

Learning in Multi-Agent Environments with Partial Observations via Self-Supervised Reconstruction and Prediction

Márton Szemenyei · Patrik Reizinger

Received: date / Accepted: date

Abstract Recent years saw numerous successes in deep reinforcement learning; still, learning in complex, multi-agent environments remains challenging, especially when cooperation between agents is necessary. Model-based structures and self-supervised learning offer potential avenues to tackle this difficulty. In this paper, new multi-agent environments are introduced, featuring cooperation and competition between agents, as well as imperfect, partial observations. Then, novel architectures employing two forms of self-supervised learning - long-term prediction and reconstruction - are proposed to improve SOTA agents' performance in these environments. The experiments presented in this paper demonstrate that the proposed architecture achieves significant improvements in the agents' performance.

Keywords Reinforcement learning · Neural networks · Self-supervised learning · Multi-agent environments

The research reported in this paper was supported by the ÚNKP-19-2 and ÚNKP-19-3 New National Excellence Program of the Ministry for Innovation and Technology and by Nokia Bell Labs Hungary.

M. Szemenyei (corresponding)
Dept. of Control Engineering and Information Technology
Budapest University of Technology and Economics
1117, Budapest, Magyar Tudosok krt. 2.
Tel.: +36-1-463-4025
E-mail: szemenyei@iit.bme.hu
<https://orcid.org/0000-0003-1397-6080>

P. Reizinger
Dept. of Control Engineering and Information Technology
Budapest University of Technology and Economics
<https://orcid.org/0000-0001-9861-0293>

1 Introduction

Reinforcement learning has undergone unprecedented evolution, resulting in matching - or even surpassing - human-level abilities in different complex tasks, such as computer games [1]. Considering the problems reinforcement learning tasks were applied to, their number has increased significantly, as benchmarks developed from a single paper-single task version to standardized benchmark suites with stable APIs [2,3]. Besides a self-induced standardization phenomenon, reinforcement learning tasks - mostly in the form of simulated environments, as cited above - got significantly more complex, e.g., by using 3D physics simulation [4,5].

However, these tasks generally abstract away from real-world scenarios in terms of simulating realistic failures of the processing pipeline (including partial observations, hidden objects, and classification errors of vision pipelines). Nonetheless, although we accept that for benchmark purposes providing an idealistic scenario can be beneficial, there is a need to evaluate reinforcement learning agents in more diverse scenarios.

In this paper, we propose a simulator environment for two tasks: robot soccer and autonomous driving in urban environments, focusing precisely on that point. A primary design goal for the environment is to keep it lightweight; thus, a two-dimensional physics backend is used. Still, the capability to provide observations with realistic disturbances such as partial observations or false positive/negative object sightings is added.

Utilizing the environments mentioned above, we propose novel neural architectures, able to handle the additional complexity of the tasks. Thus, we focus on long-term prediction (in terms of the latent space of the agents) and reconstruction. The former is done based on the paradigm of curiosity-driven exploration [6,7],

which incentivizes the agent to explore its state space. Nevertheless, we extend this paradigm as only one-step-look-ahead variants can be found in the literature [6, 8].

Furthermore, self-supervised learning [9, 10] is applied to reconstruct the features of the state space to compensate for the failures of the simulated vision system. The main advantage of the utilized paradigm is that training samples collected by self-play are from the same distribution. Besides that, as the simulator is rather lightweight, collecting those samples has a low temporal cost.

As part of the evaluation, the proposed methods are compared to actor-critic networks and the Intrinsic Curiosity Module (ICM) of [6] on the proposed robot soccer and autonomous driving environments.

2 Related work

Modern deep learning algorithms have achieved a significant performance improvement in the last years, resulting in near-human or superhuman skills in several tasks. As we strongly rely on the results of the literature, this section is dedicated to the review of relevant reinforcement learning and self-supervised learning methods.

2.1 Reinforcement Learning

A significant contribution of the field to widen the range of successful applications of the reinforcement learning paradigm was the introduction of the notion of intrinsic rewards. Among other model-based formulations, such as World Models [11], curiosity [6, 7] formulations were developed to incentivize the learning process of the agent, while generally incentivizing exploration of the environment. Another notion is that of disagreement [8], which can be thought of as an ensemble approach.

The primary motivation to utilize model-based approaches is the fact that the agent is able to construct an internal, latent representation of the environment. First, having a model makes it possible to decouple training from the real environment itself (self-play), which can accelerate training further - as in the case of simulators, which are run generally on the CPU. Whereas if the agent interacts with the real world, having a model can spare the immense cost and danger, the long training process can impose. Second, the internal model can be used to help researchers identifying critical points of the model, e.g. by determining possible reasons for the unsatisfactory performance.

Furthermore, as studies of curiosity-based methods [7] show, agents can successfully learn to perform different

tasks even if they do not receive rewards from the environment. Note that success is defined w.r.t. extrinsic rewards. Nonetheless, these methods can be generally used to improve performance even if extrinsic rewards are present.

The curiosity paradigm was extended in Reizinger et. al. [12] by utilizing the attention mechanism [13], which was proven to result in superior performance in different fields of deep learning. The main contribution of the work cited above [12] is the so-called Rational Curiosity Module (RCM), an adaptive curiosity loss formulation, which was even utilized in multi-agent environments [14].

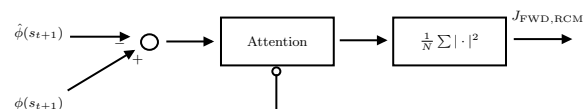


Fig. 1 Forward loss formulation of the RCM agent [12]

The main idea of RCM is - as shown in Figure 1 - to utilize the attention mechanism, conditioned on $\phi(s_{t+1})$ - where ϕ denotes the transformation of the state s_{t+1} into feature space and $\hat{\cdot}$ stands for predicted values. This way, it is possible to focus on a specific subspace of the features extracted by the neural network; and to use this weighted difference in the calculation of the loss of the forward model of the original ICM [6]. The rationale behind this idea is that although the features provide a compact representation of the whole state space, it is possible that in particular steps, different subspaces are relevant to be modeled. Thus, the curiosity loss should only incentivize the agent to learn a better model of those features.

As it was shown in Szemenyei et. al. [14], the RCM approach is equivalent to an adaptive weighted least squares formulation, where adaptivity is given by the attention mechanism. I.e., $J_{FWD,RCM}$ is equal to the following quadratic form:

$$J_{FWD,RCM} = \frac{1}{N} e_{t+1}^T e_{t+1} \quad (1)$$

$$e_{t+1} = \sqrt{W} \left(\phi(s_{t+1}) - \hat{\phi}(s_{t+1}) \right), \quad (2)$$

where W stands for the weights of the Attention network, as noted in [14].

For multi-agent tasks, solving the credit assignment problem is more complicated, as in that case, the agents' dimension increases the complexity of the problem. I.e. the question to be answered is not only that of which actions (in time) were responsible for a given result but

also which agents contributed. Among others, Foerster et. al. [15] proposes a counterfactual approach to mitigate this problem in multi-agent scenarios.

3 Environments

In the following, we describe the proposed environments, first introduced in our previous work [14]. The two OpenAI Gym-compatible [2] environments simulate robot soccer (RoboCup) and autonomous vehicles in an urban scenario (Driving). As already mentioned in the introduction, our focus was different from that of other environments. Thus, the main characteristics of both environments will be elaborated in this section, followed by a detailed description of the observation and reward systems.

In recent years, the number of available reinforcement learning benchmark frameworks increased in a significant manner. This was accompanied by including more complex scenarios, such as 3D physics simulation [3, 5], or very complicated tasks such as StarCraft II [1]. However, these environments expose an in specific ways ideal scenario to the agents. Although noise is present, realistic constraints (such as fog of war in the environment used in [1]) are not general. Furthermore, the number of multi-agent environments is not very high. There exist some for e.g. robot soccer [5], although it is computationally demanding.

Thus, we intended to implement a lightweight multi-agent scenario simulator for robot soccer and autonomous driving, while abstracting away from low-level physics (such as the control of robot joints). Instead, we decided to focus on features incorporating the failures of vision pipelines (e.g. false positive and false negative sightings). The choice for the environment types was based on the fact that both require cooperation within the team, although the setting is undoubtedly competitive. The environments are shown in Figure 2.

3.1 Observation types

As mentioned before, our aim was to create environments as realistic as possible while simultaneously being as efficient as possible, since the run time of the environment is one of the bottlenecks in reinforcement learning. Naturally, this means that our environment cannot afford expensive 3D graphics to render realistic frames; therefore, the environment provides abstract observations that an object detection algorithm would extract from the seen images.

However, in accordance with our goal to make the environment realistic, we also attempt to emulate the

failures of real computer vision pipelines, so that the agent may learn to deal with them. For this reason, both environments provide a partial observation type, in which the detected objects are returned in the agent’s own coordinate system, and the visibility of the objects is limited by their distance from the agent and - in the case of robot soccer - the field of view of the robot’s camera.

The robot soccer environment also provides an image-based observation type, which provides a semantically labeled image for each of the robot’s two cameras, aiming to emulate the output of semantic segmentation. Both of these observation types introduce the challenge of localization of both the agent and other dynamic objects in the absolute coordinate system.

Both observation types support the emulation of detection failures, including false negatives, false positives, misclassifications, and inaccurate localization. Objects may also be lost due to occlusion. The failure emulation feature also supports two modes: in the random mode, these errors occur with a fixed, adjustable probability. In the realistic mode, however, the probability of random errors depends on other factors as well, aiming to adjust the reliability of the observations depending on the situation. These additional rules are as follows:

1. The magnitude of position noise and the probability of false negatives increases with the distance from the agent.
2. False negatives are more likely to occur when two or more objects are nearby or at the edge of the field of view.
3. Visually similar objects (such as balls and penalty crosses or vehicles and obstacles) are more likely to be confused than dissimilar ones.
4. False-positive sightings are more likely to occur in certain places (such as false-positive balls at the feet of other nearby robots).

3.2 Reward system

The reward system implemented in both environments aims to encourage the agents to make moves towards reaching the end goal. The types of achievable rewards can be organized into three categories: **primary**, **rule-based**, and **auxiliary**.

Primary rewards are the simplest to describe: the agents score for reaching or failing their goals, e.g. scoring or reaching the destination in the two environments, respectively. Rule-based rewards are usually awarded for breaking a rule, and as such, they are usually negative. These rules include pushing or walking off the field

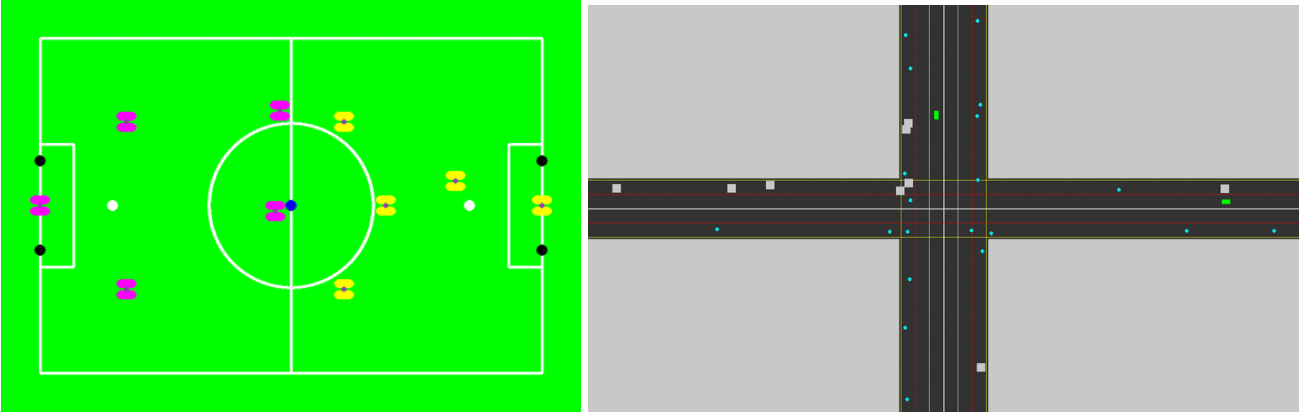


Fig. 2 The RoboCup (left) and Driving (right) environments.

in robot soccer, and crashing or moving off the road in the autonomous driving environment.

Auxiliary rewards are the trickiest, as they are awarded for actions that usually increase the probability of success, yet they are not necessarily the optimal actions. As such, the magnitude of these rewards is considerably smaller so that a later primary or rule-based reward could compensate for them. Agents can get auxiliary rewards for moving towards their goal or negative ones for driving in the opposite lane. In the robot soccer setting, getting closer to the ball or moving the ball towards the opponent’s goal is also rewarded.

In the robot soccer setting, agents are also awarded an auxiliary reward, called the **observation** reward. The magnitude of this reward increases with the number of *different* objects seen in a given interval. The magnitude also depends on the type of object seen: ball sightings are worth the most, followed by robots and field objects. This reward aims to encourage the agent to move the robot’s head to collect useful observations.

4 Model

The environments introduced in the previous section present a few novel challenges, necessitating the design of a custom model architecture. To this day, there are few environments [4] that provide agent-centric partial observations, and even those only account for occlusion. Our environment, on the other hand, introduces false negative and positive errors along with misclassifications, which renders traditional architectures that expect a fixed number of objects infeasible.

Moreover, the robot soccer environment introduces one further challenge: the Nao robots used in real-world games are relatively clumsy, meaning that executing a single action (such as moving, turning, or kicking) takes considerable time, usually between 500 – 1000 ms. This

results in the vision pipeline running at a much higher frame rate, breaking the single action-single state orthodoxy of reinforcement learning. Indeed, performing a single action in this environment results in multiple observations at different intermediate time steps.

4.1 Encoder

For the reasons detailed above, the proposed encoder architecture has to tackle noisy, incomplete observations and effectively combine them over multiple time steps. To manage this problem, the encoder first embeds every observed object using a small Multi-Layer Perceptron (MLP) independently. Each object type has its unique embedder MLP; however, the number of outputs are equal. This allows these embeddings to be concatenated to form a varied-length sequence of embeddings for each time step.

The first step is to allow the network to learn how objects in a single observation interact. This is essential since the objects in the game interact in various ways, such as causing false positives/negatives or occlusion, among others. Also, the presence of particular static objects means that other such objects should be nearby. To allow the model to learn these, a self-attention layer is applied to the sequences in every time step independently (these layers share weights, naturally).

The next step of the encoder architecture is to combine the observations at different time steps into a single sequence. Naturally, interactions between elements of these sequences are integral to this step as well. For this task, we used a new type of layer called Recurrent Temporal Attention (RTA). This module is a multi-head attention layer, where the two input sequences are the observations at two subsequent time steps. Once the output sequence is produced, it is fed back as one of the inputs, with the next observation sequence tak-

ing up the other input. The architecture is shown in Figure 3.

It is easy to see that this is simply an RNN cell, except that it uses a multi-head attention layer in place of a linear one. Naturally, there are a few further considerations: first, when combining the sequences of two subsequent time steps, the model must avoid losing objects due to false-negative sightings. To achieve this, padding is applied, which results in the final output of the RTA module having as many elements as its longest input.

However, this guarantees that false-positive sightings will remain in the sequence. To allow the model to filter these out, a simple linear layer with sigmoid activation is applied to the elements of the final output sequence, producing a confidence score for every object. Then, as the encoder’s final step, the objects are averaged, using the confidence scores as relative weights, producing a single embedding vector for the observations. The encoder is shown in Figure 4.

4.2 Localization and reconstruction

Notably, in some environments, the rules of the game depend on the *absolute* position of the relevant objects, as opposed to their position relative to the agent. Soccer is a prime example of this; thus, localization is required to avoid scoring own goals. The observations in our environments are inherently relative; however, it is easy to see how this would make learning difficult.

Two auxiliary outputs are introduced to circumvent this difficulty, aiming to help the agent to learn absolute localization and detection. The first of these outputs is called the localization output, where the agent is trained to accurately predict its own position and orientation at every step in a self-supervised manner. Note that robots can move their head from their body independently, so predicting the robot pose involves determining both orientations.

In addition to the localization output, there is also a reconstruction head, aiming to accurately predict the position and other properties of dynamic objects (the ball and other robots in this case) in the environment. This output helps the agent to learn to determine the absolute position of other relevant objects - thus, it aims to reconstruct the state vector of the environment, hence its name -, while it also explicitly trains the model to reduce position noise and to account for false negatives and positives.

When predicting multiple objects of the same type, such as robots, the reconstruction head uses anchors, similarly to YOLO [16,17]. In this case, anchors are

fixed to certain positions on the field, and objects are assigned to anchors to minimize the sum of the distances between the object and anchor positions via the Hungarian algorithm. Notably, during reconstruction, the agent is only trained to predict objects it has seen at least once before during the rollout.

4.3 Long-term forward prediction

In many cases, it is crucial that the agent is able to plan over a longer time horizon. Although several complex tasks were solved with reinforcement learning where the credit assignment - and, thus, a planning ability over a longer time horizon - was managed in a satisfactory manner. Nonetheless, in terms of intrinsic rewards, methods generally focus on explicit short-horizon predictions, such as in the case of the ICM [6] or disagreement [8]. Regarding the World model architecture, as it contains an Recurrent Neural Network (RNN), it is implicitly learning over a longer horizon, according to the capabilities of the recurrent cell. However, this is - in our assessment - an implicit goal of the model.

Thus, we propose a network explicitly conditioned on learning to be able to model the environment over a longer time-horizon (chosen as the length of a rollout in our experiments). The general approach is to do it step-by-step, i.e., estimating

$$\begin{aligned} & p\left(\hat{\phi}(s_{t+1})|\phi(s_t), a_t\right), \\ & p\left(\hat{\phi}(s_{t+2})|\phi(s_{t+1}), a_{t+1}\right), \\ & \vdots \\ & p\left(\hat{\phi}(s_{t+N+1})|\phi(s_{t+N}), a_{t+N}\right) \end{aligned}$$

Or even incorporating each available state in the predictions as

$$p\left(\hat{\phi}(s_{t+i})|\phi(s_{t+i-1}), \dots, \phi(s_t), a_{t+i-1}\right),$$

where $i \in \{1, \dots, N+1\}$,

with N denoting the length of the rollout; this way, we would only enforce one-step predictions over the rollout.

Thus, our proposal is to feed back the predicted states $\hat{\phi}(s_i)$ in an autoregressive manner; hence (note the hat symbol denoting predicted values):

$$p\left(\hat{\phi}(s_{t+i})|\hat{\phi}(s_{t+i-1}), \dots, \hat{\phi}(s_{t+1}), \phi(s_t), a_{t+i-1}\right),$$

where $i \in \{1, \dots, N+1\}$. (3)

It is clear that by optimizing the objective of Equation 3, the agent is promoted to develop a modeling

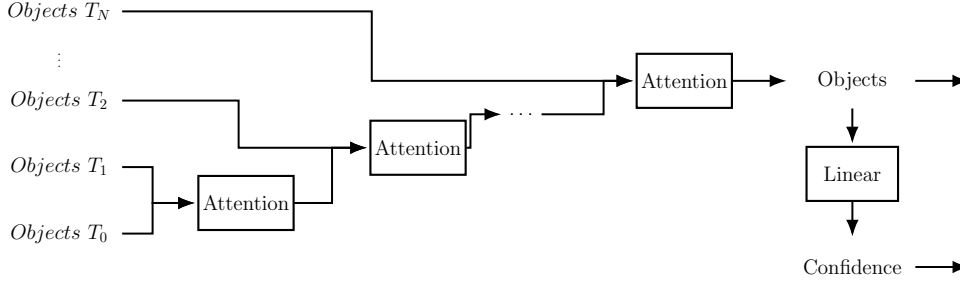


Fig. 3 The Recurrent Temporal Attention (RTA) module.

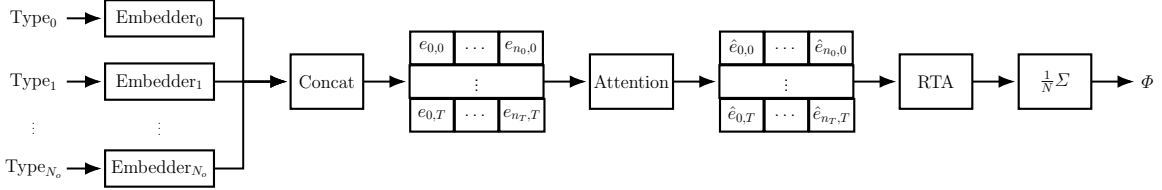


Fig. 4 The encoder architecture. Here $\hat{\cdot}$ is only used to denote that attention was applied to the embedded objects.

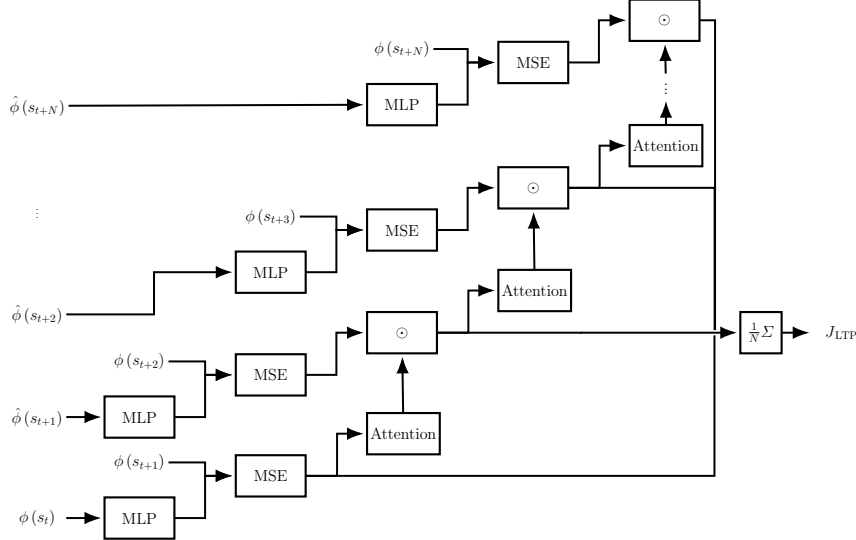


Fig. 5 The Long-term forward prediction module. The output of each MLP module is $\hat{\phi}(s_{i+1})$, given $\hat{\phi}(s_i)$ (or $\phi(s_i)$) as input. The $\odot$ denotes the Hadamard-product.

capability that provides predictions over multiple time steps. Nonetheless, there is a reason why the formulation above is generally avoided in the literature, namely, that of error accumulation. If $\hat{\phi}(s_{t+1})$ is erroneous (to a certain extent, and it will be), then - even if $p(\hat{\phi}(s_{t+2}) | \hat{\phi}(s_{t+1}), \dots)$ is of the same quality as the prediction of $\hat{\phi}(s_{t+1})$ is - loss functions will assign a higher loss to predictions in further time steps. even if the model would have made a more accurate prediction, given $\hat{\phi}(s_{t+1})$ as an input.

Thus, we propose an adaptive loss formulation, that - we conjecture - is able to compensate for this error accumulation (i.e., to consider only the error of the prediction, not that of the given inputs as well). After cal-

culating all the predictions (with MLPs; hence, we do not suffer from the slower speed of RNNs), the mean squared error-like loss function calculated with the help of a single attention network (the same is used for all predictions) is:

$$J_{LTP} = \frac{1}{N} \sum_{i=t+1}^{t+N+1} J_{LTP,i} \quad (4)$$

$$J_{LTP,i} = \sum_{i=t+1}^{t+N+1} \text{Attention}(W_i) \odot [\phi(s_i) - \hat{\phi}(s_i)]^2 \quad (5)$$

$$W_i = \begin{cases} [\phi(s_{i-1}) - \hat{\phi}(s_{i-1})]^2, & i > t+1 \\ 1, & i = t+1, \end{cases} \quad (6)$$

where LTP is a shorthand for long-term prediction, and W_i has the same dimensionality as the state tensors ($\odot$ denotes the Hadamard-product). The case for W_i in case of $i = t + 1$ expresses the fact that $\hat{\phi}(s_{t+1})$ in only conditioned on $\phi(s_t)$; hence, there is no need for compensating error accumulation in that step.

5 Results

The proposed methods have been evaluated and compared against baseline algorithms to demonstrate their efficacy. The evaluation was performed for both environments, using the partial observation mode and realistic noise with a magnitude of 1.0. We used two agents per environment in the autonomous driving case, and two agents per team for robot soccer, and the training was run with 8 parallel environments.

We used a relatively simple metric for evaluation: for every training run, we computed the best average reward achieved by the agents (the average was computed using a 10 episode moving window). For the robot soccer environment, the best average observation reward was also computed.

In the end, two agent types were compared: The baseline agent was an A2C [18] architecture extended with the RCM module, while 'our' method included the long-term prediction extension as well. Also, for the robot soccer environment, we trained an agent with both the long-term prediction and the reconstruction modules enabled. Arguably, the reconstruction module offers little gain for the autonomous driving environment; thus, it was not tested in that case.

The training was performed with 8 different random seeds, in order to make sure that the differences between the architectures were not due to a lucky random seed. Finally, a Bayesian t-test was used to determine whether the improvement provided by the proposed methods is significant. For performing the test, the independent samples version from the BEST [19] was used, since having the same random seed does not necessarily make two runs directly comparable.

Notably, the network used in the reconstruction variant was pre-trained for localization and reconstruction, using randomly generated samples from the environment. These samples were N step sequences, where N is the rollout size. To measure the effectiveness of the pre-training, the Average Precision (AP) was measured at different threshold values. Table 1 shows the results of pre-training for localization and the reconstruction.

Notably, localization achieves far better results than reconstruction. The reason for this is that for distant objects, even small errors in determining the robot's

Table 1 The AP achieved for localization and reconstruction.

Distance threshold	25 cm	50 cm	1 m
Localization	83%	100%	100%
Reconstruction	54%	70%	92%

orientation can result in significant errors in absolute localization. To help putting the accuracy in perspective, the soccer field is 9×6 meters in size.

5.1 Effects of prediction

To test the effectiveness of long-term prediction, three scenarios were tested: the Driving environment using partial noise, while the RoboCup environment was tested with full and partial observations. We performed the Bayesian t-test on the results and computed three essential characteristics. The first is the 95% confidence interval (CI) on the difference of means, the second is the mean of the CI, while the third is the Probability of Improvement (PoI). This last measure is simply the probability of the effect size being positive (in this study, a positive effect size consistently means that the proposed method outperforms the baseline). For reference, the baseline reward (R_b) is also shown.

Table 2 Results of the Bayesian t-test.

Environment	95% CI	Mean	PoI	R_b
RoboCup	[0.01, 0.35]	0.18	97.4%	13.06
RoboCup (Full)	[0.14, 0.76]	0.47	99.7%	19.28
Driving	[0.89, 5.89]	3.51	99.55%	22.96

The results (Table 2) show that the long-term prediction agent credibly outperforms the baseline in all three cases, although its effect in the two RoboCup scenarios is rather small. In the Driving case, however, it was able to provide a sizeable, 15% improvement.

5.2 Effects of reconstruction

To examine the effects of the localization and reconstruction module, the RoboCup environment was used with partial observations. Since this model also uses the long-term prediction module, comparisons against the standard baseline alone are not sufficient. To be certain that the improvements are due to the reconstruction module, this model is compared against both the standard and the prediction-only baselines. The results of these two t-tests are shown in Table 3. For the sake of brevity, the model using long-term prediction is abbreviated as **Pred**, while the model using both prediction

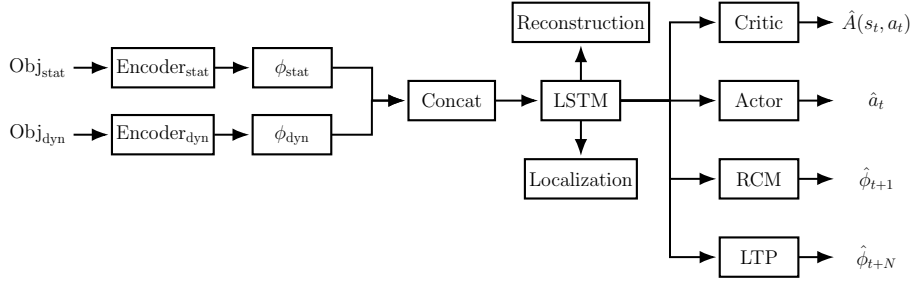


Fig. 6 The network architecture. The subscripts stat and dyn denote components (objects, encoder, and features) for static and dynamic objects, respectively. $\phi(s_t)$ stand for features, a_t for actions, while A for the advantage function; whereas $\hat{\cdot}$ denotes predicted values.

and reconstruction is referenced as **Recon**. The baseline method is denoted as **RCM**.

Table 3 Results of the Bayesian t-test.

Comparison	95% CI	Mean	PoI	R_b
RCM vs Recon	[2.22, 2.6]	2.41	100%	13.06
Pred vs Recon	[2.01, 2.44]	2.23	100%	13.25

The results clearly show a significant increase in the rewards against both baselines. Indeed, the difference between the standard and the prediction-only baselines is negligible compared to the 19% improvement provided by the reconstruction.

The same comparisons were made for the observations reward as well, to see if explicitly requiring the model to be able to tell the locations of relevant objects helps to develop efficient strategies to observe them as frequently as possible. The results shown in Table 4 show a clear, 8% improvement against both baselines.

Table 4 Results of the Bayesian t-test for the observation rewards.

Comparison	95% CI	Mean	PoI	R_b
RCM vs Recon	[0.36, 0.58]	0.47	100%	5.94
Pred vs Recon	[0.35, 0.58]	0.46	100%	5.95

Finally, as the best agent is found, we also briefly compare its performance against the random policy and the human baselines. The results in Table 5 show that the Reconstruction agent vastly outperforms the random baseline; however, it is also quite far from the human baseline, especially, since the human baseline was made by the authors, who can hardly be considered experts in playing these environments.

6 Conclusion

In this paper, two novel reinforcement learning environments were introduced for robot soccer and autonomous

Table 5 Comparison against the random and the human baselines.

Baseline	Random	Human	Pred	Recon
RoboCup	5.75	22.57	13.25	15.48
Driving	0.7	42.3	26.47	N/A

driving. Both environments are based on 2D physics simulation, and abstract over low-level control and perception, to allow researchers to focus on the behavioral aspect of learning. Nonetheless, the environments simulate the common errors of vision pipelines to allow agents to learn how to deal with noisy and incomplete inputs. Both environments provide simultaneous multi-agent cooperation and competition scenarios.

To solve learning in these environments, a novel architecture was proposed, including two forms of self-supervised learning to allow agents to learn with better sample efficiency. The first of these was the localization and reconstruction module, aiming to teach the agent to convert partial observations into an understanding of the full scene. The second was the long-term prediction module, which aims to allow the model to consider medium- to long-term consequences of its actions.

Both of these extensions were tested against the Rational Curiosity baseline proposed in our previous work. The results of the Bayesian t-test performed using multiple runs with different random seeds demonstrated that the proposed extensions improve the agents' performance with over 95% significance level, although the actual effect size is small in some of the cases. Although our agents far outperform the random policy, there is still a long way to go before reaching human performance levels.

To make our work reproducible, the environments and the code used for training are available as part of the DynEnv Python package, while the source is available at <https://github.com/szemenyeim/DynEnv>.

Conflict of interest

The authors declare that they have no conflict of interest.

References

- O. Vinyals, I. Babuschkin, W.M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D.H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J.P. Agapiou, M. Jaderberg, A.S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T.L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver, *Nature* **575**(7782), 350 (2019). DOI 10.1038/s41586-019-1724-z
- G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba,
- M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli, S. Henderson, A. Novikov, S.G. Colmenarejo, S. Cabi, C. Gulcehre, T.L. Paine, A. Cowie, Z. Wang, B. Piot, N. de Freitas, arXiv preprint arXiv:2006.00979 (2020). URL <https://arxiv.org/abs/2006.00979>
- B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, I. Mordatch,
- S. Liu, G. Lever, J. Merel, S. Tunyasuvunakool, N. Heess, T. Graepel. Emergent coordination through competition. URL <https://arxiv.org/pdf/1902.07151>
- D. Pathak, P. Agrawal, A.A. Efros, T. Darrell, in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (IEEE, 2017). DOI 10.1109/cvprw.2017.70
- Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, A.A. Efros,
- D. Pathak, D. Gandhi, A. Gupta,
- A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, S. Levine, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Curran Associates Inc., Red Hook, NY, USA, 2018), NIPS'18, p. 9209–9220
- P. Sermanet, C. Lynch, J. Hsu, S. Levine, in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2017), pp. 486–487
- D. Ha, J. Schmidhuber, in *Advances in Neural Information Processing Systems 31*, ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Curran Associates, Inc., 2018), pp. 2450–2462. URL <http://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.pdf>
- P. Reizinger, M. Szemenyei, in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2020). DOI 10.1109/icassp40776.2020.9054546
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin,
- M. Szemenyei, P. Reizinger, in *2019 International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)* (2019), pp. 176–181. DOI 10.1109/ICCAIRO47923.2019.00035
- J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson,
- J. Redmon, A. Farhadi, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2017). DOI 10.1109/cvpr.2017.690
- J. Redmon, S. Divvala, R. Girshick, A. Farhadi, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2016), pp. 779–788. DOI 10.1109/cvpr.2016.91
- V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Harley, T.P. Lillicrap, D. Silver, K. Kavukcuoglu, in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (JMLR.org, 2016), ICML'16, p. 1928–1937
- J.K. Kruschke, *Journal of Experimental Psychology: General* **142**(2), 573 (2013). DOI 10.1037/a0029146