Surprise-Adaptive Intrinsic Motivation for Unsupervised Reinforcement Learning

Adriana Hugessen **Roger Creus-Castanyer** **Glen Berseth**
Université de Montréal and Mila Quebec AI Institute
{adriana.knatchbull-hugessen,roger.creus-castanyer, glen.berseth}@mila.quebec

Abstract

Both surprise-minimizing and surprise-maximizing (curiosity) objectives for unsupervised reinforcement learning (RL) have been shown to be effective in different environments, depending on the environment's level of natural entropy. However, neither method can perform well across all entropy regimes. In an effort to find a single surprise-based method that will encourage emergent behaviors in any environment, we propose an agent that can adapt its objective depending on the entropy conditions in its environment by framing the choice as a multi-armed bandit problem. We devise a novel intrinsic feedback signal for the bandit, which captures the agent's ability to control the entropy in its environment. We demonstrate that such agents can learn to control entropy and exhibit emergent behaviors in both high- and low-entropy regimes.

1 Introduction

Unsupervised reinforcement learning (URL), or learning without access to an extrinsic reward function, has recently gained significant attention, often as a pretraining method [8] or as a reward bonus in sparse reward domains [16, 14, 4]. A recent focus has been on developing objectives where the agent has no access to extrinsic rewards and instead develops emergent behaviors from an intrinsic motivation alone [11, 3, 9]. In this context, unsupervised RL holds the promise of being able to develop natural-like intelligence, i.e. generally-capable agents that can be deployed to solve diverse tasks across diverse environments. However, thus far, no single intrinsic motivation function has succeeded in capturing the complexity of motivation that gives rise to intelligent systems.

Interestingly, two seemingly opposing methods, surprise-minimization [3] and surprise-maximization (curiosity) [14], have been proposed as intrinsic motivations, with both methods performing well depending on the properties of the environment in which they are deployed. In general, surprise-minimizing methods [3] perform well in environments with naturally high entropy that can be reduced through control, while curiosity-based methods [14] are better suited to environments where explicit exploration is necessary to encounter novel information. However, both methods are known to possess failure modes when exposed to the opposite entropy regime [17, 19].

In this work, we propose an adaptive mechanism to select between maximizing and minimizing surprise in a given environment, based on the agent's ability to exert control over its entropy conditions, which we frame as a multi-armed bandit problem. We experimentally validate our *surprise-adaptive* agent by demonstrating its ability to mirror a surprise-maximizing or -minimizing agent in didactic low- and high-entropy environments, respectively. We demonstrate more diverse emergent behaviors, as measured by the performance on extrinsic reward, than observed from the single-objective agents.

2 Related work

There is a rich body of work in the field of unsupervised RL and intrinsic motivation, upon which our method builds. The most widely explored class of intrinsic objectives is related to improving exploration by encouraging novelty-seeking behaviors, otherwise known as "curiosity" methods [4, 14, 16]. Naive implementations of novelty-seeking agents, however, can be susceptible to random noise [17]. Alternatively, intrinsic objectives which seek to minimize surprise in order to exert control over the environment [6], have shown success in high-dimensional and high-entropy environments [3, 15]. Surprise-minimizing agents, however, can fall victim to the "dark room problem" [19], where the agent cannot learn in areas of the state space without any natural entropy.

Two recent works make efforts towards combining surprise-minimization and maximization objectives to avoid the degenerate cases of prior methods, either using a complex multi-agent paradigm [5] or learned skills [21]. However, neither method uses an adaptive mechanism to control the objective, instead using fixed-length windows to alternate between objectives. In contrast, our proposed method can adapt to entropy conditions online.

Prior works have explored adaptivity in RL and found that it can be beneficial for learning [1, 13]. Similar to our work, Moskovitz et al. [13] uses a multi-armed bandit to control a learning hyper-parameter. However, their method relies on extrinsic rewards for providing feedback to the bandit, while our method relies only on intrinsic signals.

3 Background

Reinforcement learning. RL is a learning paradigm for sequential decision-making problems. In RL, an agent acts in an environment from which it receives observations and rewards. Formally, this process can be modelled as a Markov Decision Process (MDP) consisting of the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the reward function, and γ is the discount factor. The goal of the RL agent is to find a policy π_ϕ that produces actions that maximize the expected sum of discounted future rewards.

$$\pi_\phi(a_t|s_t) = \text{argmax}_\phi \mathbf{E}_{p(\tau|\phi)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1}$$

In our experiments, we use the value-based method DQN [12] to solve Equation 1.

Entropy and surprise. The notion of surprise derives from the optimization of the entropy of the state marginal distribution under the policy $\pi_\phi(a|s)$, which we denote $d^{\pi_\phi}(s_t)$. Given an estimate of this state marginal distribution, $p_{\theta_{t-1}}(s_t)$, we can express an estimate of the sum of the entropies of the state distribution across a trajectory (see Appendix A of [3] for a full derivation):

$$\sum_{t=0}^{T} \mathcal{H}(s_t) = \sum_{t=0}^{T} -\mathbf{E}_{s_t \sim d^{\pi_\phi}(s_t)} \left[\log d^{\pi_\phi}(s_t)\right] \leq \sum_{t=0}^{T} \mathbf{E}_{s_t \sim d^{\pi_\phi}(s_t)} \left[-\log p_{\theta_{t-1}}(s_t)\right] \tag{2}$$

Recalling Equation 1, we can see that minimizing the sum of the state entropy over a trajectory (Equation 2) corresponds to an RL agent with a reward function given by:

$$r_{\text{s-min}}(s_t, a_t) = \log p_{\theta_t}(s_{t+1}) \tag{3}$$

and maximizing this objective corresponds to an RL agent with a reward function given by:

$$r_{\text{s-max}}(s_t, a_t) = -\log p_{\theta_t}(s_{t+1}) \tag{4}$$

Conceptually, this means that the agent is punished (or rewarded) if the observed state s_t is "surprising", that is, if it has high negative log-likelihood under the state marginal distribution estimated so far. Hence, we refer to Equation 3 as surprise-minimization and Equation 4 as surprise-maximization.

4 Surprise-adaptive bandit

We propose a multi-armed bandit approach for selecting between minimizing or maximizing surprise. Precisely, at the start of each episode, we select an arm from the bandit according to the UCB

Algorithm 1 Surprise-adaptive agent

```
 1: Initialize network parameters $\phi$, replay buffer $\beta$, bandit parameters $\mu^{(0)}$, and $\alpha^{(0)} \sim \text{Bern}(0.5)$
 2: Compute $H(p_{\theta_{\text{rand}}})$ by rolling out a random trajectory
 3: for episode $m = 0, 1 \ldots, \text{M}$ do
 4:   $s_o \sim p(s_0)$, reset $\theta_0$, $\bar{s_0} = (s_0, \theta_0, 0, \alpha^{(m)})$      ▷ construct initial augmented state
 5:   Set $r(s_t, a_t) = (-1)^{\alpha^{(m)}} - \log p_{\theta_t}(s_t)$      ▷ set reward function
 6:   for $t = 0, \ldots, T$ do
 7:     Collect experience and update policy $\phi \leftarrow RL(\phi, \beta)$      ▷ See Berseth et al. [3]
 8:   end for
 9:   $\mu_i^{(m+1)} \leftarrow \mu_i^{(m)} + \frac{1}{N(i)}(f_m - \mu_i^{(m)})$ if $\alpha^{(m)} = i$ else $\mu_i^{(m)}$
10:   $\alpha^{(m+1)} \leftarrow \text{UCB}(\mu^{(m+1)})$      ▷ Choose new $\alpha^{(m+1)}$ based on UCB algorithm [10]
11: end for
```

algorithm [10], which determines if the agent will receive rewards according to Equation 3 or Equation 4 during the upcoming episode. The bandit receives feedback f_m on its selection at the end of each episode. Algorithm 1 shows the full training procedure.

The key question is how to provide feedback to the bandit, given access only to intrinsic rewards. We propose a feedback mechanism grounded in the observation that the general goal in both surprise minimization and surprise maximization is for the agent to be able to affect a change in the level of surprise it experiences. In a low-entropy environment, the agent can best affect change by increasing entropy, and vice versa. Hence, the bandit should receive feedback that reflects this agency. We propose using the absolute percent difference between the entropy of the state marginal distribution at the end of the mth episode ($p_{\theta_T}^{(m)}$) and that of a random agent in the same environment ($p_{\theta_{\text{rand}}}$).

$$f_m = \left| \frac{H(p_{\theta_T}^{(m)}) - H(p_{\theta_{\text{rand}}})}{H(p_{\theta_{\text{rand}}})} \right| \tag{5}$$

To instantiate the surprise-adaptive agent, we construct an augmented MDP out of the original Markov process. Following Berseth et al. [3], this augmented MDP has a state space that includes the original state s_t, as well as the sufficient statistics of the state marginal distribution θ_t. We additionally include $\alpha^{(m)}$ as defined above, which ensures the reward function remains Markovian.

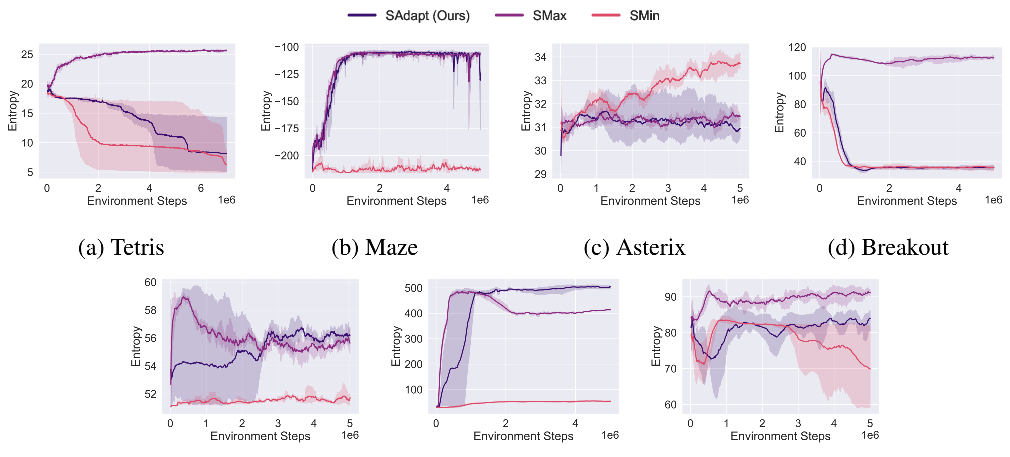
5 Experiments and analysis

Experiments are conducted to validate the surprise-adaptive agent in both low- and high-entropy environments. First, this analysis is performed over two didactic environments, each a minimal version of its respective entropy regime. For the high-entropy environment, we select the *Tetris* environment used in Berseth et al. [3] while for the low-entropy environment, we construct a maze environment (*Maze*), in which the agent navigates to a goal. Next, we apply our agent to the MinAtar [20] suite of tasks. Our method (**S-Adapt**) is compared against exclusively surprise-minimizing (**S-Min**) and exclusively surprise-maximizing (**S-Max**) agents. All agents were trained using DQN [12]. More details on environments and training can be found in Appendix A.

5.1 Adaptive entropy control

First, we consider how well our agents are able to control entropy across the two didactic environments. As expected, the **S-Min** agent achieves the lowest entropy in both environments, while the **S-Max** agent achieves the highest entropy in both environments (Figures 1a and 1b). Neither of these agents displays any adaptive behaviors. On the other hand, the **S-Adapt** agent can successfully replicate the behavior of the single-objective agents in each of the environments in which they excel.

Next, we investigate controlling entropy across the MinAtar benchmarks shown in Figure 1(c-g). Notably, these environments were not constructed with any particular entropy regime in mind. Even so, in several environments, there appears to be a wide entropy landscape for optimization. In most cases, the **S-Adapt** agent selects the correct objective for entropy control, i.e. that in which the single-objective agent has the greatest effect on entropy as training iterations increase.



(a) Tetris (b) Maze (c) Asterix (d) Breakout

(e) Freeway (f) Seaquest (g) Space Invaders

Figure 1: Entropy of the state marginal distribution versus environment interactions (average over 3 seeds, with shaded 95% CI). The **S-Adapt** agent can successfully recreate the performance of the **S-Min** agent and the **S-Max** agent in their respective didactic environments. Across the MinAtar suite, the **S-Adapt** agent generally converges to the single-objective agent with the larger change in entropy during training, i.e. learning the policy that can exert the most control over the environment.

The dynamics of the bandit are further illustrated in Figure 2, showing the average α parameter throughout training. In most environments, the bandit converges towards a single objective within the first half of the training period. In some environments, however, such as *Space Invaders* and *Asterix*, a more complex interplay between the two objectives appears to be taking place, which could lead to interesting and emergent behaviors not exhibited in either single-objective agent.

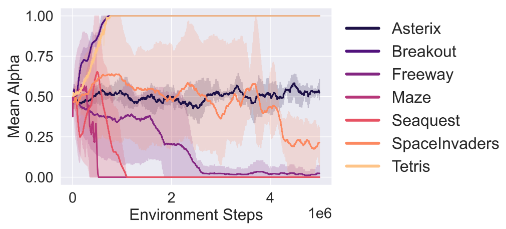


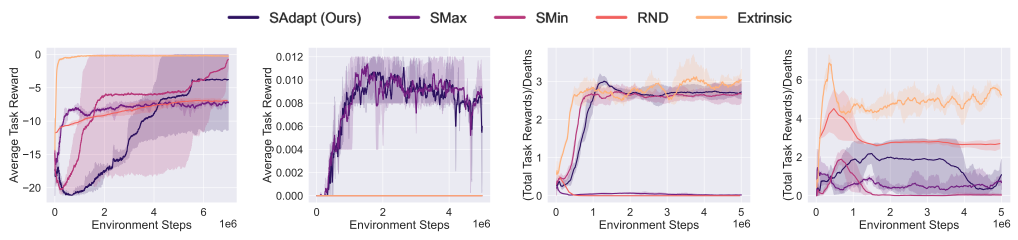
Figure 2: Rolling average of $\alpha^{(m)}$ ($\alpha^{(m)} = 0/1$ for max/min, respectively) during training (average over 3 seeds, with shaded 95% CI). In *Tetris* and *Breakout* the bandit converges towards entropy minimization, and in *Maze*, *Seaquest* and *Freeway* towards entropy maximization. In *Asterix* and *Space Invaders*, the bandit appears to be oscillating between objectives.

5.2 Entropy control and emergent behaviors

Previous work [3] has argued that controlling entropy can lead to emergent behavior. The metric for emergent behavior that we consider here is the average total extrinsic reward received throughout evaluation episodes. Since we use soft resets (see [3]), in the case of the MinAtar environments where deaths do not incur a reward penalty, we also normalize by the average number of deaths during the episode. For comparison, we include an *oracle* agent **Extrinsic**, which is a DQN agent trained with the *extrinsic* rewards, and an **RND** agent which is trained with RND rewards [4], a state-of-the-art intrinsic exploration bonus[1]. For consistency, both agents are trained with soft resets.[2]

[1]Note that we use the CleanRL [7] implementation of RND which uses PPO [18] as the RL algorithm

[2]In the *MinAtar* environments, soft resets may lead to a lower level of maximum performance than typically expected because the agent does not receive any signal upon reaching a terminal state.

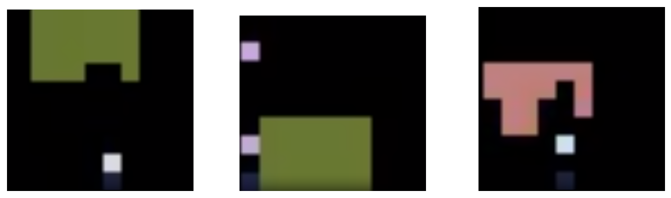


(a) Tetris (b) Maze (c) Breakout (d) Space Invaders

Figure 3: Performance on extrinsic rewards versus environment interactions in select environments (average over 3 seeds, with shaded 95% CIs). The **S-Adapt** agent successfully inherits the emergent behaviors of the **S-Min** and **S-Max** agents and even out-performs both agents in *Space Invaders* on some seeds. This outperformance indicates that some emergent abilities can manifest with the dual objectives, though the high variance across seeds indicates that these emergent behaviors are not guaranteed. The **RND** agent performs slightly better than **S-Adapt** in *Space Invaders* but worse in the other environments, demonstrating the benefit of the **S-Adapt** method vs. exploration-based rewards.

There is a distinct improvement in extrinsic rewards during training for the single-objective agents in their respective didactic environments (Figure 3a, 3b). The **S-Adapt** agent successfully reproduces these behaviors. Additionally, we observe emergent behaviors in some MinAtar environments, which were not selected with entropy control in mind. In *Breakout*, both the **S-Min** agent and the **S-Adapt** agent are able to meaningfully increase returns and in *Space Invaders*, the **S-Adapt** agent achieves higher rewards compared to single objective agents. The **S-Adapt** agent matches or outperforms the **RND** agent in the majority of environments. Extended results are available in Appendix B.

The ability for **S-Adapt** to achieve higher reward indicates that combinations of entropy control can lead to stronger results than either single objective alone. As an example of the unique emergent behavior that the **S-Adapt** agent can exhibit, we consider the example provided in Figure 4. Here, the **S-Adapt** agent is able to clear more blocks than either the **S-Min** or **S-Max** agent alone.



(a) S-Min (b) S-Max (c) S-Adapt

Figure 4: Screenshots of exemplary trained agents immediately prior to death in the *Space Invaders* environment. In this environment, the agent's goal is to use bullets to clear a block of aliens, which gradually moves toward the agent. Death occurs when an alien bullet strikes the agent or the agent contacts the block of aliens. Generally, the **S-Min** agent does not move and quickly dies from an enemy bullet. The **S-Max** agent learns to avoid enemy bullets but does not shoot at the block and hence dies when the block reaches the agent. The **S-Adapt** agent learns to shoot at the block and avoid enemy bullets long enough to clear a substantial number of aliens.

6 Conclusion

Our experiments demonstrate encouraging results for a surprise-adaptive agent. The **S-Adapt** agent can select the objective with the more controllable landscape across both didactic environments and most MinAtar environments. Moreover, the **S-Adapt** agent inherits the emergent behaviors of the single-objective agents and even shows some unique emergent behaviors in certain instances due to the complex and adaptive combination of entropy objectives. Further work is needed to understand exactly under what conditions such emergent behaviors can manifest, and how to elicit them more reliably. Moreover, an interesting extension to this work would be to apply an adaptive agent in the continual learning setting, where adaptation can occur at any time, not only at episode end.

Acknowledgements We want to acknowledge funding support from NSERC, FRQNT and CIFAR and compute support from Digital Research Alliance of Canada, Mila IDT and NVidia.

References

[1] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR, 2020.

[2] C. Bamford. Griddly: A platform for ai research in games. *Software Impacts*, 8:100066, 2021.

[3] G. Berseth, D. Geng, C. M. Devin, N. Rhinehart, C. Finn, D. Jayaraman, and S. Levine. {SM}irl: Surprise minimizing reinforcement learning in unstable environments. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=cPZOyoDloxl`.

[4] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1lJJnR5Ym`.

[5] A. Fickinger, N. Jaques, S. Parajuli, M. Chang, N. Rhinehart, G. Berseth, S. Russell, and S. Levine. Explore and control with adversarial surprise. *arXiv preprint arXiv:2107.07394*, 2021.

[6] K. Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11 (2):127–138, 2010.

[7] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *The Journal of Machine Learning Research*, 23(1):12585–12602, 2022.

[8] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJ6yPD5xg`.

[9] K. Kim, M. Sano, J. De Freitas, N. Haber, and D. Yamins. Active world model learning with progress curiosity. In *International conference on machine learning*, pages 5306–5315. PMLR, 2020.

[10] T. L. Lai, H. Robbins, et al. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[11] M. Lopes, T. Lang, M. Toussaint, and P.-Y. Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in neural information processing systems*, pages 206–214, 2012.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[13] T. Moskovitz, J. Parker-Holder, A. Pacchiano, M. Arbel, and M. Jordan. Tactical optimism and pessimism for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12849–12863, 2021.

[14] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[15] N. Rhinehart, J. Wang, G. Berseth, J. Co-Reyes, D. Hafner, C. Finn, and S. Levine. Information is power: intrinsic control via information capture. *Advances in Neural Information Processing Systems*, 34:10745–10758, 2021.

[16] J. Schmidhuber. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pages 1458–1463, 1991.

[17] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE transactions on autonomous mental development*, 2(3):230–247, 2010.

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

[19] Z. Sun and C. Firestone. The dark room problem. *Trends in Cognitive Sciences*, 24(5):346–348, 2020.

[20] K. Young and T. Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

[21] A. Zhao, M. G. Lin, Y. Li, Y. jin Liu, and G. Huang. A mixture of surprises for unsupervised reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=OHkq7qNr72-`.

A Environment and Training Details

All agents, except **RND**, were trained using DQN [12]. For all agents, we use soft resets as described in [3] which allow the agent to continue interacting in the environment after early termination, up to a static episode length, by resetting to an initial state. For the **S-Adapt** agent, we use the original UCB1 algorithm (i.e. with exploration coefficient $\sqrt{2}$). In all environments, RND was trained using the default hyperparameters and implementation from CleanRL [7].

A.1 Tetris

We take the *Tetris* environment directly from the implementation provided by the authors of [3]. In this environment, the agent receives 0 at all steps, except for a losing step which results in a -100 reward. We use an episode length of 500 for soft resets.

Following [3], we train all agents with a learning rate of 0.0003, a discount rate of 0.99, and a batch size of 256. Environment observations are flattened before being fed into a three-layer MLP with hidden dimensions 128, 64, and 32. We use a replay buffer size of 1M. We trained the agents for 7M environment interactions.

The state marginal distribution is modeled as a collection of independent Bernoulli distributions.

A.2 Maze

We constructed a custom maze environment using the Griddly platform [2]. A pixel-rendering of the environment used in our experiments can be found in Figure 5. The actual state provided to the agent is a 16x14 entity map, where each type of cell in the grid (i.e. Wall, Agent, etc) is assigned a unique numeric value. For task rewards shown in 3b, the agent receives 0 at all steps, except for when it reaches the goal, where it receives a +1 reward. We use an episode length of 250 for soft resets.

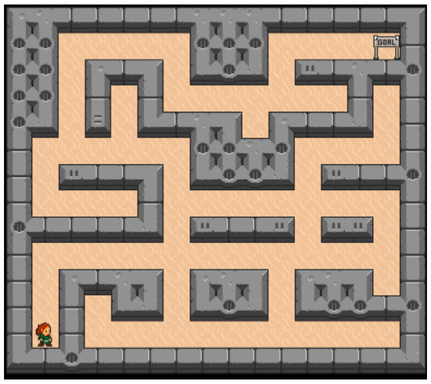


Figure 5: Pixel-rendering of the *Maze* environment

Following [3], we train all agents in *Maze* with a learning rate of 0.0003, a discount rate of 0.99, and a batch size of 256. Environment observations are flattened before being fed into a three-layer MLP with hidden dimensions 128, 64, and 32. We use a replay buffer size of 100k. We trained the agents for 5M environment interactions.

The state marginal distribution is modeled as an isotropic Gaussian distribution.

A.3 MinAtar

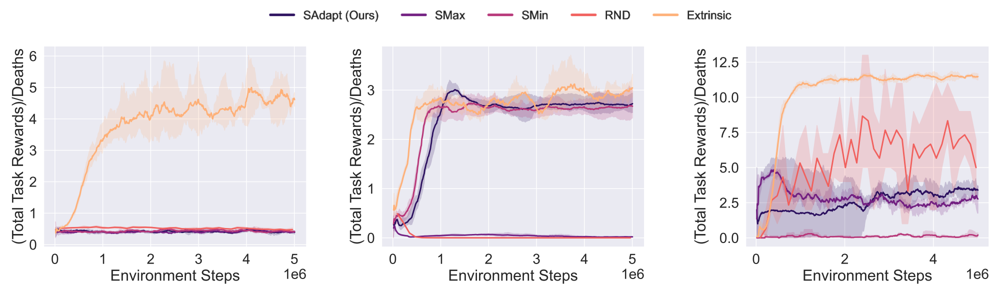
In the MinAtar environments, we select the same hyperparameters and architecture used in [20], except that we perform only 1 training update per 10 environment interactions. We use an episode length of 500 for soft resets.

The state marginal distribution is modeled as a collection of independent Bernoulli distributions.

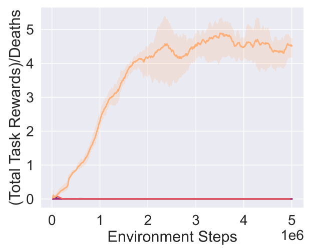
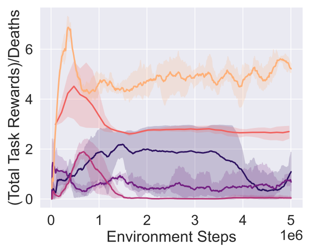
B Additional Results

B.1 MinAtar

In this section, we provide the performance for all MinAtar environments in terms of total extrinsic rewards (normalized by number of deaths).



(a) Asterix (b) Breakout (c) Freeway

(d) Seaquest (e) Space Invaders

Figure 6: Performance versus environment interactions for all tasks in the MinAtar suite. Some environments do not have a strong entropy landscape for the surprise-based agents to optimize (i.e. *Seaquest* and *Asterix*). RND performs well in some MinAtar environments where maximizing entropy is correlated with higher rewards (i.e. *Freeway*), but performs poorly in environments where minimizing entropy is correlated with higher rewards (i.e. *Breakout*).