dynamic obstacles with 6x6 grid size, entropy coefficient β was kept constant at 0.01. Results are generated using 10 random seeds.

C. Testing on out-of-distribution environments

Fig. 16 shows the results of testing a previously trained policy (16×16 Dynamic Obstacles) on an out of distribution environment called Empty Room [38], which contains a single room with the goal (no obstacles or doors). We use the best-performing η for FoLaR, and compare with SIL and vanilla PPO. We observe that FoLaR performs significantly better when $\beta = 0.01$, which is also evident in earlier experiments, where higher entropy negated the gains introduced by η.

VII. Conclusion

In this paper, we presented FoLaR, a method to learn robust latent representations in partially observable environments. We showed that the methodology could be applied to any RL algorithm, and demonstrated results using PPO and A2C. The hyperparameter η controls the magnitude of prediction loss, and can be tuned based on the predictability of the environment, but most reasonable values result in superior

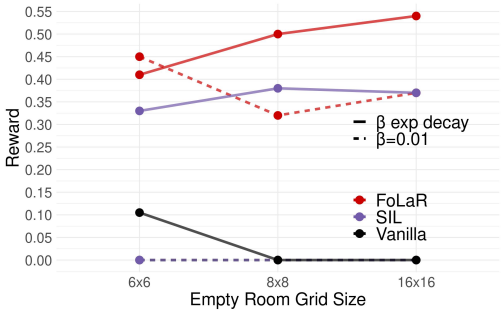


Fig. 16: Policies learned on 16×16 Dynamic Obstacles tested on three sizes of Empty Room without retraining. Rewards are averaged over 5 random seeds.

performance compared to baselines. FoLaR performs especially well in hard exploration tasks and larger grid sizes where entropy coefficient is kept static, indicating improved latent representations that lead to more focussed exploration. In future work, it would be interesting to look at adaptive η, which could learn better policies even faster.

REFERENCES

[1] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[2] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[4] J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan, "Approximate information state for approximate planning and reinforcement learning in partially observed systems," *arXiv preprint arXiv:2010.08843*, 2020.

[5] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," *arXiv preprint arXiv:1806.05635*, 2018.

[6] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018, pp. 2450–2462.

[7] D. R. Ha and J. Schmidhuber, "World models," *ArXiv*, vol. abs/1803.10122, 2018.

[8] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare, "Deepmdp: Learning continuous latent space models for representation learning," *arXiv preprint arXiv:1906.02736*, 2019.

[9] K. J. Astrom, "Optimal control of markov processes with incomplete state information," *Journal of mathematical analysis and applications*, vol. 10, no. 1, pp. 174–205, 1965.

[10] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[11] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," *arXiv preprint arXiv:2006.10742*, 2020.

[12] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, "Darla: Improving zero-shot transfer in RL," *arXiv preprint arXiv:1707.08475*, 2017.

[13] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-efficient reinforcement learning with self-predictive representations," in *International Conference on Learning Representations*, 2021.

[14] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine, "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model," *arXiv preprint arXiv:1907.00953*, 2019.

[15] M. Bellemare, W. Dabney, R. Dadashi, A. A. Taiga, P. S. Castro, N. Le Roux, D. Schuurmans, T. Lattimore, and C. Lyle, "A geometric perspective on optimal representations for reinforcement learning," in *Advances in Neural Information Processing Systems*, 2019.

[16] R. Dadashi, A. A. Taïga, N. L. Roux, D. Schuurmans, and M. G. Bellemare, "The value function polytope in reinforcement learning," *arXiv preprint arXiv:1901.11524*, 2019.

[17] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *ICML*. PMLR, 2019, pp. 2555–2565.

[18] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *IJCNN*. IEEE, 2010, pp. 1–8.

[19] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application," in *The 2012 international joint conference on neural networks (IJCNN)*. IEEE, 2012, pp. 1–8.

[20] N. Ferns, P. Panangaden, and D. Precup, "Metrics for finite markov decision processes." 2004.

[21] P. S. Castro, "Scalable methods for computing state similarity in deterministic markov decision processes," in *AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 069–10 076.

[22] P. S. Castro, P. Panangaden, and D. Precup, "Equivalence relations in fully and partially observable markov decision processes," in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[23] M. L. Littman and R. S. Sutton, "Predictive representations of state," in *Advances in neural information proc. systems*, 2002, pp. 1555–1561.

[24] S. P. Singh, M. L. Littman, N. K. Jong, D. Pardoe, and P. Stone, "Learning predictive state representations," in *International Conference on Machine Learning*, 2003, pp. 712–719.

[25] M. T. Izadi and D. Precup, "A planning algorithm for predictive state representations," in *IJCAI*. Citeseer, 2003, pp. 1520–1521.

[26] M. R. James, S. Singh, and M. L. Littman, "Planning with predictive state representations," in *International Conference on Machine Learning and Applications, 2004. Proceedings.* IEEE, 2004, pp. 304–311.

[27] M. Rosencrantz, G. Gordon, and S. Thrun, "Learning low dimensional predictive representations," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 88.

[28] B. Wolfe, M. R. James, and S. Singh, "Learning predictive state representations in dynamical systems without reset," in *International conference on Machine learning*, 2005, pp. 980–987.

[29] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber, "Solving deep memory pomdps with recurrent policy gradients," in *International Conference on Artificial Neural Networks*. Springer, 2007, pp. 697–706.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[31] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 1. IEEE, 1995, pp. 560–564.

[32] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optim. algorithms," *preprint arXiv:1707.06347*, 2017.

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018.

[35] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[36] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for openai gym," https://github.com/maximecb/gym-minigrid, 2018.

[39] N. Tasfi, "Pygame learning environment," https://github.com/ntasfi/PyGame-Learning-Environment, 2016.

[40] Z. Ahmed, N. Le Roux, M. Norouzi, and D. Schuurmans, "Understanding the impact of entropy on policy optimization," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, 09–15 Jun 2019, pp. 151–160.