Beyond Surprise: Improving Exploration Through Surprise Novelty

Hung Le
Applied AI Institute, Deakin University
Geelong, Australia
thai.le@deakin.edu.au

Dung Nguyen
Applied AI Institute, Deakin University
Geelong, Australia
dung.nguyen@deakin.edu.au

Kien Do
Applied AI Institute, Deakin University
Geelong, Australia
k.do@deakin.edu.au

Svetha Venkatesh
Applied AI Institute, Deakin University
Geelong, Australia
svetha.venkatesh@deakin.edu.au

ABSTRACT
We present a new computing model for intrinsic rewards in reinforcement learning that addresses the limitations of existing surprise-driven explorations. The reward is the novelty of the surprise rather than the surprise norm. We estimate the surprise novelty as retrieval errors of a memory network wherein the memory stores and reconstructs surprises. Our surprise memory (SM) augments the capability of surprise-based intrinsic motivators, maintaining the agent’s interest in exciting exploration while reducing unwanted attraction to unpredictable or noisy observations. Our experiments demonstrate that the SM combined with various surprise predictors exhibits efficient exploring behaviors and significantly boosts the final performance in sparse reward environments, including Noisy-TV, navigation and challenging Atari games.

KEYWORDS
Reinforcement Learning; Exploration; Intrinsic Motivation; Memory

ACM Reference Format:

1 INTRODUCTION
What motivates agents to explore? Successfully answering this question would enable agents to learn efficiently in formidable tasks. Random explorations such as $\epsilon$-greedy are inefficient in high dimensional cases, failing to learn despite training for hundreds of million steps in sparse reward games [5]. Alternative approaches propose to use intrinsic motivation to aid exploration by adding bonuses to the environment’s rewards [5, 33]. The intrinsic reward is often proportional to the novelty of the visiting state: it is high if the state is novel (e.g. different from the past ones [2, 3]) or less frequently visited [5, 35].

Another view of intrinsic motivation is from surprise, which refers to the result of the experience being unexpected, and is determined by the discrepancy between the expectation (from the agent’s prediction) and observed reality [4, 31]. Technically, surprise is the difference between prediction and observation representation vectors. The norm of the residual (i.e. prediction error) is used as the intrinsic reward. Here, we will use the terms “surprise” and “surprise norm” to refer to the residual vector and its norm, respectively. Recent works have estimated surprise with various predictive models such as dynamics [33], episodic reachability [29] and inverse dynamics [25]; and achieved significant improvements with surprise norm [7]. However, surprise-based agents tend to be overly curious about noisy or unpredictable observations [16, 30]. For example, consider an agent watching a television screen showing white noise (noisy-TV problem). The TV is boring, yet the agent cannot predict the screen’s content and will be attracted to the TV due to its high surprise norm. This distraction or “fake surprise” is common in partially observable Markov Decision Process (POMDP), including navigation tasks and Atari games [8]. Many works have addressed this issue by relying on the learning progress [1, 30] or random network distillation (RND) [8]. However, the former is computationally expensive, and the latter requires many samples to perform well.

This paper overcomes the “fake surprise” issue by using surprise novelty - a new concept that measures the uniqueness of surprise. To identify surprise novelty, the agent needs to compare the current surprise with surprises in past encounters. One way to do this is to equip the agent with some kind of associative memory, which we implement as an autoencoder whose task is to reconstruct a query surprise. The lower the reconstruction error, the lower the surprise novelty. A further mechanism is needed to deal with the rapid changes in surprise structure within an episode. As an example, if the agent meets the same surprise at two time steps, its surprise novelty should decline, and with a simple autoencoder this will not happen. To remedy this, we add an episodic memory, which stores intra-episode surprises. Given the current surprise, this memory can retrieve similar “surprises” presented earlier in the episode through an attention mechanism. These surprises act as a context added to the query to help the autoencoder better recognize whether the query surprise has been encountered in the episode or not. The
error between the query and the autoencoder’s output is defined as surprise novelty, to which the intrinsic reward is set proportionally.

We argue that using surprise novelty as an intrinsic reward is better than surprise norm. As in POMDPs, surprise norms can be very large since the agent cannot predict its environment perfectly, yet there may exist patterns of prediction failure. If the agent can remember these patterns, it will not feel surprised when similar prediction errors appear regardless of the surprise norms. An important emergent property of this architecture is that when random observations are presented (e.g., white noise in the noisy-TV problem), the autoencoder can act as an identity transformation operator, thus effectively “passing the noise through” to reconstruct it with low error. We conjecture that the autoencoder is able to do this with the surprise rather than the observation as the surprise space has lower variance, and we show this in our paper. Since our memory system works on the surprise level, we need to adopt current intrinsic motivation methods to generate surprises. The surprise generator (SG) can be of any kind based on predictive models mentioned earlier and is jointly trained with the memory to optimize its own loss function. To train the surprise memory (SM), we optimize the memory’s parameters to minimize the reconstruction error.

Our contribution is two-fold:

- We propose a new concept of surprise novelty for intrinsic motivation. We argue that it reflects better the environment originality than surprise norm (see motivating graphics Fig. 1).
- We design a novel memory system, named Surprise Memory (SM) that consists of an autoencoder associative memory and an attention-based episodic memory. Our two-memory system estimates surprise novelty within and across episodes.

In our experiments, the SM helps RND [8] perform well in our challenging noisy-TV problem while RND alone performs poorly. Not only with RND, we consistently demonstrate significant performance gain when coupling three different SGs with our SM in sparse-reward tasks. Finally, in hard exploration Atari games, we boost the scores of 2 strong SGs, resulting in better performance under the low-sample regime.

2 METHODS

2.1 Surprise Novelty

Surprise is the difference between expectation and observation [11]. If a surprise repeats, it is no longer a surprise. Based on this intuition, we hypothesize that surprises can be characterized by their novelties, and an agent’s curiosity is driven by the surprise novelty rather than the surprising magnitude. Moreover, surprise novelty should be robust against noises: it is small even for random observations. For example, watching a random-channel TV can always be full of surprises as we cannot expect which channel will appear next. However, the agent should soon find it boring since the surprise of random noises reoccurs repeatedly, and the channels are entirely unpredictable.

We propose using a memory-augmented neural network (MANN) to measure surprise novelty. The memory remembers past surprise patterns, and if a surprise can be retrieved from the memory, it is not novel, and the intrinsic motivation should be small. The memory can also be viewed as a reconstruction network. The network can pass its inputs through for random, pattern-free surprises, making them retrievable. Surprise novelty has an interesting property: if some event is unsurprising (the expectation-reality residual is $\rightarrow 0$), its surprise ($\rightarrow 0$ with norm $0$) is always perfectly retrievable (surprise novelty is $0$). In other words, low surprise norm means low surprise novelty. On the contrary, high surprise norm can have little surprise novelty as long as the surprise can be retrieved from the memory either through associative recall or pass-through mechanism. Another property is that the variance of surprise is generally lower than that of observation (state), potentially making the learning on surprise space easier. This property is formally stated as follows.

Figure 1: Montezuma Revenge: surprise novelty better reflects the originality of the environment than surprise norm. While surprise norm can be significant even for dull events such as those in the dark room due to unpredictability, surprise novelty tends to be less ($3^{rd}$ and $6^{th}$ image). On the other hand, surprise novelty can be higher in truly vivid states on the first visit to the ladder and island rooms ($1^{st}$ and $2^{nd}$ image) and reduced on the second visit ($4^{th}$ and $5^{th}$ image). Here, surprise novelty and surprise norm are quantified and averaged over steps in each room.
The surprise generated by the SG is stored and processed by a memory network dubbed Surprise Memory (SM). It consists of an episodic memory $\mathcal{M}$ and an autoencoder network $\mathcal{W}$. \(\mathcal{M}\) is slot-based, storing past surprises within the episode. At timestep $t$, given surprise $u_t$, \(\mathcal{M}\) retrieves read-out $u_t'$ to form a query surprise $q_t = [u_t', u_t]$ to $\mathcal{W}$. $\mathcal{W}$ reconstructs the query and takes the reconstruction error (surprise novelty) as the intrinsic reward $r_t$.

**Figure 2:** Surprise Generator+Surprise Memory (SG+SM). The SG takes input $I_t$ from the environment to estimate the surprise $u_t$ at state $s_t$. The SM consists of two modules: an episodic memory $\mathcal{M}$ and an autoencoder network $\mathcal{W}$. $\mathcal{M}$ is slot-based, storing past surprises within the episode. At timestep $t$, given surprise $u_t$, $\mathcal{M}$ retrieves read-out $u_t'$ to form a query surprise $q_t = [u_t', u_t]$ to $\mathcal{W}$. $\mathcal{W}$ reconstructs the query and takes the reconstruction error (surprise novelty) as the intrinsic reward $r_t$.

**Proposition 1.** Let $X$ and $U$ be random variables representing the observation and surprise at the same timestep, respectively. Under an imperfect SG, the following inequality holds:

$$\forall i: \left(\sigma _{i}^{X}\right)^{2} \geq \left(\sigma _{i}^{U}\right)^{2}$$

where $\left(\sigma _{i}^{X}\right)^{2}$ and $\left(\sigma _{i}^{U}\right)^{2}$ denote the $i$-th diagonal elements of $\text{var}(X)$ and $\text{var}(U)$, respectively.

**Proof.** See Appendix E. \(\square\)

### 2.2 Surprise Generator

Since our MANN requires surprises for its operation, it is built upon a prediction model, which will be referred to as Surprise Generators (SG). In this paper, we adopt many well-known SGs (e.g. RND [8] and ICM [25]) to predict the observation, compute the surprise $u_t$ for every step in the environment. The surprise norm is the Euclidean distance between the expectation and the reality:

$$\|u_t\| = \|\text{SG}(I_t) - O_t\|$$  \(1\)

where $u_t \in \mathbb{R}^n$ is the surprise vector of size $n$, $I_t$ the input of the SG at step $t$ of the episode, $\text{SG}(I_t)$ and $O_t$ the SG's prediction and the observation target, respectively. The input $I_t$ is specific to the SG architecture choice, which can be the current ($s_t$) or previous state, action ($s_{t-1}$, $a_t$). The observation target $O_t$ is usually a transformation (can be identical or random) of the current state $s_t$, which serves as the target for the SG’s prediction. The SG is usually trained to minimize:

$$\mathcal{L}_{\text{SG}} = \mathbb{E}_t \left[\|u_t\|\right]$$  \(2\)

Here, predictable observations have minor prediction errors or little surprise. One issue is that a great surprise norm can be simply due to noisy or distractive observations. Next, we propose a remedy for this problem.

### 2.3 Surprise Memory

The surprise generated by the SG is stored and processed by a memory network dubbed Surprise Memory (SM). It consists of
The read-out and the SG’s surprise form the query surprise to \( W \): \( q_t = [u_f', u_I] \in \mathbb{R}^{2M} \). \( M \) stores intra-episode surprises to assist the autoencoder in preventing the agent from exploring “fake surprise” within the episode. Our episodic memory formulation is unlike prior KNN-based episodic memory \([3]\) because our system learns to enforce the memory retrieval error to be small. This is critical when the representations stored in \( M \) can not be easily discriminated if we only rely on unsupervised distance-based metrics. More importantly, under our formulation, the retrieval error can still be small even when the stored items in the memory differ from the query. This helps detect the “fake surprise” when the query surprise seems unlike those in the memory if considered individually, yet can be approximated as weighted sum of the store surprises.

In general, since we optimize the parameters to reconstruct \( u_I \) using past surprises in the episode, if the agent visits a state whose surprise is predictable from those in \( M \), \( \|u_f' - u_I\| \) should be small. Hence, the read-out context \( u_f' \) contains no extra information than \( u_I \) and reconstructing \( q_I \) from \( W \) becomes easier as it is equivalent to reconstructing \( u_I \). In contrast, visiting diverse states leads to a more novel read-out \( u_f' \) and makes it more challenging to reconstruct \( q_I \), generally leading to higher intrinsic reward.

The autoencoder network \( W \) can be viewed as an associative memory of surprises that persist across episodes. At timestep \( t \) in any episode during training, \( W \) is queried with \( q_I \) to produce a reconstructed memory \( \tilde{q}_I \). The surprise novelty is determined as:

\[
r_I^t = \|\tilde{q}_I - q_I\| \tag{4}
\]

which is the norm of the surprise residual \( \tilde{q}_I - q_I \). It will be normalized and added to the external reward as an intrinsic reward bonus. The details of computing and using normalized intrinsic rewards can be found in Appendix C.

We implement \( W \) as a feed-forward neural network that learns to reconstruct its own inputs. The query surprise is encoded to the weights of the network via backpropagation as we minimize the reconstruction loss below:

\[
L_W = \mathbb{E}_t[r_I^t] = \mathbb{E}_t[\|W(q_I) - q_I\|] \tag{5}
\]

Here, \( \tilde{q}_I = W(q_I) \). Intuitively, it is easier to retrieve non-novel surprises experienced many times in past episodes. Thus, the intrinsic reward is lower for states that lead to these familiar surprises. On the contrary, rare surprises are harder to retrieve, which results in high reconstruction errors and intrinsic rewards. We note that autoencoder (AE) has been shown to be equivalent to an associative memory that supports memory encoding and retrieval through attractor dynamics \([27]\). Unlike slot-based memories, AE has a fixed memory capacity, compresses information and learns data representations. We could store the surprise in a slot-based memory across episodes, but the size of this memory would be autonomous, and the data would be stored redundantly. Hence, the quality of the stored surprise will reduce as more and more observations come in. On the other hand, AE can efficiently compress surprises to latent representations and hold them to its neural weights, and the surprise retrieval is optimized. Besides, AE can learn to switch between 2 mechanisms: pass-through and pattern retrieval, to optimally achieve its objective. We cannot do that with slot-based memory. Readers can refer to Appendix A to see the architecture details and how \( W \) can be interpreted as implementing associative memory.

The whole system SG+SM is trained end-to-end by minimizing the following loss: \( L = L_{SG} + L_M + L_W \). Here, we block the gradients from \( L_W \) backpropagated to the parameters of SG to avoid trivial reconstructions of \( q_I \). The pseudocode of our algorithm is presented in Algo. 1 and 2. We note that vector notations in the algorithm are row vectors. For simplicity, the algorithm assumes 1 actor. In practice, our algorithm works with multiple actors and mini-batch training. See Appendix C for explanation of \( \beta \) and \( r_{std} \).

### 3 EXPERIMENTAL RESULTS

#### 3.1 Noisy-TV: Robustness against Noisy Observations

We use Noisy-TV, an environment designed to fool exploration methods \([8, 29]\), to confirm that our method can generate intrinsic...
white noise image sampled from standard normal distribution and the baseline. Since our model and the baseline, the first two states are noises, the following three states are mainly by the noisy TV, resulting in the highest MNIRs. However, the Noisy-TV is designed to fool intrinsic motivation methods and thus, the number of TV channels can be considered infinity. The agent’s state is an image of its viewport, and its goal is to search for a red box randomly placed in the maze (+1 reward if the agent reaches the goal).

The main baseline is RND [8], a simple yet strong SG that is claimed to obviate the stochastic problems of Noisy-TV. Our SG+SM model uses RND as the SG, so we name it RND+SM. The source code can be accessed on our GitHub repository at https://github.com/thaihungle/SM. Since our model and the baseline share the same RND architecture, the difference in performance must be attributed to our SM. We also include a disagreement-based method (Disagreement) [26], which leverages the variance of multiple dynamic predictors as the intrinsic reward. Finally, as a reference, we use PPO without intrinsic reward as the vanilla Baseline.

The result demonstrates that the proposed SM outperforms other intrinsic motivators by a significant margin, as shown in Fig. 3 (b). Notably, SM improves RND performance by around 25%, showcasing the impact of surprise novelty. The result also shows that RND+SM outperforms the vanilla Baseline. Although the improvement is moderate (0.9 vs 0.85), the result is remarkable since the Noisy-TV is designed to fool intrinsic motivation methods and among all, only RND+SM can outperform the vanilla Baseline.

Fig. 3 (a) illustrates the mean-normalized intrinsic rewards (MNIR) measured at different states in our Noisy-TV environment. The first two states are noises, the following three states are common walls, and the last two are ones where the agent sees the box. The MNIR bars show that both models are attracted mainly by the noisy TV, resulting in the highest MNIRs. However, our model with SM suffers less from noisy TV distractions since its MNIR is lower than RND’s. We speculate that SM is able to partially reconstruct the white-noise surprise via the pass-through mechanism, making the normalized surprise novelty generally smaller than the normalized surprise norm in this case. That mechanism is enhanced in SM with surprise reconstruction (see Appendix D.1 for explanation).

On the other hand, when observing red box, RND+SM shows higher MNIR than RND. The difference between MNIR for common and rare states is also more prominent in RND+SM than in RND because RND prediction is not perfect even for common observations, creating relatively significant surprise norms for seeing walls. The SM fixes that issue by remembering surprise patterns and successfully retrieving them, producing much smaller surprise novelty compared to those of rare events like seeing red box. Consequently, the agent with SM outperforms the other by a massive margin in task rewards (Fig. 3 (b)).

As we visualize the number of watching TV actions and the value of the intrinsic reward by RND+SM and RND over training time, we realize that RND+SM helps the agent take fewer watching actions and thus, collect smaller amounts of intrinsic rewards compared to RND. See more in Appendix D.1.

### 3.2 MiniGrid: Compatibility with Different Surprise Generators

We show the versatility of our framework SG+SM by applying SM to 4 SG backbones: RND [8], ICM [25], NGU [3] and autoencoder-AE (see Appendix D.2 for implementation details). We test the models on three tasks from MiniGrid environments: Key-Door (KD), Dynamic-Obstacles (DO) and Lava-Crossing (LC) [10]. If the agent reaches the goal in the tasks, it receives a +1 reward. Otherwise, it can be punished with negative rewards if it collides with obstacles or takes too much time to finish the task. These environments are not as stochastic as the Noisy-TV but they still contain other types of distraction. For example, in KD, the agent can be attracted to irrelevant actions such as going around to drop and pick the key. In DO, instead of going to the destination, the agent may chase obstacle balls flying around the map. In LC the agent can commit unsafe actions like going near lava areas, which are different from typical paths. In any case, due to reward sparsity, intrinsic motivation is beneficial. However, surprise alone may not be enough to guide an

---

1See Appendix C for more information on this metric.
AE performance is just slightly above the no-IR baseline. The

### Table 1: MiniGrid: test performance after 10 million training steps. The numbers are average task return × (mean (surprise norm), even for special events (pick the key). Therefore, realize that if the SG is strong, SG+SM tends to have better most of the time, leading to similar AE's reconstruction errors, which can vary even when they have similar norms (Appendix Fig. 8). Therefore, AE+SM can assign high IR to special events while still providing low IR to noisy states, which is optimal in this case.

To analyze the difference between the SG+SM and SG’s MNIR structure, we visualize the MNIR for each cell in the map of Key-Door in Figs. 4 (b) and (c). We create a synthetic trajectory that scans through all the cells in the big room on the left and, at each cell, uses RND+SM and RND models to compute the corresponding surprise-norm and surprise-novelty MNIRs, respectively. As shown in Fig. 4 (b), RND+SM selectively identifies truly surprising events, where only a few cells have high surprise-novelty MNIR. Here, we can visually detect three important events that receive the most MNIR: seeing the key (bottom row), seeing the door side (in the middle of the rightmost column) and approaching the front of the door (the second and fourth rows). Other less important cells are assigned very low MNIR. On the contrary, RND often gives high IR to special events while still providing low IR to noisy states, which is optimal in this case.

Regarding the compatibility of our SM with different SGs, we realize that if the SG is strong, SG+SM tends to have better performance. For example, in LC task, SG is the best SG, resulting ICM+SM being the best performer. The exception is the AE SG in KD task. We speculate that AE generates intrinsic rewards as the reconstruction error, which can be more sensitive to the state representation of specific tasks. In KD, the viewpoint of the agent is almost empty. The state representations look similar most of the time, leading to similar AE’s reconstruction errors (surprise norm), even for special events (pick the key). Therefore, AE performance is just slightly above the no-IR baseline. The good thing is that other noisy states like throwing the key also receive low IR. When equipped with SM, AE+SM differentiates the reconstruction residual vectors, which can vary even when they have similar norms (Appendix Fig. 8). Therefore, AE+SM can assign high IR to special events while still providing low IR to noisy states, which is optimal in this case.

#### 3.3 Atari: Sample-efficient Benchmark

We adopt the sample-efficiency Atari benchmark [17] on six hard exploration games where the training budget is only 50
Table 2: Atari: average return over 128 episodes after 50 million training frames (mean over 5 runs). ♠️ is from a prior work [12]. ♦️ is our run. The last two rows are mean and median human normalized scores. Bold denotes best results. Italic denotes that SG+SM is significantly better than SG regarding Cohen effect size less than 0.5.

<table>
<thead>
<tr>
<th>Task</th>
<th>EMI*</th>
<th>LWM*</th>
<th>RND*</th>
<th>NGU°</th>
<th>LWM°</th>
<th>LWM+SM°</th>
<th>RND°</th>
<th>RND+SM°</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freeway</td>
<td>33.8</td>
<td>30.8</td>
<td>33.3</td>
<td>2.6</td>
<td>31.1</td>
<td>31.6</td>
<td>22.2</td>
<td>22.2</td>
</tr>
<tr>
<td>Frostbite</td>
<td>7002</td>
<td>8409</td>
<td>2227</td>
<td>1751</td>
<td>8598</td>
<td>10258</td>
<td>2628</td>
<td>5073</td>
</tr>
<tr>
<td>Venture</td>
<td>646</td>
<td>998</td>
<td>707</td>
<td>790</td>
<td>985</td>
<td>1381</td>
<td>1081</td>
<td>1119</td>
</tr>
<tr>
<td>Gravitar</td>
<td>558</td>
<td>1376</td>
<td>546</td>
<td>839</td>
<td>1,242</td>
<td>1693</td>
<td>739</td>
<td>987</td>
</tr>
<tr>
<td>Solaris</td>
<td>2688</td>
<td>1268</td>
<td>2051</td>
<td>1103</td>
<td>1839</td>
<td>2065</td>
<td>2206</td>
<td>2420</td>
</tr>
<tr>
<td>Montezuma</td>
<td>387</td>
<td>2276</td>
<td>377</td>
<td>2062</td>
<td>2192</td>
<td>2269</td>
<td>2475</td>
<td>5187</td>
</tr>
<tr>
<td>Norm. Mean</td>
<td>61.4</td>
<td>80.6</td>
<td>42.2</td>
<td>31.2</td>
<td>80.5</td>
<td>97.0</td>
<td>50.7</td>
<td>74.8</td>
</tr>
<tr>
<td>Norm. Median</td>
<td>34.9</td>
<td>60.8</td>
<td>32.7</td>
<td>33.1</td>
<td>66.5</td>
<td>83.7</td>
<td>58.3</td>
<td>84.6</td>
</tr>
</tbody>
</table>

3.4 Ablation Study

Role of Memories Here, we use Minigrid’s Dynamic-Obstacle task to study the role of 𝑀 and 𝐾 in the SM (built upon RND as the SG). Disabling ‘𝐾’ we directly use ∥𝑞𝑡∥ = ∥∥𝑢𝑡, 𝑢𝑡+1∥∥ as the intrinsic reward, and name this version: SM (no ‘𝐾’). To ablate the effect of 𝑀, we remove 𝑢𝑡 from 𝑞𝑡 and only use 𝑞𝑡 = 𝑢𝑡 as the query to ‘𝐾’, forming the version: SM (no 𝑀). We also consider different episodic memory capacities and slot sizes 𝑁−𝑑∈

{32 − 4, 128 − 16, 1024 − 64}. As 𝑁 and 𝑑 increase, the short-term context expands and more past surprise information is considered in the attention. In theory, a big 𝑀 is helpful to capture long-term and more accurate context for constructing the surprise query.

Fig. 5 (c) depicts the performance curves of the methods after 10 million training steps. SM (no ‘𝐾’) and SM (no 𝑀) show weak signs of learning, confirming the necessity of both modules in this task. Increasing 𝑁−𝑑 from 32 − 4 to 1024 − 64 improves the final performance. However, 1024 − 64 is not significantly better than 128 − 16, perhaps because it is unlikely to have similar surprises that are more than 128 steps apart. Thus, a larger attention span does not provide a benefit. As a result, we keep using 𝑁 = 128 and 𝑑 = 16 in all other experiments for faster computing. We also verify the necessity of 𝑀 and ‘𝐾’ in Montezuma Revenge, Frostbite, Venture and illustrate how 𝑀 generates lower MNIR when 2 similar event occurs in the same episode in Key-Door (see Appendix D.4).

No Task Reward In this experiment, we remove task rewards and merely evaluate the agent’s ability to explore using intrinsic rewards. The task is to navigate 3D rooms and get a +1 reward for picking an object [9]. The state is the agent’s image view, and there is no noise. Without task rewards, it is crucial to maintain the agent’s interest in the unique events of seeing the objects. In this partially observable environment, surprise-prediction methods may struggle to explore even without noise due to a lack of information for good predictions, leading to usually high prediction errors. For this testbed, we evaluate random exploration agent (Baseline), RND and RND+SM in 2 settings: 1 room with three objects (easy), and 4 rooms with one object (hard).

To see the difference among the models, we compare the cumulative task rewards over 100 million steps (see Appendix D.4 for details). RND is even worse than Baseline in the easy setting because predicting causes high biases (intrinsic rewards) towards the unpredictable, hindering exploration if the map is simple. In
contrast, RND+SM uses surprise novelty, generally showing smaller intrinsic rewards (see Appendix Fig. 12 (right)). Consequently, our method consistently demonstrates significant improvements over other baselines (see Fig. 5 (d) for the hard setting).

4 RELATED WORKS

Intrinsic motivation approaches usually give the agent reward bonuses for visiting novel states to encourage exploration. The bonus is proportional to the mismatch between the predicted and reality, also known as surprise [31]. One kind of predictive model is the dynamics model, wherein the surprise is the error of the models as predicting the next state given the current state and action [1, 33]. One critical problem of these approaches is the unwanted bias towards transitions where the prediction target is a stochastic function of the inputs, commonly found in partially observable environments. Recent works focus on improving the features of the predictor’s input by adopting representation learning mechanisms such as inverse dynamics [25], variational autoencoder, random/pixel features [7], or whitening transform [12]. Although better representations may improve the reward bonus, they cannot completely solve the problem of stochastic dynamics and thus, fail in extreme cases such as the noisy-TV problem [8].

Besides dynamics prediction, several works propose to predict other quantities as functions of the current state by using autoencoder [24], episodic memory [29], and random network [8]. Burda et al. (2018) claimed that using a deterministic random target network is beneficial in overcoming stochasticity issues. Other methods combine this idea with episodic memory and other techniques, achieving good results in large-scale experiments [2, 3]. From an information theory perspective, the notation of surprise can be linked to information gain or uncertainty, and predictive models can be treated as parameterized distributions [1, 15, 34]. Furthermore, to prevent the agent from unpredictable observations, the reward bonus can be measured by the progress of the model’s prediction [1, 23, 30] or disagreement through multiple dynamic models [26, 32]. However, these methods are complicated and hard to scale, requiring heavy computing. A different angle to handle stochastic observations during exploration is surprise minimization [6, 28]. In this direction, the agents get bigger rewards for seeing more familiar states. Such a strategy is somewhat opposite to our approach and suitable for unstable environments where the randomness occurs separately from the agents’ actions.

These earlier works use surprise as an incentive for exploration and differ from our principle that utilizes surprise novelty. Also, our work augments these existing works with a surprise memory module and can be used as a generic plug-in improvement for surprise-based models. We note that our memory formulation differs from the memory-based novelty concept using episodic memory [3], momentum memory [13], or counting [5, 35] because our memory operates on the surprise level, not the state level. Moreover, our utilization of memory to store surprise novelty represents a novel approach compared to other memory-based RL methods that typically use memory for storing trajectories [20], policies [19, 22], and hyperparameters [18]. In our work, exploration is discouraged not only in frequently visited states but also in states whose surprises can be reconstructed using SM. Our work provides a more general and learnable novelty detection mechanism, which is more flexible than counting lookup table.

5 DISCUSSION

This paper presents Surprise Generator-Surprise Memory (SG+SM) framework to compute surprise novelty as an intrinsic motivation for the reinforcement learning agent. Exploring with surprise novelty is beneficial when there are repeated patterns of surprises or random observations. In Noisy-TV, our SG+SM can harness the agent’s tendency to visit noisy states such as watching random TV channels while encouraging it to explore rare events with distinctive surprises. We empirically show that our SM can supplement four surprise-based SGs to achieve more rewards in fewer training steps in three grid-world environments. In 3D navigation without external reward, our method significantly outperforms the baselines. On two strong SGs, our SM also achieve superior results in hard-exploration Atari games within 50 million training frames. Even in the long run, our method maintains a clear performance gap from the baselines, as shown in Montezuma Revenge and Frostbite.

ACKNOWLEDGMENTS

This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006).
REFERENCES


