# Online Junction State Estimation for Reinforcement Learning with Intrinsic Motivation

Luka Marinko
Fakulteta za računalništvo in informatiko, Univerza v Ljubljani
Ljubljana, Slovenia
luka4454@gmail.com

Jan Babič
Jožef Stefan Institute
Ljubljana, Slovenia
jan.babic@ijs.si

Benjamin Fele
Jožef Stefan Institute
Ljubljana, Slovenia
benjamin.fele@ijs.si

## Abstract

Facilitating exploration remains a central challenge in reinforcement learning of both humans and artificial agents, particularly in environments where informative feedback is limited. A recently introduced intrinsic motivation reward based on junction state estimation improves agent's exploration by modeling the distribution of future states and using its entropy as a learning signal. However, the original approach relies on an offline dataset collected before training, making its performance sensitive to the quality and coverage of the initial data.

We propose an online variant of junction state estimation that removes this limitation by updating the underlying model directly during learning. The approach jointly trains a one-step dynamics model to generate synthetic interaction data and uses a progressive sampling schedule to gradually combine initially collected environment data with synthetic model rollouts.

Experimental results show improved sample efficiency in a limited data environment compared to the original offline approach, and that the progressive sampling schedule is crucial for robust reward model learning. The integration of the reward model and reinforcement learning makes solving a given task possible without requiring a multi-stage experimental setup.

## Keywords

junction state estimation, neural networks, reinforcement learning, intrinsic motivation

## 1 Introduction

Reinforcement learning (RL) in environments with sparse rewards remains challenging due to the difficulty of efficient exploration. A widely studied group of approaches for addressing this problem is intrinsic motivation (IM), where agents receive additional rewards that are separate from the task reward and encourage exploration of potentially useful states [7, 1]. From a cognitive science perspective, IM can be understood in terms of curiosity-driven information seeking, where agents are motivated to explore states that reduce uncertainty and support learning [8]. Even though the purpose of IM is similar in human and artificial learning scenarios, designing stable and informative motivation signals in the latter case remains difficult, especially in the presence of computational constraints [1].

A recent approach to IM uses junction state estimation to identify states leading to diverse future outcomes. The Door(s) framework estimates the distribution of future states and uses its entropy as an intrinsic reward signal [3]. The underlying Mixture Density Network (MDN) [2] is trained on a fixed dataset collected with uniformly random actions during an initial exploration phase [3]. Consequently, the quality and coverage of this dataset can affect the resulting reward signal.

In this work, we propose an online variant of this junction state estimation approach which removes the need for a separate offline pretraining phase. The reward model is trained continuously during RL, together with an added dynamics model that predicts one-step environment transitions. The dynamics model is used to generate synthetic rollouts which are combined with collected environment data for reward model training. A progressive data mixing schedule is implemented to gradually introduce synthetic data, thus reducing the impact of model inaccuracies early in training.

We evaluate the proposed method in the `Pendulum-v1` environment [10], a standard RL benchmark for the inverse pendulum stabilization task, and compare it to the original offline approach. The results show improved sample efficiency in an environment with limited initial data, suggesting that continuously updating the junction state estimation model can be used for concurrent learning of both the junction state estimation model and the given task.

## 2 Related Work

Existing research on human learning shows that learners often maximize more than a given task objective [11]. For example, when learning a motor task they can also minimize the amount of effort for its completion, or perform actions that minimize or maximize the certainty of success. Additionally, belief in control over future outcomes (i.e., self-efficacy) also shapes how people learn [12]. While Door(s) doesn't quantify learner's control, it does quantify the diversity of expected future outcomes. This quantification can be seen as a kind of an auxiliary objective to be optimized in addition to primary learning goals [12].

The present paper studies exploration of artificial agents. In this scope, additional rewards in RL with sparse rewards can provide more informative feedback and shape agent behavior [6]. In particular, intrinsic motivation methods can encourage exploration by promoting visitation of novel states or maximizing predictive error [1, 7].

Our work belongs to approaches that use information theory to formalize intrinsic motivation signals. Such approaches commonly use entropy or mutual information to characterize aspects of agent behavior or environment dynamics [1]. Door(s) [3], which forms the basis of our work, uses the entropy of predicted future-state distributions to identify junction states and encourage exploration of states with diverse possible outcomes.

The training scheme in [3] uses a fixed dataset to pretrain the reward model, which is then kept unchanged during RL. This makes the resulting reward signal dependent on the coverage of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*Information Society 2026, Ljubljana, Slovenia*
© 2026 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2026.cogni.86

the initial data and prevents adaptation to newly explored regions. Pretraining also relies on uniformly random actions rather than the distribution induced by the learned policy. In contrast, we update the reward model during RL and use a learned dynamics model to generate synthetic rollouts, allowing the shaping signal to evolve throughout training.

## 3 Method

In this section, we first describe the original offline junction state estimation method used as a baseline. We then present our proposed method, where the reward and the environment dynamics models are jointly updated during RL. Finally, we describe the generation of synthetic rollouts, the progressive mixture ramp used to combine initially collected environment data with synthetic data, and the activation of reward shaping during training.

### 3.1 The Baseline Method for Junction State Estimation

The baseline method estimates locations of junction states in an environment, and aims to characterize the distribution of future states reachable from a given state. A reward model using an MDN [2] conditioned on the current state $s$ and the prediction time horizon $h$ is trained to approximate this distribution.

Originally, the MDN is trained on trajectories collected using uniformly random actions during an exploration phase. Each training sample is represented as a tuple:

$$(s_0, h, s_h),$$

where $s_0$ is an initial state sampled from a trajectory, $h$ is the horizon, and $s_h$ is a sampled state reached during $h$ timesteps in the future starting from $s_0$.

The MDN models a distribution of future states conditioned on the initial state and horizon, which we denote as $p(s_h \mid s_0, h)$. The entropy of this distribution is used to define the intrinsic reward. Specifically, the Door(s) framework computes the intrinsic reward as:

$$\text{Door}(s) = \frac{1}{H} \sum_{h=1}^{H} \mathcal{H}^{(h)}(S \mid s),$$

where $\mathcal{H}[\cdot]$ denotes the entropy of the predicted future-state distribution (over a state space $S$) $p$ obtained from the MDN and $H$ is the maximum prediction horizon. The high values of Door(s) therefore correspond to high mean entropies over horizons $h \in [1, H]$.

The original method uses a three-stage pipeline: (i) an initial dataset is collected through uniformly random exploration, followed by (ii) MDN pretraining on this fixed dataset, and (iii) RL with the pretrained MDN kept fixed and Door(s) used as a reward shaping signal.

### 3.2 Online Junction State Estimation Model Training

We propose a new method that integrates data collection and MDN training directly with the RL phase.

*3.2.1 Bootstrap dataset.* The training begins with a bootstrap phase in which the replay buffer is filled with sampled environment transitions collected during the standard initial RL exploration phase. This provides the initial dataset for training an MDN and a dynamics model.

*3.2.2 Joint learning framework.* After the bootstrap phase, the RL policy, MDN, and dynamics model are trained jointly. We assume an off-policy RL algorithm that learns the agent's policy from transitions stored in the replay buffer. In our setup, the agent is trained using the task reward together with the Door(s) intrinsic reward. Door(s) relies on an MDN for junction state estimation, while the dynamics model is used to generate synthetic rollouts for additional training data.

*3.2.3 Dynamics model.* A neural network is trained to model one-step environment dynamics:

$$s_{t+1} = f_\theta(s_t, a_t)$$

where $s_t$ is the state at time step $t$, $a_t$ is the action at that time step, and $f_\theta$ is a neural network with parameters $\theta$.

The model is trained using mean squared error (MSE) loss on transitions available in the replay buffer.

*3.2.4 MDN update mechanism.* Each update of an MDN consists of multiple gradient steps over a mixture of:

- Sampled environment data collected during the initial exploration phase stored in the replay buffer.
- Synthetic data generated using the learned dynamics model.

The initial sampled transitions consist of exploration data under the initial policy using randomly uniform actions. Synthetic data is generated by performing batched rollouts using the learned one-step dynamics model $f_\theta$. The procedure is as follows: an initial state $s_0$ is first sampled from the replay buffer, followed by randomly sampling $h \in [1, H]$ for each sample in the batch, and finally the dynamics model is applied for $h$ steps:

$$s_{t+1} = f_\theta(s_t, a_t), \quad t = 0, \ldots, h-1$$

Actions $a_t$ are uniformly sampled from an environment action space at each step. The predicted state is then used as the input to the next step, resulting in a synthetic rollout:

$$s_0 \to s_1 \to \cdots \to s_h$$

where $s_h$ is the predicted state after $h$ model steps.

To improve efficiency, rollouts are performed in batches with different horizons $h$, where only samples that have not yet reached their horizon are updated at each step.

The final state $s_h$ is used together with the initial state $s_0$ and horizon $h$ to create tuples of the form $(s_0, h, s_h)$. Notice that these tuples have the same format as the samples used in the original training scheme.

Because the dynamics model is trained using one-step predictions, prediction errors can accumulate over longer rollouts.

*3.2.5 Progressive mixture ramp.* To reduce the impact of model inaccuracies, synthetic samples are introduced gradually during training using a progressive mixture ramp. A progressive sampling schedule is used to control the proportion of synthetic data included for MDN training. Instead of immediately training fully on synthetic data, the probability of sampling synthetic data increases over time.

This procedure has three hyperparameters: a start probability $p_{\text{start}}$, an end probability $p_{\text{end}}$, and a ramp duration $T$. The ramp duration defines the number of environment steps during which the MDN is periodically updated, while the sampling probability of synthetic data increases from $p_{\text{start}}$ to $p_{\text{end}}$. The ramp starts after $U$ collected environment transitions, allowing for MDN and dynamics model training on non-synthetic data.

The probability of sampling a synthetic transition at each update step $u$ is defined as:

$$p_{\text{synthetic}}(u) = (1-\alpha)\, p_{\text{start}} + \alpha\, p_{\text{end}}, \quad \alpha = \min\left(1, \frac{u}{T}\right)$$

This results in a linear increase in the proportion of synthetic samples during training. Early in training, the MDN is trained primarily on environment data collected during the initial RL exploration phase. As training progresses, synthetic rollouts generated by the learned dynamics model are included more often, eventually reaching the defined maximum proportion $p_{\text{end}}$.

By progressively introducing synthetic data, the influence of the dynamics model on MDN training is gradually increasing during training.

*3.2.6 Reward shaping activation.* Reward shaping with the Door(s) reward is initially disabled and is only activated after the first MDN update. This ensures that the exploration bonuses obtained from the proposed method do not bias exploration towards arbitrary states and negatively influence the data from which RL policy and the dynamics model are trained.

## 4 Experiments

In this section, we describe the experimental setup used for evaluation of the proposed method in the `Pendulum-v1` environment [10]. We compare the proposed online junction estimation method with the original offline approach. We then present state-space heatmaps and the dynamics model prediction error to analyze the resulting learning behavior.

*4.0.1 Environment.* The experiments are conducted on the `Pendulum-v1` task [10], a standard continuous-control environment for RL. The state space in the environment consists of the pendulum angle and angular velocity, while the action is a continuous torque applied to the pendulum. Each episode has a fixed length of 200 steps [10], and we set the horizon $H$ for junction state estimation to match this value. The original dense reward function defining the `Pendulum-v1` task is replaced with sparse binary feedback indicating successful balancing at an upright angle with sufficiently small angular velocity. This reward is maximized together with the Door(s) intrinsic reward described above.

*4.0.2 Implementation details.* In all experiments we use the Truncated Quantile Critics algorithm [5], implemented through the Stable-Baselines3 RL library [9], together with custom environment wrappers required by our approach.

An MDN uses a mixture of Gaussians conditioned on the starting state $s_0$ and horizon $h$. Both an MDN and a dynamics model use feedforward neural networks with hidden layers of size 512 and are trained using an Adam optimizer [4].

The progressive mixture ramp is configured with a synthetic data probability of $p_{\text{start}} = 0.0$ at the beginning and $p_{\text{end}} = 1.0$ at the end of the ramp. The ramp duration is $T = 10$, corresponding to 2000 environment steps, as the MDN is updated once every 200 environment steps. Before the ramp, the dynamics model and MDN are trained on the first $U$ transitions collected during the initial exploration phase, effectively kick-starting the online learning process. The progressive ramp then gradually increases the proportion of synthetic samples during subsequent RL steps.

An MDN, a dynamics model, and a TQC policy are updated at the end of every episode.

*4.0.3 Summary of experimental conditions.* We evaluate two experimental configurations:

(1) **Original training scheme:** an MDN is pretrained on an initial dataset collected in the pretraining phase.
(2) **Online training scheme (proposed):** joint training of an MDN, a dynamics model, and an RL agent during data collection.

### 4.1 Results

Figure 1 shows example state-space heatmaps obtained under different initial data configurations for the two methods. Figures 1a and 1b present two example runs of the original junction state estimation method. Figure 1a shows a successful run, while Figure 1b shows an unsuccessful run obtained with a smaller amount of data sampled under uniformly random action distribution.

Figures 1c and 1d show results from two runs of the proposed method. Figure 1c shows a successful run after performing 2000 environment steps. Out of these, a dynamics model and an MDN are first trained on 1000 collected transitions obtained in the initial exploration phase, after which a progressive ramp gradually introduces synthetic samples. Comparing Figure 1b and 1c shows that the reward landscapes after the same amount of environment steps result in better accuracy with the latter. Figure 1d shows an unsuccessful run after performing 2000 steps during the RL phase, during which synthetic data is generated using

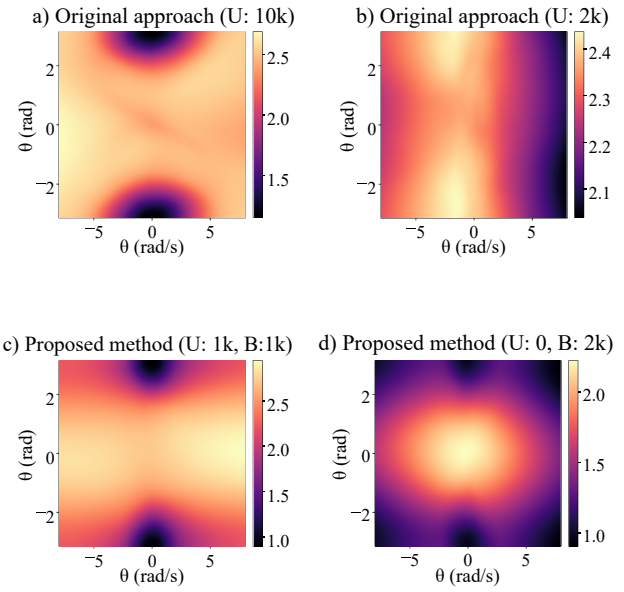



**Figure 1: The heatmaps indicate the amount of intrinsic reward with respect to the pendulum angle and angular velocity. Different data regimes result in different reward landscapes. The parameter $U$ denotes the number of environment steps using transitions collected during pretraining (a-b) or the initial exploration phase (c-d), while $B$ denotes the number of executed environment steps during RL. During this phase, a progressive mixture ramp is introduced, and synthetic samples generated by the learned dynamics model are gradually added to the MDN training dataset.**

the learned dynamics model, but without any environment steps using uniformly sampled environment data. This illustrates the importance of initial environment data for training the dynamics model and MDN.

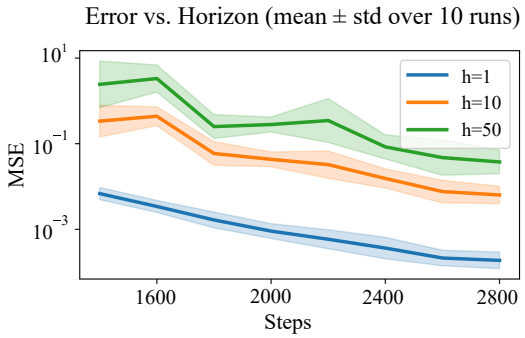



**Figure 2: Dynamics model prediction error (MSE) across training for different rollout horizons $h \in \{1, 10, 50\}$. The prediction error grows smaller with the number of timesteps (reflecting the amount of MDN training), and grows larger with the horizon size $h$.**

Figure 2 shows the MSE of the dynamics model during training after the bootstrap phase for rollout horizons $H = 1$, $H = 10$, and $H = 50$. The curves show the mean over 10 independent runs, with shaded regions indicating the standard deviation. The error decreases as more transitions become available for training, but increases with the rollout horizon $h$.

## 5 Discussion and conclusion

In this work, we proposed an online training variant of the junction state estimation method introduced in [3]. The proposed method replaces the fixed offline dataset by continuously updating the MDN during RL and jointly training a dynamics model to generate synthetic trajectories.

The experimental results on the `Pendulum-v1` environment show that the proposed method can successfully train the MDN using fewer environment steps collected with uniformly sampled actions than the original approach. In particular, the proposed online training scheme achieves a successful reward landscape after 2000 environment steps, whereas this amount of data is not sufficient for the original approach.

However, the results also show that the proposed method remains sensitive to the amount of initial RL exploration data. When very little initial data is available, the dynamics model produces less accurate rollouts, which can negatively affect MDN training and lead to unsuccessful experimental runs. Although synthetic data can increase the coverage of the training data, it also introduces model error that can accumulate over longer rollouts. This is also reflected in Figure 1d, where directly mixing uniformly sampled environment data and synthetic data from the beginning resulted in worse performance and less stable learning. The progressive mixture ramp instead gradually introduces synthetic samples as training progresses, reducing their influence during the early stages of learning. These results suggest that gradually introducing synthetic data is important for maintaining training stability.

Longer rollout horizons exhibit larger prediction errors than short-horizon predictions, indicating that errors in the one-step dynamics model accumulate as the rollout length increases. The progressive mixture ramp reduces the influence of low-quality synthetic data during the early stages of training, but it does not eliminate the effect of inaccurate long-horizon rollouts.

Overall, the results show that the MDN model used for junction state estimation can be trained online during RL while requiring less uniformly sampled environment data than a separate training approach involving multiple phases. This demonstrates that junction state estimation does not require a fixed pre-collected dataset and can instead be adapted as new data becomes available during learning. Future work could investigate methods for improving the quality of synthetic rollouts. Evaluating the approach on more complex environments would also provide a stronger assessment of robustness of our approach, and studying learning of a given task under the proposed changed training regime would give more insight into the learning dynamics. Finally, using Door(s) as a guiding signal could be researched in experiments studying exploration and expedited learning involving humans.

## References

[1] Arthur Aubret, Laetitia Matignon, and Salima Hassas. 2023. An information-theoretic perspective on intrinsic motivation in reinforcement learning: a survey. *Entropy*, 25, 2. doi:10.3390/e25020327.

[2] Christopher M. Bishop. Mixture density networks. Birmingham, (1994). https://publications.aston.ac.uk/id/eprint/373/.

[3] Benjamin Fele and Jan Babic. 2025. $Door(s)$: junction state estimation for efficient exploration in reinforcement learning. In *9th Annual Conference on Robot Learning*. https://openreview.net/forum?id=NtnPVwUCAH.

[4] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. (Jan. 2017). Retrieved Jan. 3, 2023 from arXiv: 1412.6980 [cs].

[5] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. 2020. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. (2020). https://arxiv.org/abs/2005.04269 arXiv: 2005.04269 [cs.LG].

[6] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning* (ICML '99). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 278–287. ISBN: 1558606122.

[7] Pierre-Yves Oudeyer and Frederic Kaplan. 2007. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, Volume 1 - 2007. doi:10.3389/neuro.12.006.2007.

[8] Francesco Poli, Jill X. O'Reilly, Rogier B. Mars, and Sabine Hunnius. 2024. Curiosity and the dynamics of optimal exploration. *Trends in Cognitive Sciences*, 28, 5, 441–453. doi:https://doi.org/10.1016/j.tics.2024.02.001.

[9] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22, 268, 1–8. http://jmlr.org/papers/v22/20-1364.html.

[10] Mark Towers et al. 2025. Gymnasium a standard interface for reinforcement learning environments. (2025). httpsarxiv.orgabs2407.17032 arXiv: 2407.17032 [cs.LG].

[11] Daniel M. Wolpert, Jörn Diedrichsen, and J. Randall Flanagan. 2011. Principles of sensorimotor learning. *Nature Reviews Neuroscience*, 12, 12, (Dec. 2011), 739–751. doi:10.1038/nrn3112.

[12] Gabriele Wulf and Rebecca Lewthwaite. 2016. Optimizing performance through intrinsic motivation and attention for learning: The OPTIMAL theory of motor learning. *Psychonomic Bulletin & Review*, 23, 5, (Oct. 2016), 1382–1414. doi:10.3758/s13423-015-0999-9.