Exploration via Elliptical Episodic Bonuses

Anonymous Author(s)
Affiliation
Address
email

Abstract

In recent years, a number of reinforcement learning (RL) methods have been proposed to explore complex environments which differ across episodes. In this work, we show that the effectiveness of these methods critically relies on a count-based episodic term in their exploration bonus. As a result, despite their success in relatively simple, noise-free settings, these methods fall short in more realistic scenarios where the state space is vast and prone to noise. To address this limitation, we introduce **E**xploration via **E**lliptical **E**pisodic **B**onuses (**E3B**), a new method which extends count-based episodic bonuses to continuous state spaces and encourages an agent to explore states that are diverse under a learned embedding within each episode. The embedding is learned using an inverse dynamics model in order to capture controllable aspects of the environment. Our method sets a new state-of-the-art across 22 challenging tasks from the MiniHack suite, without requiring task-specific inductive biases. E3B also matches current state-of-the-art methods on hard exploration tasks in VizDoom, demonstrating that it can scale to high-dimensional pixel-based observations.

1 Introduction

Exploration in environments with sparse rewards is a fundamental challenge in reinforcement learning (RL). In the tabular setting, provably optimal algorithms have existed since the early 2000s [35, 10]. More recently, exploration has been studied in the context of deep RL, and a number of empirically successful methods have been proposed, such as pseudocounts [9], intrinsic curiosity modules (ICM)[51], and random network distillation (RND) [13]. These methods rely on intrinsically generated exploration bonuses that reward the agent for visiting states that are novel according to some measure. Different measures of novelty have been proposed, such as the likelihood of a state under a learned density model, the error of a forward dynamics model, or the loss on a random prediction task. These approaches have proven effective on hard exploration problems, as exemplified by the Atari games Montezuma's Revenge and PitFall [22].

The approaches above are, however, designed for *singleton* RL tasks, where the agent is spawned in the same environment in every episode. Recently, several studies have drawn attention to the fact that RL agents exhibit poor generalization across environments, and that even minor changes to the environment can lead to substantial degradation in performance [72, 34, 71, 18, 37]. This has motivated the creation of benchmarks in the Contextual Markov Decision Process (CMDP) framework, where different episodes correspond to different environments that nevertheless share certain characteristics. Examples of CMDPs include procedurally generated (PCG) environments [15, 56, 40, 33, 17, 7, 29, 52] or embodied AI tasks where the agent must generalize its behavior to unseen physical spaces at test time [57, 59, 28, 69]. A number of methods have been proposed which have shown promising performance in PCG environments with sparse rewards, such as RIDE [55], AGAC [25] and NovelD [73]. These methods propose different intrinsic reward functions, such as the change in representation in a latent space, the divergence between the predictions of a policy and

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

an adversary, or the difference between random network prediction errors at two consecutive states. Although not presented as a central algorithmic feature, these methods also include a count-based bonus which is computed at the episode level.

In this work, we take a closer look at exploration in CMDPs, where each episode corresponds to a different environment context. We first show that, surprisingly, the count-based episodic bonus that is often included as a heuristic is in fact essential for good performance, and current methods fail if it is omitted. Furthermore, due to this dependence on a count-based term, existing methods fail on more complex tasks with irrelevant features or dynamic entities, where each observation is rarely seen more than once. We find that performance can be improved by counting certain features extracted from the observations, rather than the observations themselves. However, different features are useful for different tasks, making it difficult to design a feature extractor that performs well across all tasks.

To address this fundamental limitation, we propose a new method, E3B, which uses an elliptical episodic bonus that can be seen as a natural generalization of a count-based episodic bonus to continuous state spaces, and that is paired with a self-supervised feature learning method using an inverse dynamics model. Our algorithm is simple to implement, scalable to large or infinite state spaces, and achieves state-of-the-art performance across 22 challenging tasks from the MiniHack suite [56], without the need for task-specific prior knowledge. It also matches existing methods on hard exploration tasks from the VizDoom environment [36], demonstrating that it scales well to high dimensional pixel-based observations. Our code is available at `web-link`.

2 Background

2.1 Contextual MDPs

We consider a Contextual Markov Decision Process [1] (CMDP [30]) given by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, P, r, \mu_C, \mu_S)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{C}$ is the context space, P is the transition function, r is the reward function, μ_C is the distribution over contexts and μ_S is the conditional initial state distribution. At each episode, we sample a context $c \sim \mu_C$, an initial state $s_0 \sim \mu(\cdot|c)$, and subsequent states in the episode are sampled according to $s_{t+1} \sim P(\cdot|s_t, a_t, c)$. Let d_π^c denote the distribution over states induced by following policy π in context c. The goal is to learn a policy π which maximizes the expected return over all contexts, i.e. $R = \mathbb{E}_{c\sim\mu_C, s\sim d_\pi^c, a\sim\pi(s)}[r(s,a)]$. Examples of CMDPs include procedurally-generated environments, such as ProcGen [17], MiniGrid [15], NetHack [40], or MiniHack [56], where each context c corresponds to the random seed used to generate the environment; in this case, the number of contexts $|\mathcal{C}|$ is effectively infinite. Other examples include embodied AI environments [57, 67, 28, 59, 69], where the agent is placed in different simulated houses and must navigate to a location or find an object. In this setting, each context $c \in \mathcal{C}$ represents a house identifier and the number of houses $|\mathcal{C}|$ is typically between 20 and 500. For an in-depth review of the literature on CMDPs and generalization in RL, see [38].

2.2 Exploration Bonuses

If the environment rewards are sparse, learning a policy using simple ϵ-greedy exploration may require intractably many samples. We therefore consider methods that augment the external reward function r with an intrinsic reward bonus b. A number of intrinsic bonuses that encourage exploration in singleton (non-contextual) MDPs have been proposed, including pseudocounts [9], intrinsic curiosity modules (ICM) [51] and random network distillation (RND) error [13]. At a high level, these methods define an intrinsic reward bonus that is high if the current state is different from the previous states visited by the agent, and low if it is similar (according to some measure).

More recently, several methods have been proposed for and evaluated on procedurally-generated MDPs. All use different exploration bonuses, which are summarized in Table 1. RIDE defines a bonus based on the distance between the embeddings of two consecutive observations, NovelD uses a bonus based on the difference between two consecutive RND bonuses, and AGAC uses the KL divergence between the predictions of the agent's policy and those of an adversarial policy trained to mimic it (see the original works for details). In addition, all three methods have a term in their bonus

[1] Technically, some of the environments we consider are Contextual Partially Observed MDPs (CPOMDPs), but we follow the convention in [38] and adopt the CMDP framework for simplicity. For CPOMDPs, we use recurrent networks or frame stacking to convert to CMDPs, as done in prior work [45]

Method	Exploration bonus
RIDE [55]	$\|\phi(s_{t+1}) - \phi(s_t)\|_2 \cdot 1/\sqrt{N_e(s_{t+1})}$
NovelD [73]	$\Big[b_{\text{RND}}(s_{t+1}) - \alpha \cdot b_{\text{RND}}(s_t)\Big]_+ \cdot \mathbb{I}[N_e(s_{t+1}) = 1]$
AGAC [25]	$D_{\text{KL}}(\pi(\cdot\|s_t)\|\pi_{\text{adv}}(\cdot\|s_t)) + \beta \cdot 1/\sqrt{N_e(s_{t+1})}$

Table 1: Summary of recent exploration methods for procedurally-generated environments. Each exploration bonus has a count-based episodic term, marked in blue.

(marked in blue) which depends on $N_e(s_{t+1})$ – the number of times s_{t+1} has been encountered during the *current episode*. Although presented as a heuristic, below we will show that without this count-based episodic term, all three methods fail to learn. This in turn limits their effectiveness in more complex, dynamic, and noisy environments.

3 Importance and Limitations of Count-Based Episodic Bonuses

We now discuss in more detail the importance and limitations of the count-based episodic bonuses used in RIDE, AGAC and NovelD, which depend on $N_e(s_t)$. Figure 1 shows results for the three methods with and without their respective count-based episodic terms, on one of the MiniGrid environments used in prior work. When the count-based terms are removed, all three methods fail to learn. Similar trends apply for other MiniGrid environments (see Appendix D.1). This shows that the episodic bonus is in fact essential for good performance.

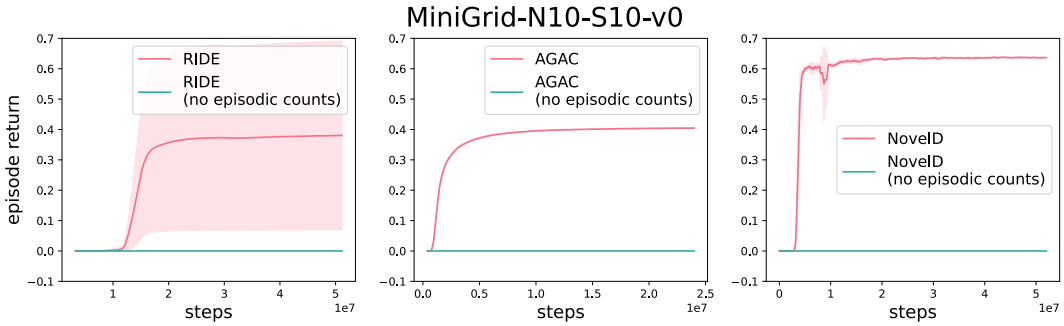

Figure 1: Results for RIDE, AGAC and NovelD with and without the count-based episodic bonus, over 5 random seeds (solid line indicates the mean, shaded region indicates one standard deviation). All three algorithms fail if the count-based episodic bonus is removed.

However, the count-based episodic bonus suffers from a fundamental limitation, which is similar to that faced by count-based approaches in general: if each state is unique, then $N_e(s_t)$ will always be 1 and the episodic bonus is no longer meaningful. This is the case for many real-world applications. For example, a household robot's state as recorded by its camera might include moving trees outside the window, clocks showing the time or images on a television screen which are not relevant for its tasks, but nevertheless make each state unique.

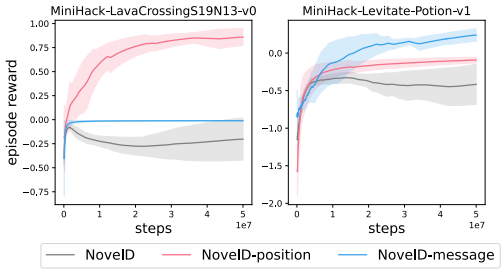

Figure 2: Performance of NovelD with different feature extractors: entire observation, agent location, or environment message. Results are averaged over 5 seeds and the shaded region represents one standard deviation.

Previous works [55, 25, 73] have used the MiniGrid test suite [15] for evaluation, where observations are less noisy and do not typically contain irrelevant information. Thus, methods relying on episodic counts have been effective in these scenarios. However, in more complex environments such as MiniHack [56] or with high-dimensional pixel-based observations, episodic count-based approaches can cease to be viable.

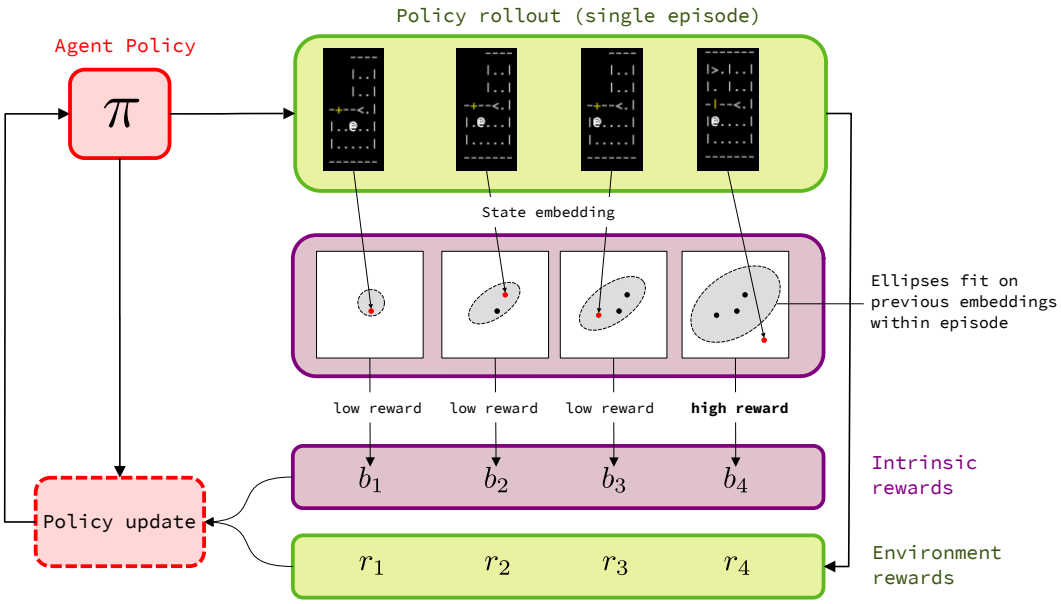

Figure 3: Overview of E3B. At each step in an episode, an ellipse is fit on the embeddings of previous states encountered within the episode. Intrinsic rewards are computed based on the position of the current state's embedding with respect to the ellipse: the further outside the ellipse, the higher the reward. These are combined with environment rewards, the policy is updated, and the process repeated. The state embeddings are learned using an inverse dynamics model.

A possible alternative could be to design a function to extract relevant features from each state and feed them to the count-based episodic bonus. Specifically, instead of defining a bonus using $N_e(s_t)$, we could define the bonus using $N_e(\phi(s_t))$, where ϕ is a hand-designed feature extractor. For example, in the paper introducing the MiniHack suite [56], the RIDE implementation uses $\phi(s_t) = (x_t, y_t)$, where (x_t, y_t) is the spatial location of the agent at time t. However, this approach relies heavily on task-specific knowledge.

Figure 2 shows results on two tasks from the MiniHack suite for three NovelD variants. The first, NovelD, denotes the standard formulation which uses the bonus $\mathbb{I}[N_e(s_t) = 1]$. The second, NovelD-position, uses the positional feature encoding described above, i.e. the episodic bonus is defined as $\mathbb{I}[N_e(\phi(s_t)) = 1]$ with $\phi(s_t) = (x_t, y_t)$. The third, NovelD-message, uses a feature encoding where $\phi(s_t)$ extracts the message portion of the state s_t similarly to [46] (both encodings are explained in more detail in Section 5). In contrast to the MiniGrid environments, here standard NovelD fails completely due to the presence of a time counter feature in the MiniHack observations, which makes each observation in the episode unique. Using the positional encoding enables NovelD-position to solve the `LavaCross-S19-N13` task, but this method fails on the `Levitate-Potion` task. On the other hand, using the message encoding enables NovelD-message to succeed on the `Levitate-Potion` task, but it fails on the `LavaCrossing-S19-N13` one. This illustrates that different feature extractors are effective on different tasks, and that designing one which is broadly effective is challenging. Therefore, robust new methods which do not require task-specific engineering are needed.

4 Elliptical Episodic Bonuses

In this section we describe **Exploration via Elliptical Episodic Bonuses**, (**E3B**), our algorithm for exploration in contextual MDPs. It is designed to address the shortcomings of count-based episodic bonuses described above, with two aims in mind. First, we would like an episodic bonus that can be used with *continuous* state representations, unlike the count-based bonus which requires discrete states. Second, we would like a representation learning method that only captures information about the environment that is relevant for the task at hand. The first requirement is met by using an elliptical

bonus, which provides a continuous analog to the count-based bonus, while the second requirement is met by using a representation learned with an inverse dynamics model [51, 55].

A summary of the method is shown in Figure 3. We define an intrinsic reward based on the position of the current state's embedding with respect to an ellipse fit on the embeddings of previous states encountered within the same episode. This bonus is then combined with the environment reward and used to update the agent's policy. The elliptical bonus and the embedding method are described in the next two sections, and the full algorithm is detailed in Appendix B.

4.1 Elliptical Episodic Bonus

Given a feature encoding ϕ, at each time step t in the episode the elliptical bonus b is defined as follows:

$$b(s_t) = \phi(s_t)^\top C_{t-1}^{-1} \phi(s_t), \qquad C_{t-1} = \sum_{i=1}^{t-1} \phi(s_i)\phi(s_i)^\top + \lambda I \tag{1}$$

Here λI is a regularization term to ensure that the matrix C_{t-1} is non-singular, where λ is a scalar coefficient and I is the identity matrix.

Intuition. One perspective is that the elliptical bonus is a natural generalization of a count-based episodic bonus [2]. To see this, observe that if the problem is tabular and ϕ is a one-hot encoding of the state, then C_{t-1} will be a diagonal matrix whose entries contain the counts corresponding to each state encountered in the episode. Its inverse C_{t-1}^{-1} will also be a diagonal matrix whose entries are inverse state visitation counts, and the bilinear form $\phi(s_t)^\top C_{t-1}^{-1}\phi(s_t)$ reads off the entry corresponding to the current state s_t, yielding a bonus of $1/N_e(s_t)$.

For a more general geometric interpretation, if $\phi(s_0), ..., \phi(s_{t-1})$ are roughly centered at zero, then C_{t-1} can be viewed as their unnormalized covariance matrix. Now consider the eigendecomposition $C_{t-1} = U^\top \Lambda U$, where Λ is the diagonal matrix whose entries are the eigenvalues $\lambda_1, ..., \lambda_n$ (these are real since C_{t-1} is symmetric). Letting $z = U\phi(s_t) = (z_1, ..., z_n)$ be the set of coordinates of $\phi(s_t)$ in the eigenspace of C_{t-1}, we can rewrite the elliptical bonus as:

$$b(s_t) = z^\top \Lambda^{-1} z = \sum_{i=1}^{n} \frac{z_i^2}{\lambda_i}$$

The bonus increases the more $\phi(s_t)$ is aligned with the eigenvectors corresponding to smaller eigenvalues of C_{t-1} (directions of low data density), and decreases the more it is aligned with eigenvectors corresponding to larger eigenvalues (directions of high data density). An illustration for $n = 2$ is shown in Figure 4. In our experiments, n is typically between 256 and 1024.

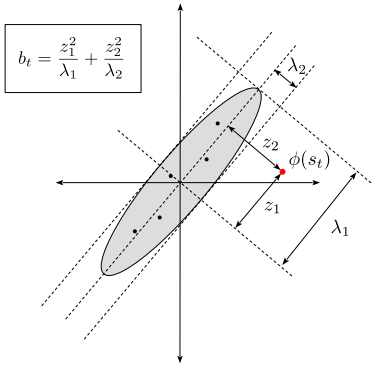

Figure 4: Illustration of the elliptical bonus in 2 dimensions.

The elliptical bonus is related to the Mahalanobis distance [43] which uses a similar bilinear form. However, the Mahalanobis distance would normalize the matrix C_{t-1} in equation 4.1 by the number of observations $t-1$ in the episode, whereas the elliptical bonus does not. The elliptical bonus thus tends to decrease with the number of observations, similarly to the count-based bonus.

Efficient Computation. The matrix C_{t-1} needs to be inverted at each step t in the episode, an operation which is cubic in the dimension of ϕ and may thus be expensive. To address this, we use the Sherman-Morrison matrix identity [60] to perform fast rank-1 updates in quadratic time:

$$C_t^{-1} = \begin{cases} \frac{1}{\lambda} I & \text{if } t = 0 \\ C_{t-1}^{-1} - \frac{C_{t-1}^{-1}\phi(s_t)\phi(s_t)^\top C_{t-1}^{-1\,\top}}{1+\phi(s_t)^\top C_{t-1}^{-1}\phi(s_t)} & \text{if } t \geq 1 \end{cases}$$

This results in an approximately $3\times$ speedup over naive matrix inversion (details in Appendix D.2).

4.2 Learned Feature Encoder

Any feature learning method could in principle be used to learn ϕ. Here we use the inverse dynamics model approach proposed in [51], which trains a model g along with ϕ to map each pair of consecutive embeddings $\phi(s_t), \phi(s_{t+1})$ to a distribution over actions a_t linking them. In our setup, ϕ is separate from the policy network. The g model is trained jointly with ϕ using the following per-sample loss:

$$\ell(s_t, a_t, s_{t+1}; \phi, g) = -\log p(a_t | g(\phi(s_t), \phi(s_{t+1})))$$

The motivation is that the mapping ϕ will discard information about the environment which is not useful for predicting the agent's actions. Previous work [51] has shown that this can make learning more robust to random noise or other parts of the state which are not controllable by the agent. In our experiments, we compare this to other approaches such as using the policy network [5] or random networks [11, 2] to produce state embeddings.

5 Experiments

5.1 MiniHack Suite

In order to probe the capabilities of existing methods and evaluate E3B, we seek CMDP environments which exhibit challenges associated with realistic scenarios, such as sparse rewards, noisy or irrelevant features, and large state spaces. We opted for the procedurally generated tasks from the MiniHack suite [56], which is itself based on the NetHack Learning Environment [40]. NetHack is a notoriously challenging roguelike video game where the agent must navigate through procedurally generated dungeons to recover a magical amulet. The MiniHack tasks contain numerous challenges such as finding and using magical objects, navigating through levels while avoiding lava, and fighting monsters. Furthermore, rewards are sparse and as detailed below, the state representation contains a large amount of information, only some of which is relevant for a given task.

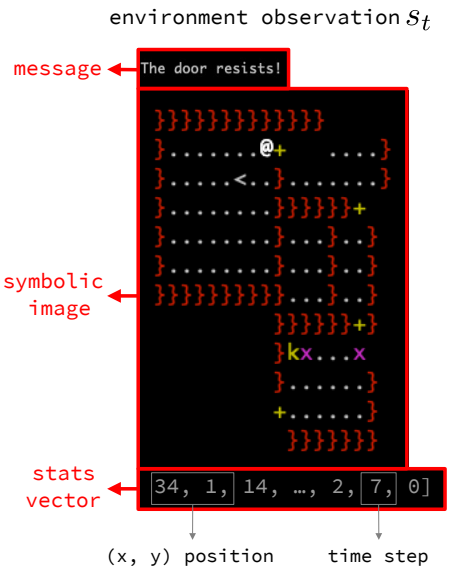

Figure 5: Observation for MiniHack

For all our experiments we use the Torchbeast [39] implementation of IMPALA [23] as our base RL algorithm. For certain skill-based tasks, we restricted the action space to only the actions which are useful for solving the task at hand, since we found that none of the methods were able to make progress with the full action space (see Appendix C.1.4 for more discussion). Full experimental details can be found in Appendix C.

5.1.1 Modalities for Episodic Bonus

The MiniHack environments provide an observation at each step that includes three different modalities: i) a symbolic image, which indicates the location of the agent, monsters or other entities, objects and different types of terrain (such as walls, lava, water); ii) a statistics vector which indicates the agent's current (x, y) location, hit points, time step t, and other features such as strength, dexterity and constitution, and iii) a textual message which gives different types of feedback about the environment ("you see a key of master thievery", "the wall is solid rock"). These are illustrated in Figure 5.

We now draw attention to some important subtleties regarding how existing methods implement count-based episodic bonuses—as we will see, these can have a large impact on performance. The works of [55, 25, 73], which use the MiniGrid suite [15] for evaluation, use a hash table whose keys are the full observations (in this case symbolic images) encountered by the agent. The occurrences of these observations are then counted to compute the episodic bonus. The baselines run in the MiniHack tasks [56], on the other hand, use a hash table whose keys are the (x, y) positions of the agent, which are extracted from the observation and are then similarly counted to compute the episodic bonus.

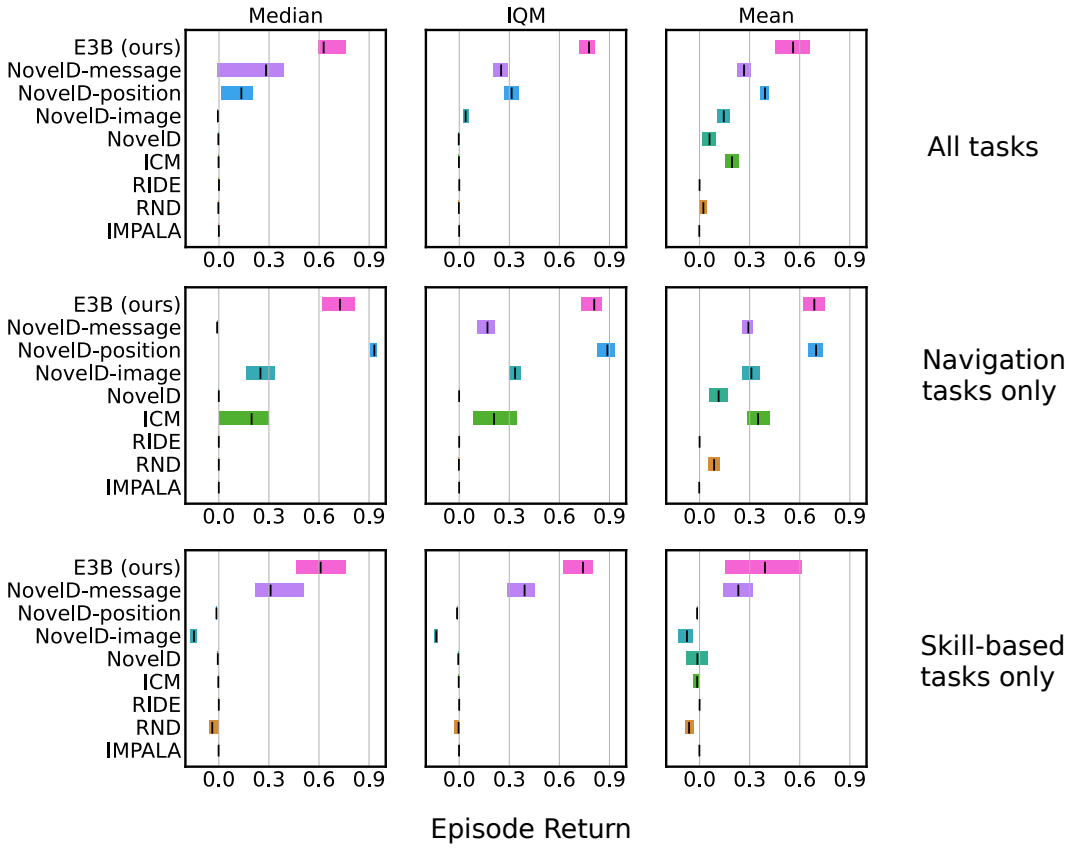

Figure 6: Aggregate results over 22 tasks from the MiniHack environment. Bars represent 95% confidence intervals computed using stratified bootstrapping with 5 random seeds.

Which of these is preferable? On one hand, using the full observation does not discard any potentially useful information, but if there are irrelevant features in the observations, similar states might be counted as separate, which can make the count-based episodic bonus meaningless. On the other hand, using the (x, y) positions might be useful for navigation-based tasks, and can discard distracting information, but might discard useful information for tasks which are not navigation based.

In order to understand the effect of these different input modalities on count-based episodic bonuses, we consider three variants of the NovelD algorithm, in addition to the original formulation. These methods all have a count-based episodic bonus of the form $\mathbb{I}[N_e(\phi(s_t)) = 1]$, with different choices of ϕ detailed below. We decided to focus our study on variants of NovelD, since it was previously shown to outperform competing methods on MiniGrid [73]. Furthermore, in preliminary experiments we were not able to get analogous variants of RIDE to yield good results (see Appendix D.3).

NOVELD: in the original version of the algorithm, ϕ is the identity. Here the input to the count-based episodic bonus is the full state, namely the concatenation of the symbol image, the message and the stats vector. This makes no assumptions about which part of the state is most useful.

NOVELD-POSITION: in this variant, ϕ extracts the (x, y) position of the agent from the statistics vector, which reflects a strong inductive bias that the task is navigation-based.

NOVELD-IMAGE: in this variant, ϕ extracts the symbol image only. This reflects an inductive bias that the message and the stats vector are not useful since they are discarded.

NOVELD-MESSAGE: in this variant, ϕ extracts the message only. This reflects an inductive bias that the messages are important but the stats and symbolic images are not.

For E3B, we feed the full state to the algorithm and do not make any assumptions about which part is useful for the task at hand. Despite the lack of task-specific inductive biases, our method is still able

to extract the relevant features for each task, thus outperforming the other exploration approaches (or matching their performance when prior knowledge of the task is used).

5.1.2 Results on MiniHack

Aggregate results for IMPALA, RND, RIDE, ICM, NovelD (with the three variants described above) and E3B over 22 sparse reward tasks from the MiniHack suite are shown in Figure 6. Out of 22 environments, 11 are based on the MiniGrid suite, but use the MiniHack interface and observation space (environment details are included in Appendix C.1.3). We used the performance metrics and bootstrapping protocol from [4] to compute confidence intervals, which are more informative than simple point estimates. We see that over all tasks (top row), E3B outperforms all other methods by a statistically significant margin across all three performance metrics.

Standard NOVELD performs poorly due to the fact that the MiniHack observations contain a time counter in the statistics vector (see Figure 5), which makes each observation in the episode unique and hence renders the count-based episodic bonus meaningless. The three NovelD variants which use different features extracted from the state for the count-based bonus perform better. Figure 6 shows aggregate performance across a subset of 13 navigation-based tasks (middle row) and 9 skill-based tasks (bottom row) (see Appendix C.1.3 for the task breakdown). On the navigation-based tasks, NOVELD-POSITION has excellent performance, since visiting a large number of different (x, y) locations is closely aligned with the true task reward. However, NOVELD-POSITION fails on all the skill-based tasks, resulting in poor performance overall. We observe an opposite trend for the NOVELD-MESSAGE variant, which performs fairly well on the skill-based tasks, but poorly on the navigation-based ones.

This highlights that although certain inductive biases can help for certain tasks, it is difficult to find one which performs well across all of them. Our method, E3B, performs well on both the navigation-based tasks and the skill-based tasks, resulting in superior performance overall. It is worth noting that E3B does not require any task-specific engineering: the exact same algorithm is run for all tasks, and it uses the unprocessed states. Results for individual tasks, along with more analysis and discussion, can be found in Appendix D.4.

5.1.3 Ablation Experiments

We next report results for ablation experiments measuring the effects of different algorithmic components of E3B, namely the feature encoding ϕ and the episodic nature of the bonus. Results are shown in Figure 7. E3B (random enc.) indicates E3B where ϕ is a randomly initialized network which is kept fixed throughout training. E3B (policy enc.) indicates E3B where the weights of ϕ are tied to those of the policy network (with the last layer producing action probabilities removed). E3B (non-episodic) indicates E3B where the elliptical bonus is computed using observations across *all* timesteps in the lifetime of the agent, and not just the current episode. All three variants perform significantly worse than E3B, which uses an inverse dynamics model encoding for ϕ and an episodic bonus. This highlights that both the inverse dynamics model and episodic bonus are important for success.

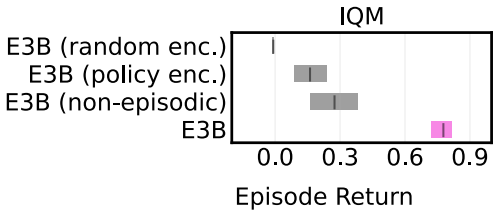

Figure 7: Interquartile Mean (IQM) performance over 22 MiniHack tasks. Bars represent 95% confidence intervals based on stratified bootstrapping.

5.2 Pixel-Based VizDoom

As our second evaluation testbed, we used the sparse reward, pixel-based VizDoom [36] environments used in prior work [51, 55]. Although these are singleton MDPs, they still constitute challenging exploration problems and probe whether our method scales to continuous high-dimensional pixel-based observations. Results comparing E3B to RIDE, ICM and IMPALA on three versions of the task are shown in Figure 8 (hyperparameters can be found in Appendix C.2). IMPALA succeeds on the dense reward task but fails on the two sparse reward ones. E3B is able to solve both versions of the sparse reward task, similar to RIDE and ICM. This provides evidence that our method is applicable in settings with pixel-based observations.

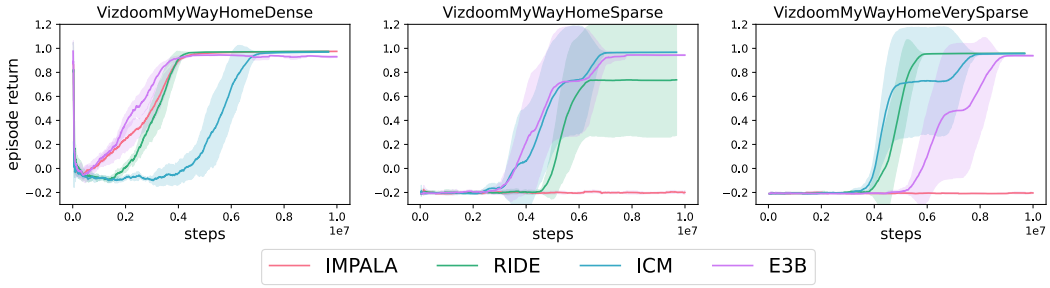

Figure 8: Results on pixel-based Vizdoom tasks with dense and sparse rewards. Results are averaged over 5 random seeds, shaded region indicates one standard deviation.

6 Related Work

Exploration in RL. Exploration remains a long-standing problem in RL. Common approaches include ϵ-greedy [66], count-based exploration [64, 8, 47, 44, 68, 42], curiosity-based exploration [58, 62, 63, 11], and other types of intrinsic motivation [48, 49, 61, 1, 12]. These methods were largely designed for singleton MDPs, where the environment remains the same across episodes. As a result, they measure novelty over the agent's lifetime in a static environment rather than on an episodic basis under a distribution of diverse contexts, as necessary in CMDPs. Other intrinsic motivation methods have recently been developed for exploration in CMDPs [55, 70, 25, 73]. As our experiments show, they critically rely on episodic count-based bonuses, which E3B generalizes.

Another class of methods automatically generate curricula over variations of the CMDP to encourage efficient learning, effectively performing a form of curiosity-driven exploration in the context space. Goal-conditioned automatic curriculum methods provide the goal subset of the context to the agent [27, 26, 24, 54, 19, 14, 22], while goal-free variants can explore contexts without *a priori* disclosing any knowledge about the context [65, 53, 32, 21]. Unlike our method, these methods assume the ability to actively configure the environment context. In principle, E3B can be combined with these approaches, whereby E3B explores at an episodic level, while the automatic curriculum method explores at a context level, making our method complementary to this body of work.

Elliptical Bonuses. The use of elliptical bonuses has a long history in the contextual bandit literature [6, 20, 41], which corresponds to the RL setting with a single time step. More recently, several works have begun to explore the use of elliptical bonuses in the context of multi-step RL. The PC-PG algorithm [2] considers MDPs with linear dynamics and uses an elliptical bonus based on a policy cover to explore in a provably efficient manner. The ACB algorithm [5] also uses an elliptical bonus, approximated using linear regressors trained on random noise, while FLAMBE [3] uses an elliptical bonus inside a learned dynamics model. All of these algorithms operate on singleton MDPs, whereas ours is designed for contextual MDPs and constructs elliptical bonuses at the episode level rather than across episodes. We also use different feature learning methods, namely inverse dynamics models, instead of the policy encoders, random networks or kernel methods used in the aforementioned works.

7 Conclusion

In this work, we identified a fundamental limitation of existing methods for exploration in CMDPs: their performance relies heavily on an episodic count-based term, which is not meaningful when each state is unique. This is a common scenario in realistic applications, and it is difficult to alleviate through feature engineering. To remedy this limitation, we introduce a new method, E3B, which extends episodic count-based methods to continuous state spaces using an elliptical episodic bonus, as well as an inverse dynamics model to automatically extract useful features from states. E3B achieves a new state-of-the-art on a wide range of complex tasks from the MiniHack suite, without the need for feature engineering. Our approach also scales to high-dimensional pixel-based environments such as VizDoom, matching the results of other top exploration methods. Future research directions include applying E3B to complex embodied AI tasks, experimenting with more advanced feature learning methods, and investigating ways to integrate within-episode and across-episode novelty bonuses.

References

[1] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[2] Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *Advances in neural information processing systems*, (33).

[3] Alekh Agarwal, Sham M. Kakade, Akshay Krishnamurthy, and Wen Sun. FLAMBE: structural complexity and representation learning of low rank mdps. *CoRR*, abs/2006.10814, 2020.

[4] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

[5] Jordan T. Ash, Cyril Zhang, Surbhi Goel, Akshay Krishnamurthy, and Sham M. Kakade. Anti-concentrated confidence bonuses for scalable exploration. *CoRR*, abs/2110.11202, 2021.

[6] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, March 2002.

[7] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[8] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[9] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 1479–1487, Red Hook, NY, USA, 2016. Curran Associates Inc.

[10] Ronen I. Brafman and Moshe Tennenholtz. R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, 2002.

[11] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[12] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[13] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[14] Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

[15] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

[16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[17] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2048–2056. PMLR, 13–18 Jul 2020.

[18] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 09–15 Jun 2019.

[19] Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Intrinsically motivated goal-conditioned reinforcement learning: a short survey. *ArXiv*, abs/2012.09830, 2020.

[20] Varsha Dani, Thomas P. Hayes, and Sham M. Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, 2008.

[21] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in Neural Information Processing Systems*, 33:13049–13061, 2020.

[22] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[23] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1406–1415. PMLR, 2018.

[24] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. In *NeurIPS*, 2019.

[25] Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. Adversarially guided actor-critic. *CoRR*, abs/2102.04376, 2021.

[26] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[27] Sébastien Forestier, Rémy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[28] Chuang Gan, Jeremy Schwartz, Seth Alter, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

[29] Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

[30] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021.

[33] Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle tower: A generalization challenge in vision, control, and planning. *CoRR*, abs/1902.01378, 2019.

[34] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Procedural level generation improves generality of deep reinforcement learning. *CoRR*, abs/1806.10729, 2018.

[35] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. In *Machine Learning*, pages 209–232. Morgan Kaufmann, 2002.

[36] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. The best paper award.

[37] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *CoRR*, abs/2111.09794, 2021.

[38] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *CoRR*, abs/2111.09794, 2021.

[39] Heinrich Küttler, Nantas Nardelli, Thibaut Lavril, Marco Selvatici, Viswanath Sivakumar, Tim Rocktäschel, and Edward Grefenstette. Torchbeast: A pytorch platform for distributed RL. *CoRR*, abs/1910.03552, 2019.

[40] Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *CoRR*, abs/2006.13760, 2020.

[41] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 661–670. ACM, 2010.

[42] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020.

[43] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[44] Jarryd Martin, Suraj Narayanan Sasikumar, Tom Everitt, and Marcus Hutter. Count-based exploration in feature space for reinforcement learning. *arXiv preprint arXiv:1706.08090*, 2017.

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[46] Jesse Mu, Victor Zhong, Roberta Raileanu, Minqi Jiang, Noah D. Goodman, Tim Rocktäschel, and Edward Grefenstette. Improving intrinsic exploration with language abstractions. *CoRR*, abs/2202.08938, 2022.

[47] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[48] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[49] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[51] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017.

[52] Aleksei Petrenko, Erik Wijmans, Brennan Shacklett, and Vladlen Koltun. Megaverse: Simulating embodied agents at one million experiences per second. In *International Conference on Machine Learning*, pages 8556–8566. PMLR, 2021.

[53] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *CoRL*, 2019.

[54] Sébastien Racanière, Andrew Kyle Lampinen, Adam Santoro, David P. Reichert, Vlad Firoiu, and Timothy P. Lillicrap. Automated curricula through setter-solver interactions. *ArXiv*, abs/1909.12892, 2019.

[55] Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*, 2020.

[56] Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Küttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. *CoRR*, abs/2109.13202, 2021.

[57] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[58] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[59] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7520–7527. IEEE, 2020.

[60] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Stat.*, 21(1):124–127, 1950.

[61] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[62] Christopher Stanton and Jeff Clune. Curiosity search: Producing generalists by encouraging individuals to continually explore and acquire skills throughout their lifetime. *PLOS ONE*, 11(9):1–20, 09 2016.

[63] Christopher Stanton and Jeff Clune. Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems. *CoRR*, abs/1806.00553, 2018.

[64] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[65] Sainbayar Sukhbaatar, Ilya Kostrikov, Arthur D. Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *ArXiv*, abs/1703.05407, 2018.

[66] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[67] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[68] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[69] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[70] Daochen Zha, Wenye Ma, Lei Yuan, Xia Hu, and Ji Liu. Rank the episodes: A simple approach for exploration in procedurally-generated environments. *arXiv preprint arXiv:2101.08152*, 2021.

[71] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

[72] Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893, 2018.

[73] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss how our method (as well as existing methods) are not able to solve MiniHack tasks with large action spaces in Section 5 as well as Appendix C.1.4.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix A.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We included the code with instructions in the supplement, and will release the code on GitHub upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We use the `rliable` library to compute error bars using stratified bootstrapping in Figure 6.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C.1.5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited all the codebases we built upon, see Section C.3.
   (b) Did you mention the license of the assets? [Yes] We mentioned all licenses associated with the codebases we used, see Section C.3.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] All of our new code is included in our code release.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]