Interesting Object, Curious Agent: Learning Task-Agnostic Exploration

Simone Parisi[1]* **Victoria Dean**[2]* **Deepak Pathak**[2] **Abhinav Gupta**[1]
[1]Facebook AI Research [2]Carnegie Mellon University

Abstract

Common approaches for task-agnostic exploration learn tabula-rasa –the agent assumes isolated environments and no prior knowledge or experience. However, in the real world, agents learn in many environments and always come with prior experiences as they explore new ones. Exploration is a lifelong process. In this paper, we propose a paradigm change in the formulation and evaluation of task-agnostic exploration. In this setup, the agent first *learns to explore* across many environments without any extrinsic goal in a task-agnostic manner. Later on, the agent effectively transfers the learned *exploration policy* to better explore new environments when solving tasks. In this context, we evaluate several baseline exploration strategies and present a simple yet effective approach to learning task-agnostic exploration policies. Our key idea is that there are two components of exploration: (1) an agent-centric component encouraging exploration of unseen parts of the environment based on an agent's belief; (2) an environment-centric component encouraging exploration of inherently interesting objects. We show that our formulation is effective and provides the most consistent exploration across several training-testing environment pairs. We also introduce benchmarks and metrics for evaluating task-agnostic exploration strategies. The source code is available at `https://github.com/sparisi/cbet/`.

1 Introduction

Exploration is one of the key unsolved problems in building intelligent agents capable of behaving like humans. In reinforcement learning (RL), exploration is usually studied under two different settings. The first is task-driven exploration, where the reward is well-defined and the agent's goal is to explore in order to maximize long-term rewards. However, in real life, external rewards are either sparse or unknown altogether. In this setting, exploration is task-agnostic: given a new environment, the agent has to explore it in absence of any external reward. Common approaches to encourage task-agnostic exploration use intrinsically motivated rewards such as prediction curiosity [35, 46], empowerment [38], or visitation counts [4, 34]. But does this setup represent how humans explore?

We argue that the commonly-used task-agnostic exploration setup is unrealistic, both from practical and academic viewpoints. This setup assumes environments in isolation and agents exploring tabula-rasa, i.e., with no prior knowledge or experience. By contrast, we as humans do not learn from one environment in isolation and we do not throw away our past knowledge every time we encounter a new environment [14]. Exploration is rather a lifelong process: every time we encounter new environments, we use our prior knowledge and experience to develop new efficient exploration strategies. In this paper, we view the exploration problem from a continual learning lens. More specifically, in this setup, the learning agent interacts with one or many environments without any extrinsic goal. At this time, the agent *learns to explore* the environments. Later on, the agent effectively transfers the learned *exploration policy* to explore new environments, rather than exploring the new environment tabula-rasa.

[*]Equal contribution. Contacts: `sparisi@fb.com` and `vdean@cmu.edu`

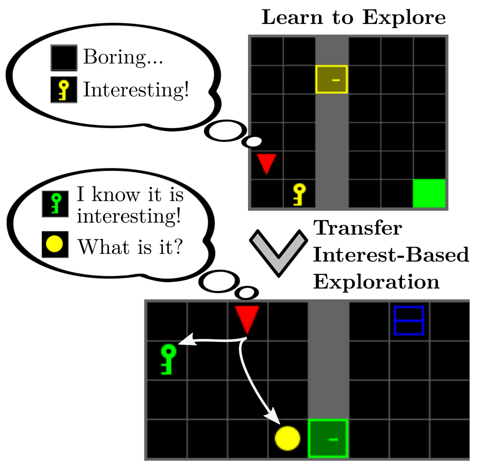

Learn to Explore
Boring...
Interesting!
I know it is interesting!
What is it?
Transfer Interest-Based Exploration

Figure 1: **Change-Based Exploration Transfer (C-BET)** trains task-agnostic exploration agents that transfer to new environments. Here the agent learns that keys are interesting, as they allow further interaction with the environment (opening doors). Later, when tasked with reaching a box behind a door, the agent starts by picking up the key.

A key question in learning how to explore is what to learn and how to transfer prior knowledge from one environment to another. Most existing task-agnostic exploration approaches, such as visitation counts, curiosity, or empowerment, define intrinsic rewards in an *agent-centric* manner: they encourage exploration of unseen parts of the environment based on the agent's own belief. In these approaches, exploration is driven by what the agent knows about the world. However, most do not make a distinction between what the agent believes it is interested in and states that would make any agent interested. For example, if the agent uses a visitation count model and has seen many objects of one kind in one environment, it would not explore the same type of objects again in a new environment. This seems to be in stark contrast to how humans explore. Consider a switch with a bell sign. Even though we might have pressed hundreds of doorbell switches (and even this instance), we are still attracted to press it. Some objects in the world just demand curiosity. We argue that apart from an 'agent-centric' component, there is an 'environment-centric' component to exploration, which can be learned from prior knowledge and experiences.

In this paper, we propose a paradigm change to move away from stand-alone isolated task-agnostic environment exploration to a more realistic multi-environment transfer-exploration setup[2]. We show how to learn exploration policies both from single- and multi-environment interaction, and how to transfer them to unseen environments. This transfer-exploration setup allows agents to use prior experiences for learning task-agnostic exploration. Notably, classic stand-alone task-agnostic approaches were designed for tabula-rasa exploration and hence only explore in an agent-centric manner. They fail to capture the inherent interestingness of some environment components. With this insight, we propose *Change-Based Exploration Transfer (C-BET)*, a simple yet effective approach learning joint agent-centric and environment-centric exploration. The key idea is for an agent to seek out both surprises (unseen areas) and high-impact (interesting) components of the environment. The experiments show that C-BET (a) learns more effectively when placed in a multi-environment setup, and (b) either outperforms or performs competitively with prior methods across several unseen testing environments. We hope this paper will inspire exploration research to focus more on learning from multiple environments and transferring experiences rather than tabula-rasa exploration.

2 Preliminaries and Related Work

We consider environments governed by Markov Decision Processes (MDPs). In MDPs, an agent observes the state of the environment s and selects actions a according to a policy $\pi(a|s)$. In turn, the environment changes, providing a new observation s' and a reward r. Through environment interaction, the agent collects episodes, i.e., sequences of states, actions and rewards $(s_t, a_t, r_t)_{t=1\dots T}$. The goal of RL is to learn a policy maximizing the sum of rewards during episodes, i.e., the return. In this setting, exploration poses many questions. If the environment provides no rewards, what should the agent look for? When should it act greedily with respect to the rewards it has found and stop looking for more? In the history of RL, many approaches have been proposed to tackle these questions. On one hand, classic single-environment approaches range from intrinsic motivation with visitation counts [2, 4, 13, 26, 52], optimism [1, 5, 25, 27, 31], or curiosity [7, 24, 35, 44, 48, 50], to bootstrapping [12, 33] or empowerment [29, 38]. On the other hand, we find approaches to incrementally learn tasks, such as transfer learning [57], continual learning [28], curriculum learning [32], and meta learning [37]. Below, we review approaches closely related to ours.

[2]While it can be argued that the real world has no explicit distinction between training and testing, we use this dichotomy only for the purpose of evaluation.

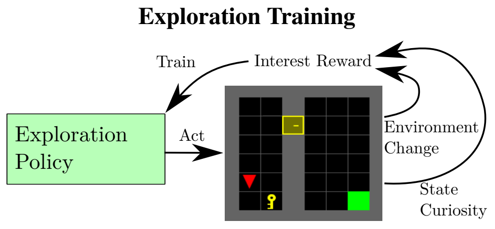


Figure 2: **C-BET pre-training.** Our agent interacts with environments and learns using intrinsic rewards computed from state and change counts.

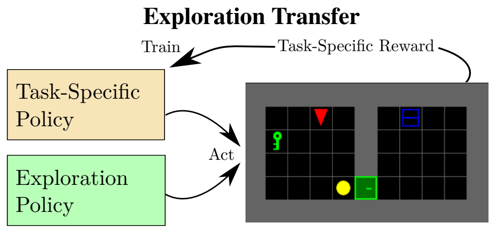


Figure 3: **C-BET transfer.** The pre-trained exploration policy is fixed and guides task-specific policy learning in new environments.

Intrinsic motivation. Exploration strategies relying on intrinsic rewards date back to Schmidhuber [46], who proposed to encourage exploration by visiting hard-to-predict states. More recently, the idea of auxiliary rewards to make up for the lack of external rewards has been extensively studied in RL, supported by evidence from psychology and neuroscience [20]. Several intrinsic rewards have been proposed, ranging from visitation count bonuses [4, 52] to bonuses based on prediction error of some quantity. For example, the agent may learn a dynamics model and try to predict the next state [24, 35, 47, 50]. By giving a bonus proportional to the prediction error, the agent is incentivized to explore unpredictable states. Schultheis et al. [48], instead, proposed to learn intrinsic rewards function by maximizing extrinsic rewards by meta-gradient.

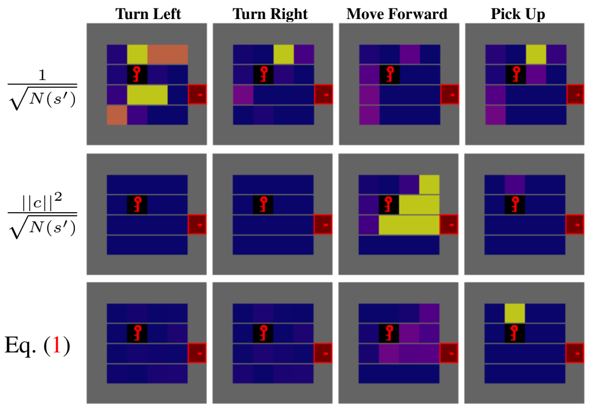
However, in these approaches exploration is agent-centric, i.e., based on an agent's belief such as the forward model error. In contrast, with this work we propose additionally learning *environment-centric* exploration policies. C-BET neither requires a model nor knowledge of extrinsic rewards. Instead, it encourages the agent to perform actions causing *interesting changes* to the environment. We should note that while Raileanu and Rocktäschel [36] proposed a similar approach, their exploration policy lacks the transfer component and also requires to learn models.

Transfer learning. The idea of agents capable of incrementally learning tasks is well-known in the field of machine learning, with the first approaches dating back to the 90s' [39, 40, 55]. In RL, recent methods have focused on policy and feature transfer. In the former, a pre-trained agent (teacher) is used to transfer behaviors to a new agent (student). Examples include policy distillation, where the student is trained to minimize the Kullback-Leibler divergence to the teacher [43] or to multiple teachers at the same time [54]. Alternative approaches, instead, directly reuse policies from source tasks to build the student policy [3, 17, 21]. In feature transfer, a pre-learned state representation is used to encourage exploration when tasks are presented to the agent [22, 58]. Similar to transfer RL, continual RL studies how learning on one or more tasks can help accelerate learning on different tasks, and how to prevent catastrophic forgetting [28, 41, 49]. Meta RL, instead, tries to exploit underlying common structures between tasks to learn new tasks more quickly [18, 37].

However, the setup in these approaches is not task-agnostic, i.e., task-specific policies are transferred rather than exploration policies. For example, after learning a policy maximizing the rewards of one task, the agent starts exploring guided by the same policy as a second task is given. Transfer is task-centric rather than task-agnostic and environment-centric. Consequently, if tasks are too dissimilar information cannot be reused, even if the environments are similar. By contrast, in this work we propose learning task-agnostic exploration from one or many environments and show transfer to unseen environments. We should note that while Pathak et al. [35] did demonstrate fine-tuning on different maze maps, their focus and large-scale evaluations remain on tabula-rasa exploration.

3 Learning to Explore

Our goal is to decouple the environment-centric nature of exploration from its agent-centric component. Contrary to prior work, we propose to first learn an environment-centric exploration policy and then to transfer it to unseen environments. The policy is driven by the inherent interestingness of states and is learned over time via interaction. First, during a pre-training phase, the agent interacts with many environments without any tasks and learns an exploration policy. Then, when new environments and tasks are presented, the agent uses the previously learned policy to explore more efficiently and learn task-specific policies. C-BET's key components are (1) a novel intrinsic reward and the learning of a policy to disentangle exploration from exploitation, and (2) the use of such policy to help exploration for new tasks. Figures 2 and 3 summarize our framework.



Gridworld with a key and a door. Observations encode cells depending on their content (e.g., 5 for the key, 10 for the agent). In each cell the agent is facing downward, and can pick up the key only from the cell above it. Samples have been collected randomly.

Figure 4: **Visualization of intrinsic rewards (row) for the agent's actions (column).** Brighter color denotes higher reward. Considering only state counts (top) does not provide useful information, and going to the corners is rewarded more than picking up the key. If we consider the L2 norm of state changes (middle), the agent is biased in favor of moving, because its position is encoded with the highest value in the observation space. The resulting policy will prefer to navigate the grid without picking up the key. By contrast, C-BET (bottom) gives picking up the key the highest reward.

3.1 Interestingness of State-Action Pairs

The natural world is filled with states or scenarios that are inherently interesting and our goal is to capture this inherent interestingness via intrinsic rewards. In this paper, we propose adding an environment-centric component of interestingness to the existing agent-centric component of surprise. Specifically, we hypothesize that the environment can *change* on interaction, and the changes that are *rare* are inherently interesting. That is, we penalize actions not affecting the environment, and favor actions producing rare changes. For instance, moving around, bumping into walls, or trying to open locked doors without keys all result in no change and thus will be of low interest.
We also want to keep the agent-centric component in exploration –that is, the exploration policy should look for surprises or unseen states. Thus, we further reward actions leading to less-visited states. By combining these two components, the resulting C-BET interest-based reward is

$$r_i(s, a, s') = 1/(N(s) + N(c')), \tag{1}$$

where $c(s, s')$ defines the *environment change* of a transition (s, a, s'), and N denotes (pseudo)counts of changes and states. Figure 4 empirically shows its effectiveness. In Section 4 we discuss change encodings used in our experiments. In Appendix B we study the performance of similar rewards based on different counts combinations.

3.2 Exploration Learning

In this phase, we want to learn task-agnostic exploration policies from interaction with many environments. The agent has no goal, but states where it can 'die' are still terminal. In this setting, we would like to treat the problem of learning exploration as an MDP with intrinsic-rewards only, and train the agent to maximize discounted intrinsic-returns averaged over episodes.
Formally, the agent explores many environments $\mathcal{E}_{\text{EXP}} = \{E_1, E_2, \ldots, E_N\}$, each governed by MDP $\langle S_n, A, P, r_i, \gamma_i \rangle$. That is, each environment has its own states but the action space is the same, and all environments obey the same dynamics P and the same intrinsic reward function r_i. The agent's goal is to learn an exploration policy maximizing the sum of discounted intrinsic rewards, i.e., $\pi_{\text{EXP}}(s, a) = \arg\max_\pi \mathbb{E}_{\mathcal{E},\pi}[\sum_t \gamma_i^t r_i(s_t, a_t)]$. To approximate the average, after a maximum number of steps the environment is reset and a new episode starts, as typically done in RL.
However, both common [7, 35, 36] and Eq. (1) intrinsic rewards decrease over time as the agent explores, to the point that they vanish to zero given enough samples. For instance, counts will grow to infinity, or prediction models error will go to zero. While this is not an issue in the tabula-rasa setup where the agent also gets extrinsic rewards, it can be problematic in the proposed task-agnostic exploration framework. Any policy, indeed, would be optimal if all rewards are zero.

To prevent the C-BET intrinsic reward of Eq. (1) from vanishing, we randomly reset counts any given time step. To explain why resets need to be random, we start by considering 'episodic counts' proposed by Raileanu and Rocktäschel [36]. These counts are reset at the beginning of every episode and are used to normalize other intrinsic rewards. In the tabula-rasa exploration setup this ensures that the agent does not go back and forth between a sequence of states with high intrinsic rewards. While episodic counts work fine when extrinsic rewards are also given, they can cause problems if intrinsic rewards are the only feedback given to the agent. The reason is that if intrinsic rewards are reset at a fixed time step then they will also have 'fixed memory'. By always starting an episode with zero-counts, the agent thinks that every state and change happening in every episode are new. If reset are episodic, 'forgetting' past transitions at a fixed time can be a problem, especially if the agent always starts an episode in the same state. Consider for example the chainworld in Figure 5. The agent always starts in A, from where it can go to B or D. From B, it loops between B and C. From D, it cannot go anywhere else. The optimal exploration policy should visit all states uniformly, by randomly going to B and D. However, if we reset counts at every episode the agent forgets that it has already visited B and C. Thus, the intrinsic rewards for B and C are high again, and trajectory ABCBCBC... gives higher intrinsic return than ADDD... Consequently, the optimal policy with respect to episodic counts will always prefer to visit B rather than D.

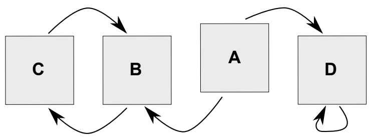


Figure 5: This chainworld illustrates that if counts are resets at the beginning of every episode, the learned policy will never visit D.

The optimal exploration policy, instead, should keep some randomness to visit the environment uniformly, prioritizing interesting states. For this reason, we propose to reset counts at any given time step with probability p. When a new episode starts, counts may not be reset yet so the agent remembers what he has visited before. As the agent explores, on average common states and changes will have higher count more often, and the agent will correctly prioritize rarer ones. In this paper, we propose $p \leq 1 - \gamma_i$ where γ_i is the intrinsic reward discount factor. This is a fitting choice because in an MDP the sum of discounted rewards can be interpreted as the expected sum of undiscounted rewards if every time step had a $1 - \gamma_i$ probability of terminating. Intuitively, this means that the discount factor γ_i implies a 'life expectancy' of $1/(1 - \gamma_i)$ steps, and thus resets should not happen more frequently than that.

The resulting MDP with rewards of Eq. (1) and random count resets can be solved by any RL algorithm. However, we should note that this MDP is non-stationary, because the agent may receive different rewards for the same state, depending on how many times the state has been visited in the past. Nonetheless, the use of classic intrinsic rewards –even in tabula-rasa exploration– either based on prediction errors [35, 36] or counts (both with and without resets) also introduces non-stationarity because these rewards change over time as well. In practice, this non-stationarity is not an issue, because intrinsic rewards change slowly over time.

3.3 Exploration Transfer

Now, the agent is presented with new environments and asked to solve tasks. Formally, each environment is governed by the classic MDP $\langle S, A, P, r, \gamma \rangle$ and the agent's goal is to learn a policy maximizing the sum of extrinsic rewards, i.e., $\pi_{\text{TASK}}(s, a) = \arg\max_\pi \mathbb{E}_\pi[\sum_t \gamma^t r(s_t, a_t)]$. Note that while during pre-training the policy was learned across all environments (one exploration policy for all environments), at transfer we learn one task-specific policy for each environment.
In this phase, the interest-based exploration policy learned in the previous phase is used to drive exploration as tasks and environments are presented to the agent. In order not to forget interestingness over time, the exploration policy is added as a fixed bias to the task-specific policy of the learning algorithm, similarly to what Hailu and Sommer [21] proposed. Thanks to the decoupling of the interest-based policy (based on the intrinsic reward) from the task-value policy (based on the extrinsic reward), the latter can be also learned independently via any RL algorithm.
In our experiments, we use IMPALA [16] to learn both π_{EXP} and π_{TASK}. IMPALA learns policies of the form $\pi(s, a) = \sigma(f(s, a))$, where σ is the softmax function. The policy is trained to maximize a function representing the value of states $V(s)$, trained on the given rewards. In our framework, we combine the two policies as follows.

- During pre-training, by using intrinsic rewards we learn $V_i(s)$ and $\pi_{\text{EXP}}(s, a) = \sigma(f_i(s, a))$.
- At transfer, we learn $V_e(s)$ on extrinsic rewards. The policy is $\pi_{\text{TASK}}(s, a) = \sigma(f_e(s, a) + f_i(s, a))$. The interestingness f_i is transferred but not trained, i.e., it acts as fixed bias to encourage interaction. Initially the policy follows f_i since f_e is initialized randomly. As it finds extrinsic rewards, the sum $f_e + f_i$ becomes greedier with respect to extrinsic rewards, and f_e slowly overtakes f_i[3].

[3]If exploration and the task goals are misaligned, we can decay exploration, e.g., $\pi_{\text{TASK}}(s, a) = \sigma(\alpha f_i(s, a) + f_e(s, a))$, where α decays over time, similarly to common ϵ-greedy policies.

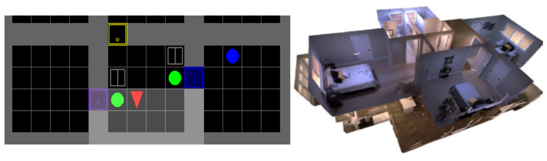


Figure 6: **Examples of the environments used in our experiments.** In MiniGrid (left), the agent navigates through a grid and interacts with different objects (keys, doors, boxes, and balls) to fulfill a task. In Habitat (right), the agent navigates through visually realistic rooms.

Note that we transfer only f_i (the policy) and not V_i (the state value). We could think of transferring V_i as fixed bias as well, i.e., by having $V_e(s) = V(s) + V_i(s)$. The policy would be trained on V_e –the states value with respect to the given task– where V_i is fixed and only V is updated. However, we believe it is more beneficial to isolate the exploratory component within the policy, in order to keep the task-specific value function targeted on extrinsic rewards. By not transferring V_i, V_e can be accurately trained on extrinsic rewards –that the agent will see often thanks to f_i from the pre-trained policy. V_e, in turn, can make π_{TASK} greedy with respect to extrinsic rewards as V_e is learned.

4 Experiments

Our experiments are designed to highlight the benefits of disentangling the environment-centric nature of exploration from agent-centric behavior by learning a separate exploration policy and then transferring it to new environments. We stress that for learning task-agnostic exploration there are no standard benchmark environments, experimental setups, well-defined evaluation metrics, or even baselines to compare against. One of our contributions is to provide an exhaustive evaluation framework for the transfer exploration paradigm.

Environments. The experiments are divided into two main sections. The first is about MiniGrid [10] (Section 4.1), a set of procedurally-generated environments where the agent can interact with many objects. The second is about Habitat [45] (Section 4.2), a navigation simulator showcasing the generality of our MiniGrid experiments to a visually realistic domain.

Change encoding. In both MiniGrid and Habitat the agent partially observes the environment, since it cannot see through corners, closed door, or inside boxes, and has a limited field of view. Rather than egocentric views (i.e., what the agent sees in front of itself), we use 360° panoramic views to count environment changes, as this is a rotation-invariant representation of the observed state. Similar to Chaplot et al. [8], we concatenate four egocentric views taken from 0°, 90°, 180°, and 270° with respect to the North. Then, the change of a transition is the difference between panoramic views $\texttt{pano}(s)$, i.e., $c(s, s') := \texttt{pano}(s') - \texttt{pano}(s)$.

Baselines. We evaluate against the following algorithms. For more details, refer to Appendix A.1.
- *Count* [4]. The intrinsic reward is inversely proportional to the next state visitation count.
- *Random Network Distillation (RND)* [7]. The intrinsic reward is the prediction error of states' random features between a trained network and a fixed one. This can be interpreted as similar to using state counts because the prediction improves states are seen more often.
- *Rewarding Impact-Driven Exploration (RIDE)* [36]. The intrinsic reward is the prediction error between consecutive embedded states, normalized by episodic state counts.
- *Curiosity* [35]. The intrinsic reward is the prediction error between consecutive states.

In Appendix B we extensively investigate the importance of count resets, panoramic changes, and different count-based rewards. Source code is available at `https://github.com/sparisi/cbet/`

4.1 MiniGrid Experiments

MiniGrid environments [10] are procedurally-generated gridworlds where the agent can interact with objects, such as keys, doors, and boxes (Figure 6). Exploration is challenging because rewards are sparse, observations are partial, and specific actions are needed to visit all states (e.g., pickup key to open door). With MiniGrid, we can generate several pairs of train and test environments that are related but still different in many ways. These pairs enable evaluation of both the learning and transfer abilities of an exploration method and its ability to deal with unseen components.

Implementation details. All environments give a 7×7×3 partial observation encoding the content of the 7×7 tiles in front of the agent (including the agent's tile). The agent cannot see through walls, closed doors, or inside boxes. The action space is discrete: left, right, forward, pick up, drop, toggle, and done. For a complete description of the environments, we refer to Appendix A.4.

Setups. We present three setups, to study different exploration transfers against tabula-rasa.

- *MultiEnv (many-to-many transfer).* The agent loops over three environments episode by episode, and learns the exploration policy using intrinsic rewards only. There is one state count and one change count for all three environments rather than separate counts for each. The environments are: KeyCorridorS3R3, BlockedUnlockPickup, and MultiRoom-N4-S5, and have been chosen for size and interaction variety: the first has both a locked and an unlocked door, a key, and a ball; the second adds a box; the third has more rooms. Note that even if these environments have all object types, the agent cannot experience all kinds of interactions. For example, it will not know that keys can be hidden in boxes, as in the ObstructedMazes. The policy is then transferred to ten environments, seven of which are new. A good intrinsic reward should help learn better exploration faster from multiple environments, thanks to sharing experience from diverse interaction.
- *SingleEnv (one-to-many transfer).* The policy is pre-trained on a single environment. DoorKey and KeyCorridor are used for pre-training because they have some –but not all– objects.
- *Tabula-rasa (no pre-training / transfer).* A task-specific policy is learned as in classic intrinsic motivation by summing intrinsic and extrinsic rewards. While it is a non-realistic setup, it is the most common RL exploration approach, and thus serves as baseline against our transfer framework.

Evaluation metrics. Our goal is to learn exploration policies that encourage interaction with the environments and transfer well to new environments, i.e., that can further be trained to solve extrinsic tasks faster. Therefore, we evaluate policies according to the following criteria.

- Unique interactions over 100 episodes at transfer to new environments, after intrinsic-reward pre-training (no extrinsic-reward training yet). Unique interactions are picks/drops/toggles resulting in new environment changes. For instance, repeatedly picking and dropping the same key in the same cell results in only two interactions.
- Tasks success rate over 100 episodes at transfer to new environments, after intrinsic-reward pre-training (no extrinsic-reward training yet). The task success rate denotes in how many episodes the exploration policy visits goal states –thus, would have already solved the environment task.
- Extrinsic return during extrinsic-reward training, after intrinsic-reward training.

4.1.1 MiniGrid Pre-Training Results

Figure 7 shows results after pre-training in MultiEnv. In Appendix C we report results for the two SingleEnv setups as well. C-BET policy both interacts with the environment and find goal states more often than all baselines. As we will see in the next section, this will result in faster extrinsic-reward learning. Furthermore, C-BET's policy transfers well to all environments, even the ones with unknown dynamics (e.g., boxes in ObstructedMazes needs to be toggled to reveal keys). Of the baselines, only Count scores high average interactions and success rate, but it does not generalize as well as C-BET. Indeed, most of Count's success comes from environments visited at pre-training (the first five), but most of its interactions are in environments with unseen dynamics (ObstructedMazes). That is, Count's policy can explore familiar environments prioritizing state coverage (high success rate and few interactions), but not unfamiliar ones (low success rate yet high interactions).

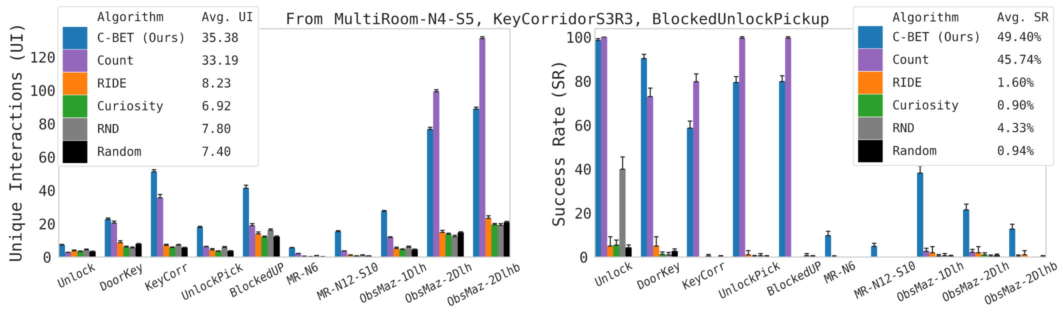


Figure 7: **Unique interactions and success rate** at the beginning of transfer of policies pre-trained in MultiEnv. Not only C-BET interacts the most and achieves the highest success rate, but also interacts and succeeds in **all** environments. Naturally, it interacts more in environment with many keys/balls/boxes to pick (KeyCorridor, BlockedUnblockPickup, ObstructedMazes), and less if there is nothing to pick (MultiRooms). On the contrary, Count overfits to the training environments and performs well only on the first five. Other baselines perform poorly, almost as a random policy.

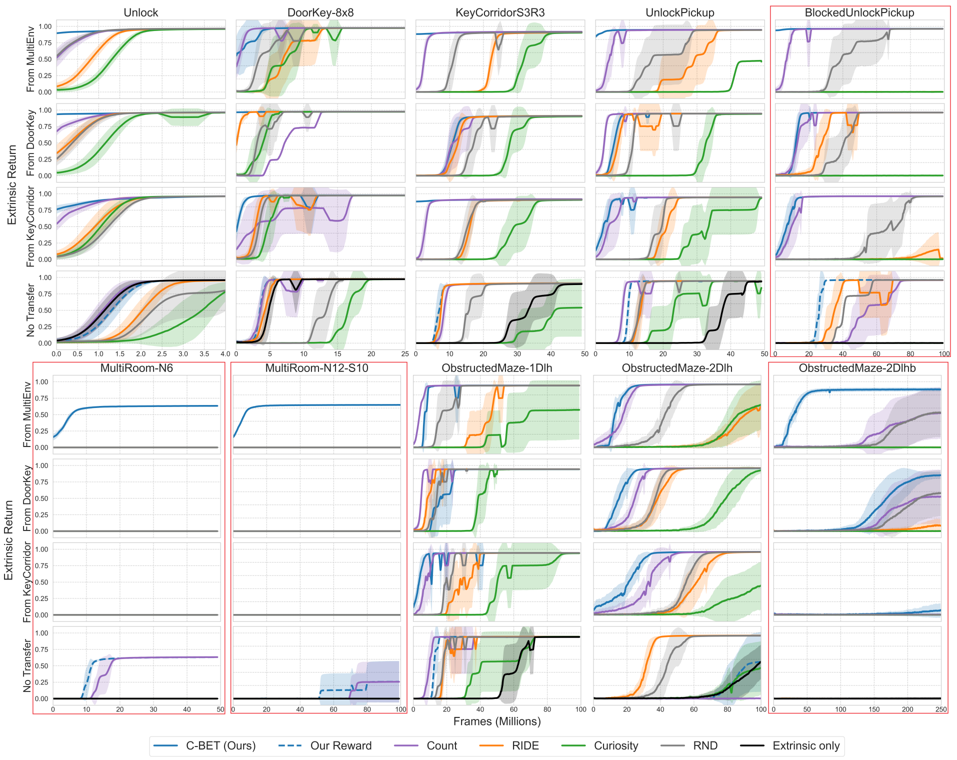


Figure 8: **MiniGrid task learning, for both transfer and tabula-rasa exploration.** The hardest tasks are outlined in red. C-BET (blue) from MultiEnv (top row under each environment) performs the best, starting with nearly optimal policies in most environments. This demonstrates the effectiveness of pre-training on multiple environments using the C-BET intrinsic reward.

Finally, RIDE, Curiosity, and RND baselines perform poorly. This is unsurprising if we consider that they rely on predictive models and that MiniGrid dynamics are deterministic and simple. Dynamics and embeddings models are learned quickly, without giving the policy time to explore. In Appendix C.3 we support this claim by showing the intrinsic reward trend during pre-training.

4.1.2 MiniGrid Transfer Results

We transfer the exploration policies learned in Figure 7 as discussed in Section 3.3. Figure 8 shows how the proposed transfer frameworks (many-to-many and one-to-many) perform against tabula-rasa exploration. Recall that seven out of ten environments are new to agents trained in MultiEnv, and nine out of ten for agents trained in DoorKey and KeyCorridor. Consequently, pre-trained task-agnostic exploration policies must generalize to unseen environments in order to perform well.
The first takeaway is that policies pre-trained with the C-BET intrinsic reward outperform baselines in both transfer and tabula-rasa. In MultiEnv transfer, C-BET performs the best, especially on the hardest environments (outlined in red). In particular, only C-BET is able to solve ObstructedMaze-2Dlhb, despite never having seen an ObstructedMaze during pre-training. Our interest-reward is also the only helping tabula-rasa exploration solving MultiRoom-N6, together with Count. Contrary to Count, though, C-BET achieves non-zero return in MultiRooms from the beginning when exploration is transferred from MultiEnv.
The second takeaway is that baselines relying on models are not suited to the transfer framework. Curiosity and RND perform better with tabula-rasa exploration in all environments except for the easiest (Unlock and DoorKey), meaning that transfer is actually harmful. RIDE performance also decreases when pre-trained on MultiEnv and KeyCorridor. These results are in line with Figure 7, where Curiosity and RND do not learn useful policies, and RIDE shows signs of interactions only in DoorKey. Furthermore, all three methods perform worst when transfer is from MultiEnv, highlighting that their intrinsic rewards are not suited to transfer, especially in a multi-environment setup.
Finally, no algorithm learns MultiRooms when transferring from SingleEnv. MultiRooms are notably different from other grids, as all doors are already unlocked and there is no object to pick up. Thus, pre-training on only DoorKey or KeyCorridor does not generalize to such diverse environments.

4.2 Habitat Experiments

To demonstrate that C-BET's efficacy extends to realistic settings with visual inputs, we perform experiments on Habitat [45] with Replica scenes [51].

Implementation details. Egocentric views have resolution 64×64×3. The action space is discrete: forward 0.25 meter, turn 10° left, and turn 10° right. To ease computational demands, we use #Exploration [53] with SimHash [9] to map both egocentric and panoramic views to hash codes and count their occurrences with a hash table. More details in Appendix A.6.

Setups. We evaluate Habitat on the *one-to-many transfer*. First, we pre-train exploration policies with only intrinsic rewards in one scene. Then, we evaluate them on new scenes without further learning. Given a fixed amount of steps, better policies will visit more of the new scenes.

Evaluation metrics. Unlike MiniGrid, we use no extrinsic rewards in Habitat. Since the agent has to navigate through rooms and spaces, we evaluate exploration policies using scene coverage measured by the agent's true state in Cartesian coordinates (not accessible by the agent)[4]. Faster, larger and more uniform coverage corresponds to better exploration. Plots show mean and confidence interval over seven random seeds per method with no smoothing.

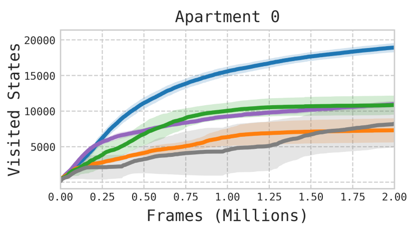
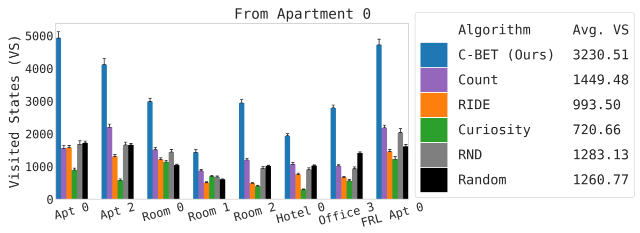


Figure 9: **Habitat pre-training.** C-BET explores the scene faster and scores the highest unique state count.



	Algorithm	Avg. VS
🟦	C-BET (Ours)	3230.51
🟪	Count	1449.48
🟧	RIDE	993.50
🟩	Curiosity	720.66
⬜	RND	1283.13
⬛	Random	1260.77

Figure 10: **Habitat offline transfer.** Bars denote the unique state count in an new scene during one episode. C-BET visits more than twice as many states than all baselines.

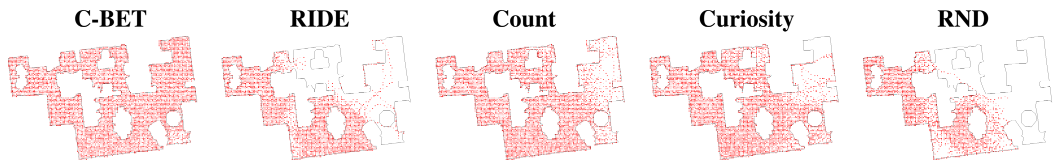


Figure 11: **States discovered by exploration policies during pre-training in the Apartment 0 scene.** Darker red cells denote higher visitation rates. Only C-BET visits all of the scene uniformly.

4.2.1 Habitat Pre-Training Results

We pre-train exploration policies on Apartment 0 (Figure 6), the largest Replica scene in the dataset. Figures 9 and 11 show state coverage throughout pre-training and its end, respectively. C-BET explores more efficiently, covering twice as much of the scene than all baselines. In particular, at the end of pre-training it has explored almost all Apartment 0 uniformly.

4.2.2 Habitat Transfer Results

Here, we evaluate state coverage of pre-trained policies in seven unseen scenes for episodes of fixed steps. A better exploration policy will exhibit generalization by covering a larger portion of the scene as evenly as possible, an impressive feat given the visual complexity of the observations. Figures 10 and 12 show that, once again, C-BET clearly outperforms all baselines. Its exploration policy transfer well to all scenes, as it uniformly discovers more states. No baseline comes closer to its results. Actually, in many scenes baselines perform worse than a random policy.

[4]To ease memory usage, we round states to 0.05 precision, e.g., 1.26 is rounded to 1.25, and 1.28 to 1.30.

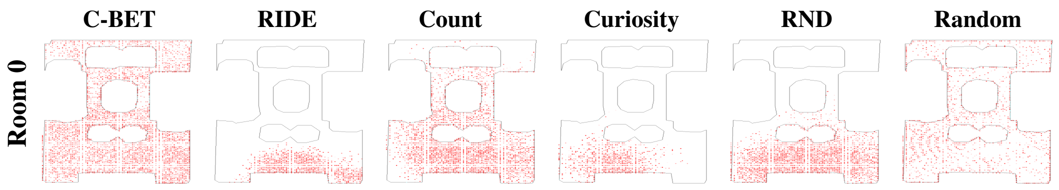


Figure 12: **States discovered by exploration policies during one episode (500 steps) at offline transfer to Room 0.** C-BET outperforms baselines and exhibits great transfer by visiting all of the scene uniformly. In the Appendix, Figure 29 shows heatmaps for all transfer scenes.

5 Discussion

In this paper, we proposed a paradigm change in task-agnostic exploration. Instead of studying task-agnostic exploration in isolated environments, we proposed to (1) learn task-agnostic exploration policies from one or multiple environments, and (2) transfer learned exploration policies to unseen environments at testing time. In our setup, the agent interacts with the environment without any extrinsic goal and learns to explore environments in a task-agnostic manner. To this end, we proposed a novel intrinsic reward to encourage interaction with the environment and the visitation of unseen states. Subsequently, our agent effectively transfers its exploration policy to unseen environments.

Advantages. The proposed two-phase framework achieves two important features, making it fundamentally different from prior work. First, we account for *environment interestingness* without relying on additional models. Instead, we use a data-driven approach, estimating the rarity of states and environment changes. Rare changes are considered more interesting, actions causing them receive higher intrinsic rewards, and the agent is encouraged to perform them again. For instance, when navigating through rooms, opening doors will be more interesting due to rarity: the agent must navigate to the corresponding key, collect it, navigate to the door, and finally open it. Thus opening a door is rarer than picking up a key, in turn rarer than simple navigation movements. Furthermore, relying on environment-centric intrinsic rewards rather than task-centric extrinsic rewards facilitates learning from multiple environments at the same time.
Second, contrary to prior transfer and continual learning algorithms we transfer policies learned on *interestingness of the environment* rather than task-specific policies. In the interest-based pre-training phase, we learn through interaction with the environment in a task-agnostic fashion, i.e., the agent freely explores the environment without any extrinsic task.

Limitations. In this paper, we assumed that interacting with the environment while looking for rare changes helps find better extrinsic rewards faster. However, exploration and the task goals may be misaligned, thus a highly exploratory policy may slow down the discovery of extrinsic rewards. For instance, the environment may have dangerous states or harmful objects that the agent should avoid, even though they would make it curious during pre-training. Furthermore, C-BET is currently tied to (pseudo)counts to compute the rarity of states and changes. While extensions to continuous spaces exist, count-based metrics are more suited for discrete spaces.

Impact. RL can positively impact real-world problems, e.g., healthcare [19], assistive robotics [15], and climate change [42]. Yet, RL may have negative impacts, e.g., in autonomous weapons or workforce displacement [6]. Our work focuses on exploration in RL. Better understanding of what is interesting to do or visit helps exploration in unseen environments, as the agent will not waste time with random actions. Similarly, transferring policies learned in a related setting –as we do– can help narrow the range of the agent's expected behavior. Conversely, in many real-world scenarios exploration by curiosity and interestingness is unacceptable. For instance, autonomous cars cannot run over pedestrians just for the sake of curiosity. At present, our work is far from these impacts, but we hope to direct research to focus more on learning from multiple environments and transferring experiences, while at the same time ensuring the safety and reliability of autonomous agents.

Acknowledgments and Disclosure of Funding

The authors would like to thank Davide Tateo for his thoughtful discussions, Roberta Raileanu for providing support for RIDE's codebase, and Sudeep Dasari for helping us with the experiments.

References

[1] P. Auer and R. Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 49–56, 2007. 2

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002. 2

[3] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. Van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 3

[4] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 1, 2, 3, 6

[5] R. I. Brafman and M. Tennenholtz. R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3(Oct): 213–231, 2002. 2

[6] E. Brynjolfsson and T. Mitchell. What can machine learning do? workforce implications. *Science*, 358(6370), 2017. 10

[7] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019. 2, 4, 6

[8] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12875–12884, 2020. 6

[9] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Symposium on Theory of Computing*, 2002. 9, 16

[10] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic Gridworld Environment for OpenAI Gym, 2018. URL `https://github.com/maximecb/gym-minigrid`. 6

[11] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs), 2015. 16

[12] C. D'Eramo, A. Cini, and M. Restelli. Exploiting Action-Value uncertainty to drive exploration in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2019. 2

[13] K. Dong, Y. Wang, X. Chen, and L. Wang. Q-learning with UCB exploration is sample efficient for Infinite-Horizon MDP. In *International Conference on Learning Representation (ICLR)*, 2020. 2

[14] R. Dubey, P. Agrawal, D. Pathak, T. L. Griffiths, and A. A. Efros. Investigating human priors for playing video games. In *International Conference on Machine Learning (ICML)*, 2018. 1

[15] Z. Erickson, V. Gangaram, A. Kapusta, C. K. Liu, and C. C. Kemp. Assistive gym: A physics simulation framework for assistive robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10169–10176. IEEE, 2020. 10

[16] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018. 5, 16

[17] F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006. 3

[18] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017. 3

[19] O. Gottesman, F. Johansson, M. Komorowski, A. Faisal, D. Sontag, F. Doshi-Velez, and L. A. Celi. Guidelines for reinforcement learning in healthcare. *Nature medicine*, 25(1):16–18, 2019. 10

[20] J. Gottlieb, P. Oudeyer, M. Lopes, and A. Baranes. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in Cognitive Sciences*, 17(11):585–593, 2013. 3

[21] G. Hailu and G. Sommer. On amount and quality of bias in reinforcement learning. In *International Conference on Systems, Man, and Cybernetics (SMC)*, 1999. 3, 5

[22] S. Hansen, W. Dabney, A. Barreto, T. Van de Wiele, D. Warde-Farley, and V. Mnih. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations (ICLR)*, 2020. 3

[23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. 16

[24] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. VIME: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2, 3

[25] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 11(Apr):1563–1600, 2010. 2

[26] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan. Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 2

[27] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002. 2

[28] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, 114(13):3521–3526, 2017. 2, 3

[29] A. S. Klyubin, D. Polani, and C. L. Nehaniv. All else being equal be empowered. In *European Conference on Artificial Life*, 2005. 2

[30] H. Küttler, N. Nardelli, T. Lavril, M. Selvatici, V. Sivakumar, T. Rocktäschel, and E. Grefenstette. TorchBeast: A PyTorch Platform for Distributed RL, 2019. URL https://github.com/facebookresearch/torchbeast. 16

[31] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6 (1):4–22, Mar 1985. 2

[32] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020. 2

[33] I. Osband, B. V. Roy, D. J. Russo, and Z. Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research (JMLR)*, 20(124):1–62, 2019. 2

[34] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning (ICML)*, 2017. 1

[35] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, 2017. 1, 2, 3, 4, 5, 6, 28

[36] R. Raileanu and T. Rocktäschel. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *International Conference on Learning Representations (ICLR)*, 2020. 3, 4, 5, 6, 16, 28

[37] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning (ICML)*, 2019. 2, 3

[38] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 2015. 1, 2

[39] M. B. Ring. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712, 1994. 3

[40] M. B. Ring. CHILD: A first step towards continual learning. In *Learning to learn*, pages 261–292. Springer, 1998. 3

[41] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3

[42] D. Rolnick, P. L. Donti, L. H. Kaack, K. Kochanski, A. Lacoste, K. Sankaran, A. S. Ross, N. Milojevic-Dupont, N. Jaques, A. Waldman-Brown, et al. Tackling climate change with machine learning. *arXiv preprint arXiv:1906.05433*, 2019. 10

[43] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. In *International Conference on Learning Representations (ICLR)*, 2015. 3

[44] R. M. Ryan and E. L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 25(1):54–67, 2000. 2

[45] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *International Conference on Computer Vision (ICCV)*, 2019. 6, 9

[46] J. Schmidhuber. A possibility for lmplementing curiosity and boredom in Model-Building neural controllers. In *International Conference on Simulation of Adaptive Behavior (SAB)*, 1991. 1, 3

[47] J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006. 3

[48] M. Schultheis, B. Belousov, H. Abdulsamad, and J. Peters. Receding horizon curiosity. In *Conference on Robot Learning (CoRL)*, 2019. 2, 3

[49] J. Schwarz, J. Luketina, W. M. Czarnecki, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine learning (ICML)*, 2018. 3

[50] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. In *NIPS Workshop on Deep Reinforcement Learning*, 2015. 2, 3

[51] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. Newcombe. The Replica dataset: A digital replica of indoor spaces, 2019. 9

[52] A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences (JCSS)*, 74(8):1309–1331, 2008. 2, 3

[53] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 9, 16

[54] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 3

[55] S. Thrun and T. M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15 (1-2):25–46, 1995. 3

[56] T. Tieleman and G. Hinton. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *Technical Report.*, 2017. 16

[57] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016. 2

[58] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning (ICML)*, 2021. 3

Checklist

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] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss this in Section 5.
   (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] Source code is available at `https://github.com/sparisi/cbet/`
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify all the training details in Appendix A.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All our plots show confidence intervals either as shaded areas or error bars.
   (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] We specify all the compute details in Appendix A.3.
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 provide bibliographic references or URLs for the relevant resources.
   (b) Did you mention the license of the assets? [No] We provide instead links or references that will allow the interested reader to find out about the licenses of the various asset.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Source code is available at `https://github.com/sparisi/cbet/`
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We are using existing assets in compliance with their respective licenses, that do not require direct consent by the creators.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Our code does not contain personally identifiable information or offensive content.
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]

A Supplemental Details

A.1 Algorithms Details

Here, we formally define all intrinsic rewards evaluated in the paper. Let's denote by $N(s)$ the state visitation (pseudo)count, by $N(c)$ the state-change visitation (pseudo)count, and by $\phi(s)$ some state embeddings parameterized by a neural network. Formally, the rewards $r_i(s, a, s')$ we evaluate are:

- **C-BET (Ours)**: $r_i = 1/(N(c) + N(s'))$. At pre-training, both counts are reset with probability $p \leq 1 - \gamma_i$ at every step.
- **Count**: $r_i = 1/\sqrt{N(s')}$. $N(s')$ is never reset.
- **RND**: $r_i = ||\phi(s') - \hat{\phi}(s')||^2$, where $\hat{\phi}$ is a fixed random network, and ϕ is trained to minimize the same error.
- **Curiosity**: $r_i = ||f(\phi(s), a) - \phi(s')||^2$, where ϕ is trained to minimize the prediction error of both an inverse and a forward model, and f is the forward model.
- **RIDE**: $r_i = ||\phi(s) - \phi(s')||^2/\sqrt{N(s')}$, where ϕ is trained to minimize the prediction error of both an inverse and a forward model. $N(s')$ is reset at the beginning of every episode during both pre-training and transfer.

A.2 Hyperparameter Details

Experiments are built on the codebase developed by Raileanu and Rocktäschel [36], which includes the Torchbeast implementation of IMPALA [30]. All algorithms use the same network architectures.

- **Policy and value function**. The input is the environment partial observation, which is $7\times7\times3$ for MiniGrid and $64\times64\times3$ for Habitat. It is passed through three (for MiniGrid) or five (for Habitat) convolutional layers with 32 filters each, kernel size 3×3, stride 2, and padding 1. An exponential linear unit [11] is after each convolution layer. The output of the last convolution layer is fed into two layers with ReLU activation and 1,024 units, and then into an LSTM [23] with 1,024 units. Finally, two separate fully connected layers of 1,024 units each are after the LSTM, and are used for the value function and the policy, respectively.
- **State embeddings**. The partial observation is passed through five convolutional layers with 32 filters each, kernel size 3×3, stride 2, and padding 1.
- **Inverse model**. The input is the concatenation of two successive state embeddings. It is fed into two layers with ReLU activation and 256 units.
- **Forward model**. The input is the concatenation of the state embedding and the action. It is fed into two layers with ReLU activation and 256 and 128 units, respectively.

Observations and changes counts are based on egocentric and panoramic views, respectively. For MiniGrid, egocentric views are 147-dimensional while panoramic views are 588-dimensional.
For Habitat, views are encoded using HashSim [9, 53] to ease computational demands. Habitat views are 12,288- and 49,152-dimensional and are hashed with 128 bits.

Value functions and policies are updated every 100 steps with IMPALA [16], using mini-batches of 32 samples and the RMSProp optimizer [56] (learning rate 0.0001 linearly decaying to 0, momentum 0, and $\varepsilon = 0.00001$). The gradient is clipped at 40.

Both the intrinsic and extrinsic rewards discount factors are 0.99 as goals can be reached in less than 100 steps. This is also the default discount factor used by Raileanu and Rocktäschel [36]. The counts reset probability is $p = 0.001$, meaning that the agent has an approximate 'life expectancy' of 1,000 steps. This is a fitting choice because the largest time horizon for the environments is 640 steps (see Appendix A.4), and $p \leq 1 - \gamma_i$.

Intrinsic rewards and losses are further scaled down by a coefficient for numerical stability.

- Intrinsic reward: 0.1 (RIDE, RND, Curiosity), 0.005 (C-BET, Count).
- Policy entropy loss (all algorithms): 0.0005.
- Value function loss (all algorithms): 0.5.
- Forward model loss (RIDE, Curiosity): 10.
- Inverse model loss (RIDE, Curiosity): 0.1.
- Random network loss (RND): 0.1.

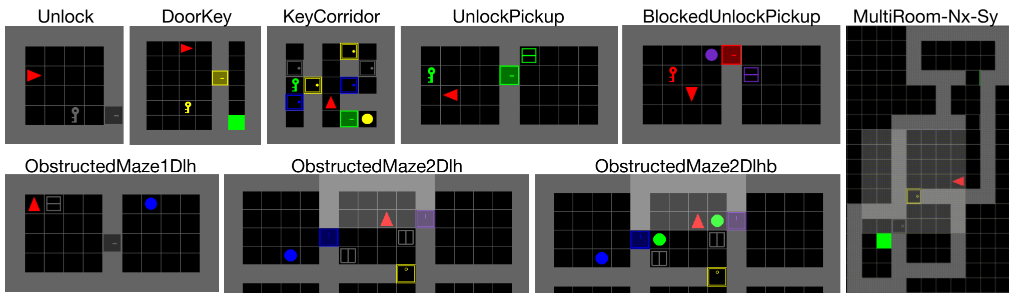


Figure 13: **The MiniGrid environments.** The agent has to navigate through a grid and interact with different objects (keys, doors, boxes, balls) to fulfil a task. At each episode, the grids are procedurally generated, changing rooms layout, objects positioning and color.

A.3 Compute Details

Experiments were run on a SLURM-based cluster, using a NVIDIA Quadro GP100 GPU and 40 CPUs. For MiniGrid, IMPALA uses 40 actors. One pre-training run takes ~8 hours for 50M steps. Learning time after transfer ranges from ~30 minutes (5M steps in *Unlock*), to ~55 hours (200M steps in *ObstructedMaze2Dlhb*). For Habitat, IMPALA uses 4 actors to prevent memory errors. One pre-training run takes ~45 hours for 2M steps (because Habitat simulation is slower, network inputs –observations– are larger, and true state counts are saved for visitation heatmaps).

A.4 MiniGrid Environments Details

Figure 13 shows the MiniGrid environments used in this paper. Below is a list of their respective tasks (the hardest are bolded). T denotes the maximum number of steps per episode. Everything is as implemented by default in MiniGrid codebase.

- *Unlock*: pick up the key and unlock the door ($T = 288$).
- *DoorKey-8x8*: pick up the key, unlock the door, and go to green goal ($T = 640$).
- *KeyCorridorS3R3*: pick up the key, unlock the door, and pick up the ball (only the door before the ball is locked) ($T = 270$).
- *UnlockPickup*: pick up the key, unlock the door, and open the box ($T = 288$).
- ***BlockedUnlockPickup***: pick up the ball in front of the door, drop it somewhere else, pick up the key, unlock the door, and open the box ($T = 576$).
- *ObstructedMaze1Dlh*: open the box to reveal the key, pick it up, unlock the door, and pick up the ball ($T = 288$).
- *ObstructedMaze2Dlh*: same as above, but with two doors to unlock ($T = 576$).
- ***ObstructedMaze2Dlhb***: same as above, but with two balls in front of the doors (like in BlockedUnblockPickup) ($T = 576$).
- ***MultiRoom-N6***: navigate through six rooms of maximum size ten and go to the green goal (all doors are already unlocked) ($T = 120$).
- ***MultiRoom-N12-S10***: same as above, but with ten rooms of maximum size twelve ($T = 240$).
- *MultiRoom-N4-S5*: smaller MultiRoom environment (five rooms of maximum size five), used only for pre-training ($T = 100$).

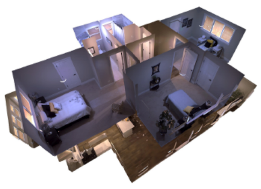
In all tasks, the extrinsic reward is r_t = 1 - 0.9 (t/T). This reward is given only for solving the task. The action space is discrete with seven actions: left, right, forward, pick up, drop, toggle, and done. 'Toggle' unlocks a door if the agent has the corresponding key, opens/closes a door if unlocked, and open boxes to reveal keys. 'Done' is implemented by default in MiniGrid codebase and it is used for language-based tasks only, thus it does nothing in our tasks.
The grid is procedurally generated at each episode, and the agent's initial position is random within a fixed area far from the goal (e.g., in DoorKey the agent starts in the area with the key, or in MultiRooms it starts in the farthest room from the goal).
BlockedUnblockPickup, ObstructedMaze2Dlh, and ObstructedMaze2Dlhb provide all types of interaction, but only ObstructedMazes have hidden keys. MultiRooms differ from the other environments because they have more rooms, all doors are already unlocked, and the goals are further away from the agent start position. They require more steps to be completed, and this explains their smallest extrinsic return compared to other environments in Figure 8.

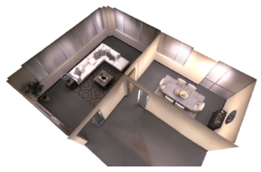
A.5 MiniGrid Plot Smoothing Details

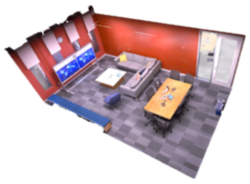
MiniGrid's plots show the mean and standard error over seven random seeds per method, smoothed over 300 epochs with a sliding window. Each epoch consists of 3,200 samples (32 mini-batches of 100 steps). These samples are used for both updating the agent and computing evaluation metrics. This smoothing is necessary because we need to wait for the end of the episode to compute the return (i.e., the sum of rewards), but environments' episodes are longer than 100 steps (see Appendix A.4). Therefore, for some epochs we cannot compute the expected return and we end up having less than one evaluation data point per epoch.
On the contrary, for Habitat visitation counts we can always retrieve the number of visited states at every step without waiting for the end of the episode, therefore there is no need for smoothing.

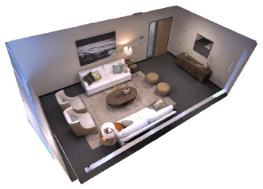
A.6 Habitat Environments Details

Figure 14 shows the Replica scenes used in this paper. At each step, the agent can either move forward by 0.25 meter or turn left/right by $10°$. The RL task is to reach a fixed (x, y) position within 0.2 meters radius. The episode time limit is $T = 500$ steps. The agent initial position is drawn randomly from the set of navigable points, and ensured to be compatible with the goal.
Egocentric views (used as policy input and to count states) have resolution $64\times64\times3$. Figure 15 shows the agent's $360°$ panoramic view, used to count environment changes.

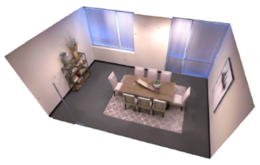

(a) Apartment 0

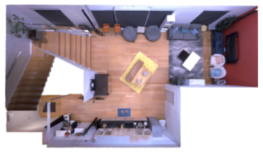

(b) Apartment 1


(c) Office 3


(d) Room 0

(e) Room 1


(f) Room 2


(g) FRL Apartment 0

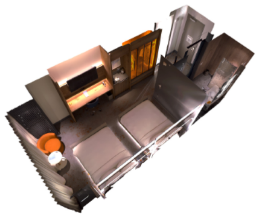

(h) Hotel 0

Figure 14: **Real-world scenes from the Replica dataset used for our Habitat experiments.** The scenes have several rooms and obstacles that make exploration challenging. Apartment 0 is used for pre-training, while the remainder are used to evaluate transfer policies.

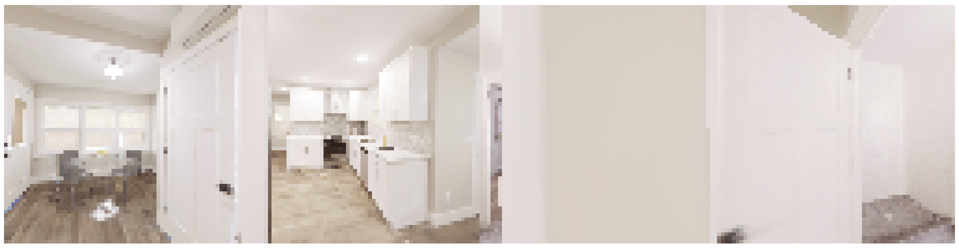


Figure 15: **Agent's 360° panoramic view of Apartment 0.** The view concatenates four egocentric images taken from 0°, 90°, 180°, and 270° with respect to the North.

B Ablation Study on MiniGrid

In this paper, we propose the following C-BET intrinsic reward:

$$r_i(s, a, s') = \frac{1}{\underbrace{N(c)}_{\text{change part}} + \underbrace{N(s')}_{\text{state part}}}, \tag{2}$$

where $c(s, s')$ is the environment change of a transition (s, a, s'), and N denotes (pseudo)counts of changes and states. Both counts are randomly reset at any given time step, independently from each other. We use this reward for two reasons. First, as shown in Figure 16, summing the two parts **within** the square root yields high rewards **only** to transitions with **both** low state and change counts. Second, the ablation study in Appendix B.2 shows that squaring the reward performs best.

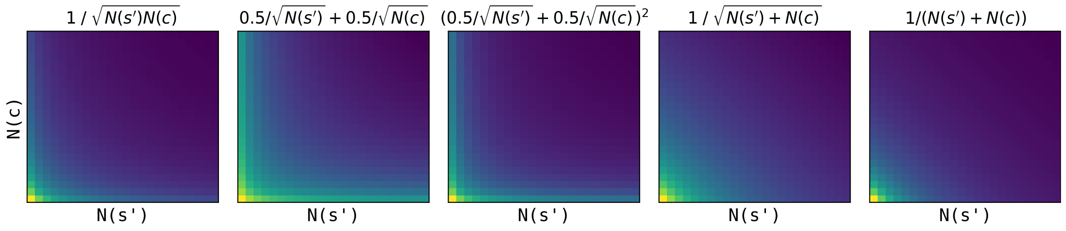


Figure 16: Heatmaps of different intrinsic rewards on varying state and change counts. The brighter the color, the higher the reward. Considering the sum of the parts –either squared or not– (second and third maps) still rewards transitions with either low state or change counts –not necessarily both. This is shown by the bright edges of the heatmap. The same happens with only the product (first map), albeit to a lesser extent. If we sum the parts inside the square root (fourth and fifth map) we correctly reward only transitions with both low state and change counts. Between the latter two, squaring the reward further penalizes transitions with high counts, because the reward decreases faster as either the state or change count increases.

Aim of the Study

The key components of C-BET reward are count random resets and the change-based reward. In this section, we investigate these components and answer the following fundamental questions:

- How important are count resets?
- How important is how we encode changes? Are panoramic views crucial for C-BET?
- How do different combinations of the state and change terms in Figure 16 perform? Is Eq. 2 truly the best?
- How do pre-training environments affect the quality of the exploration policy?

We start by investigating if panoramic views are good for counting states, and if squaring the reward can help state-only baseline as well (Appendix B.1). We continue by comparing the different types of rewards shown in Figure 16 (Appendix B.2). Subsequently, we evaluate C-BET with egocentric views for counting changes (Appendix B.3). Then, we further review interesting questions arising from such evaluation regarding change-based rewards (Appendix B.4). Finally, we study the importance of count resets. Finally, we conclude by showing what really matters for C-BET (Appendix B.5).

As evaluation metric, we consider the ‘average episode success’ of pre-trained policies, in both the *one-to-many* (SingleEnv) and *many-to-many* (MultiEnv) setups discussed in Section 4. After pre-training, policies are transferred to new environments and have to solve their respective task. Without further training, we evaluate their average success rate over 100 episodes. The higher the success rate, the faster the policy is likely to solve the task if further extrinsic-reward training would be carried one, as the extrinsic reward will be seen more often.
We present results through error bar plots and recap tables, reporting mean and confidence interval over seven seeds. Error bar plots show the success rate for each new environment a policy is transferred to, and each setup (SingleEnv and MultiEnv) has a dedicated plot. Figure legends average the rate over the ten environments. Tables further average it over the three setups.

B.1 State-Only Counts: Egocentric vs Panoramic Views, Squaring vs Classic Reward

Questions: Are panoramic views good for counting states? Does squaring the state-only baseline increase its performance?
Discussion: Figure 17 shows that egocentric views (plain patch) perform best, with an average success rate of 28.3% over the three setups. Panoramic views (dashed patch) perform well in MultiEnv but poorly in SingleEnvs, for an average success rate of 24.9%. The reason is that panoramic views are rotation-invariant: when the agent turns left/right the panoramic state does not change, the state count increases and the intrinsic reward decreases. In the end, the agent will not be encourage to turn at all. This is not surprising, because panoramic views are designed to represent *environment changes* rather than the agent's state[5]. Only in MultiEnv panoramic views perform well. Thanks to the diversity of visited states, counts do not increase too rapidly, thus turning around is not penalized too much. In Appendix B.4 we further investigate this issue with 'change-only counts' rewards.
Figure 17 also shows that squaring the reward (yellow) performs slightly worse, since policies seem to overfit to the training environments.

$\frac{1}{\sqrt{N(s')}}$, Ego	$\frac{1}{N(s')}$, Ego	$\frac{1}{\sqrt{N(s')}}$, Pano	$\frac{1}{N(s')}$, Pano
28.35%	**29.08%**	24.90%	20.93%

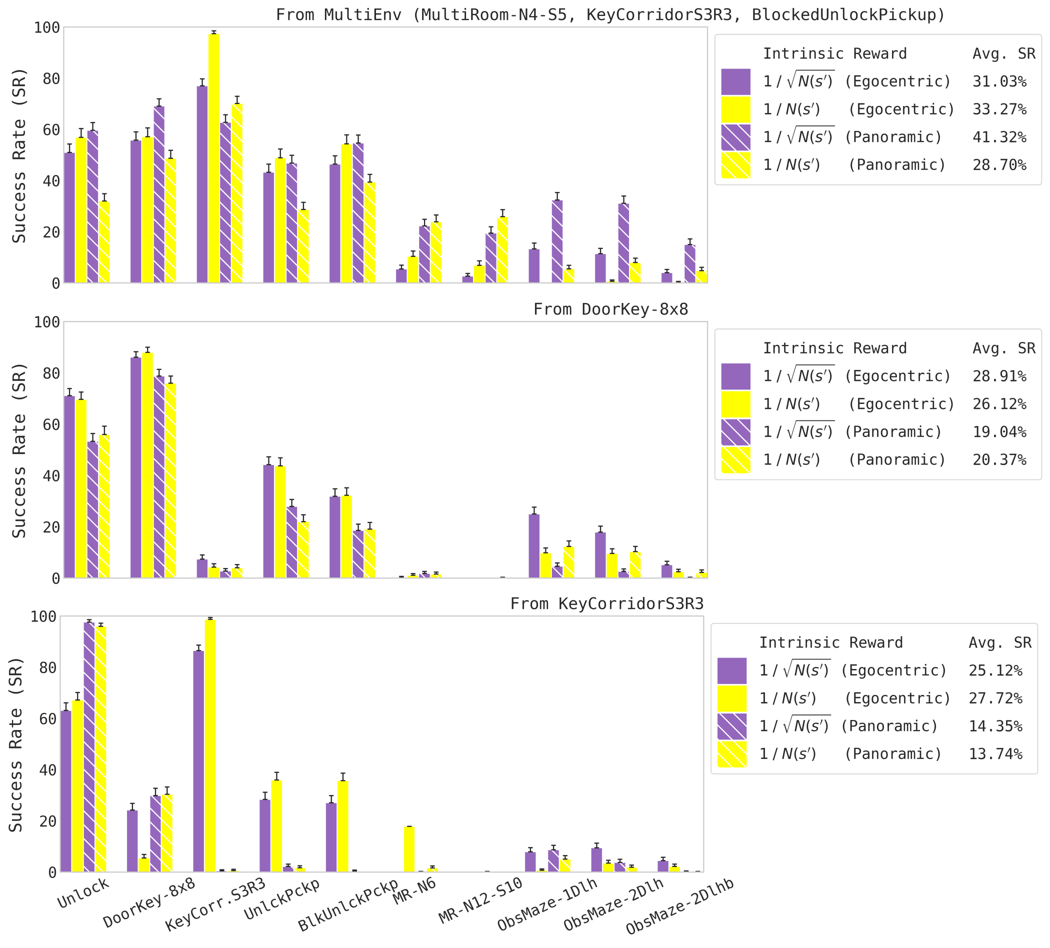


Figure 17: Success rate of policies pre-trained on different state-only intrinsic rewards with count resets. Squared rewards tend to overfit to the training environments. Panoramic views perform worse than egocentric views in SingleEnvs, but better in MultiEnv due to the diversity of visited states.

[5]Note that panoramic views are used only for the intrinsic reward. The policy still receives egocentric views.

B.2 C-BET: Different Reward Combinations

Question: How do different combinations of state and change rewards perform? Is Eq. (2) better than other combinations?
Discussion: Figures 18 shows that Eq. (2) reward indeed performs best, achieving the highest success rate (blue, 33.64%). Furthermore, it also generalizes to the most environments, as it is the only showing good transfer to ObstructedMazes, especially when pre-trained on MultiEnv. Considering both the sum and the product of the parts (violet, 31.9%) comes in a close second, but does not transfer to ObstructedMazes when pre-trained on MultiEnv. Other reward combinations perform comparably (red, gold, green, 30.4%). Nonetheless, they all perform better than 'state-only count' (purple, 28.35%, Figure 17). Interestingly, 'change-only count' perform poorly (brown, 18.15%). We will come back to this in Appendix B.4.

$\frac{1}{\sqrt{N(s')N(c)}}$	$\frac{0.5}{\sqrt{N(s')}} + \frac{0.5}{\sqrt{N(s')}}$	$\left(\frac{0.5}{\sqrt{N(s')}} + \frac{0.5}{\sqrt{N(s')}}\right)^2$	$\frac{1}{\sqrt{N(c)+N(s')}}$	$\frac{1}{N(c)+N(s')}$	$\frac{1}{\sqrt{N(c)}}$
30.43%	30.40%	31.92%	30.45%	**33.64%**	18.15%

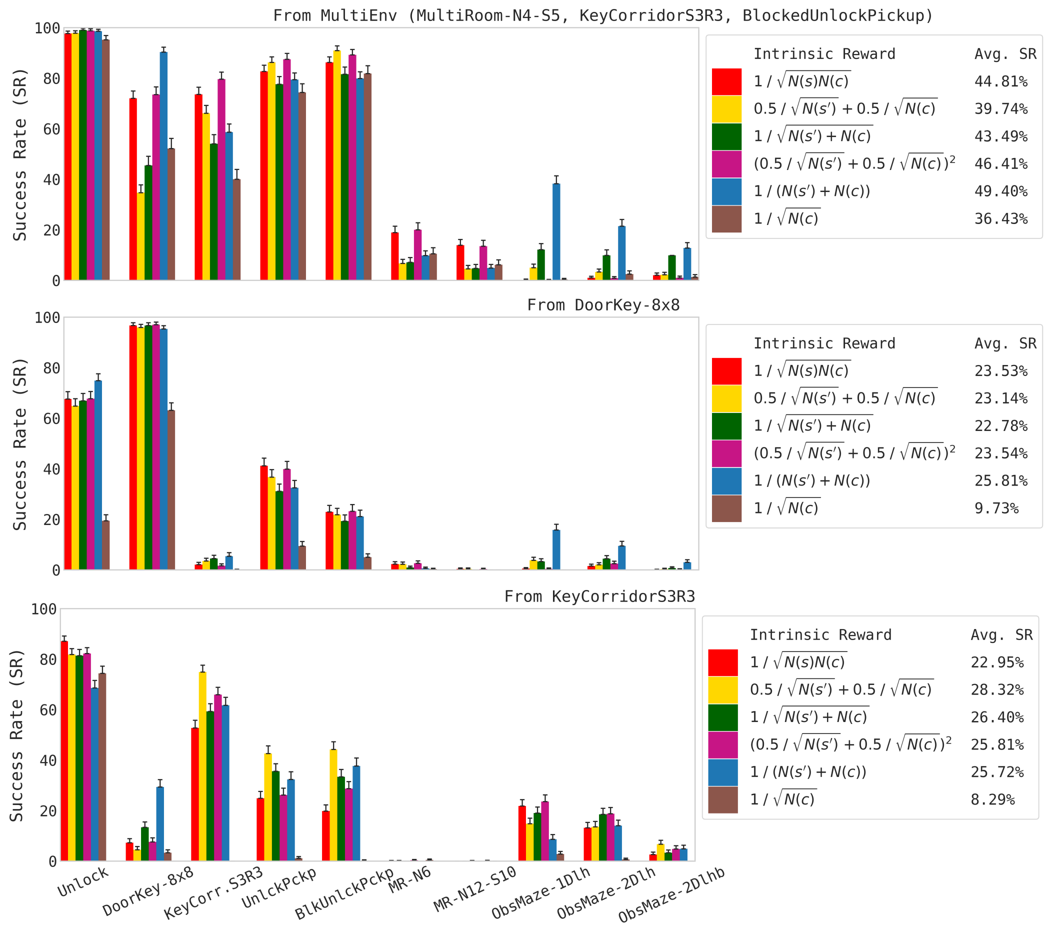


Figure 18: Success rate of policies pre-trained on different combinations of state and change count rewards, with panoramic changes and random resets. States are always counted with egocentric views. C-BET (blue) outperforms other rewards, achieving the highest overall success rate and being the only transferring well to ObstructedMazes.

B.3 C-BET: Egocentric Views

Question: How does C-BET perform when environment changes are encoded with egocentric views?
Discussion: With egocentric views, the performance of all rewards decreases. This is not surprising, because panoramic views better encode environment changes, being rotation-invariant w.r.t. the orientation of the agent. The key to better exploration, thus, is the combination of agent-centric encoding of the state (egocentric views) and environment-centric encoding of the change (panoramic views). This conclusion is also strengthened by the following fact: neither 'state-only count' (Figure 17, purple and yellow) nor 'change-only count' (brown) achieve the same success rate and transfer generalization as C-BET (blue). *C-BET does not perform better because of panoramic change counts, but because of how it combines them with egocentric state counts.*

$\frac{1}{\sqrt{N(s')N(c)}}$	$\frac{0.5}{\sqrt{N(s')}} + \frac{0.5}{\sqrt{N(s')}}$	$\left(\frac{0.5}{\sqrt{N(s')}} + \frac{0.5}{\sqrt{N(s')}}\right)^2$	$\frac{1}{\sqrt{N(c)+N(s')}}$	$\frac{1}{N(c)+N(s')}$	$\frac{1}{\sqrt{N(c)}}$
27.19%	19.61%	**26.95%**	17.33%	21.69%	6.64%

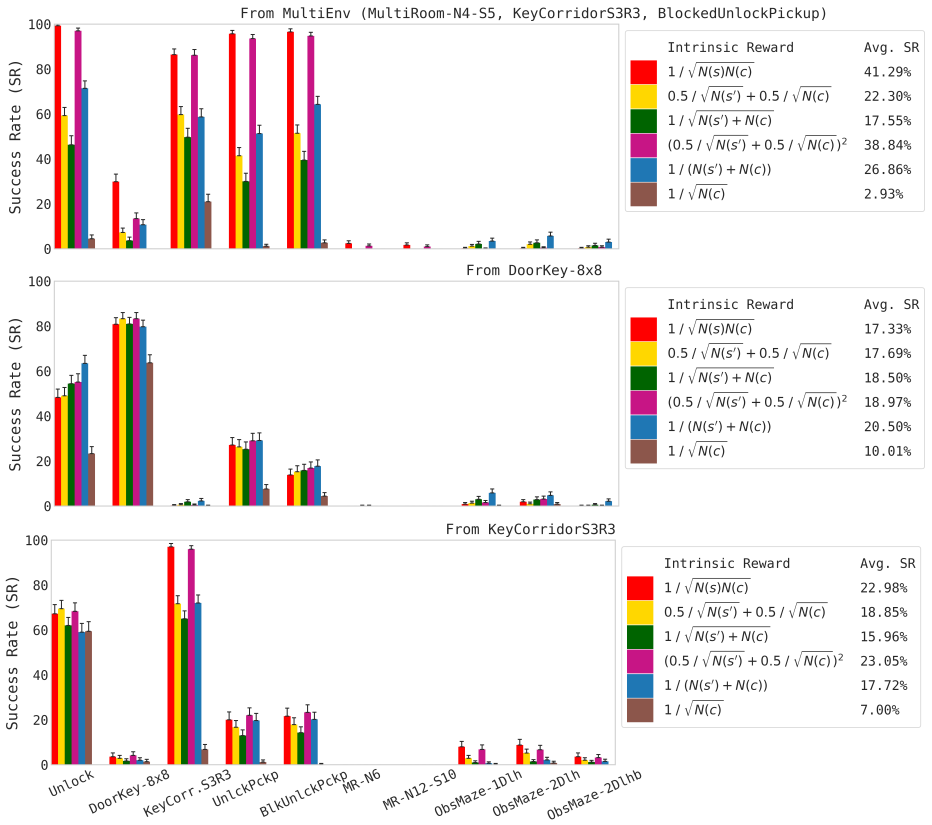


Figure 19: Compared to Figure 18, egocentric changes decrease the performance of all rewards, especially the one based only on change counts (brown). Despite its lower success rate, Eq. (2) still transfer well to many environments, including ObstructedMazes when pre-trained on MultiEnv.

B.4 C-BET: Why Do 'Change-Only Counts' Perform Poorly?

Question: In the previous sections, 'change-only count' (brown) performed poorly compared to other rewards. However, 'state-count only' (purple) does not. Why is that so?
Discussion: First, 'panoramic change-only counts' performs poorly only in SingleEnvs and not in MultiEnv. This is similar to what we have seen in Figure 17 with 'state-only counts': in SingleEnvs panoramic change counts for turning left/right increase too fast and rewards decay too rapidly, while in MultiEnv the diversity of states prevents that. Thus, the resulting policy will learn not to turn.

Then why does 'change-only count' perform even worse with egocentric changes, especially in MultiEnv? In this case, the problem is the opposite. Turning left/right is the only action **always** resulting in some egocentric change, while other actions (including moving forward) can fail and thus produce no change. Furthermore, in visually rich environmnents, such as KeyCorridor, the egocentric change is likely to be unique. Therefore, the policy will overfit to turning left/right.
This is confirmed by Figure 20, showing the policy distribution at the end of pre-training. When trained with panoramic changes (second plot), 'change-only counts' overfits to moving forward. When trained with egocentric views (third plot), the policy is drastically different and prefers to turn most of the time. On the contrary, C-BET policy (first plot) moves less and interacts more.
Interestingly, C-BET also does nothing ('done') more than other policies, but this can be explained by recalling what we discussed in Section 3.2: the optimal exploration policy should keep some randomness to visit the environment uniformly. In practice, C-BET does not have to always interact with the environment if the resulting change is not diverse. That is, C-BET intrinsic reward tries to keep uniform state/changes counts over **all** possible state/changes, including 'no change' counts.
Finally, Figure 20 also shows how the action distribution changes depending on the environment. In larger environments (DoorKey, BlockedUnblockPickup, MultiRooms and ObstructedMazes), the agent moves more. In environments where doors are already unlocked (KeyCorridor and MultiRooms) the agent toggles more and picks/drops fewer times.

	Left	Right	Forward	Pick	Drop	Toggle	Done	Entropy
C-BET	10.7%	11.1%	51.5%	7.5%	7.0%	9.3%	2.6%	0.78
Pano Change-Only	11.5%	9.8%	58.4%	5.5%	5.1%	9.5%	0.05%	0.68
Ego Change-Only	34.9%	35.0%	24.7%	2.0%	1.9%	1.2%	0.1%	0.66

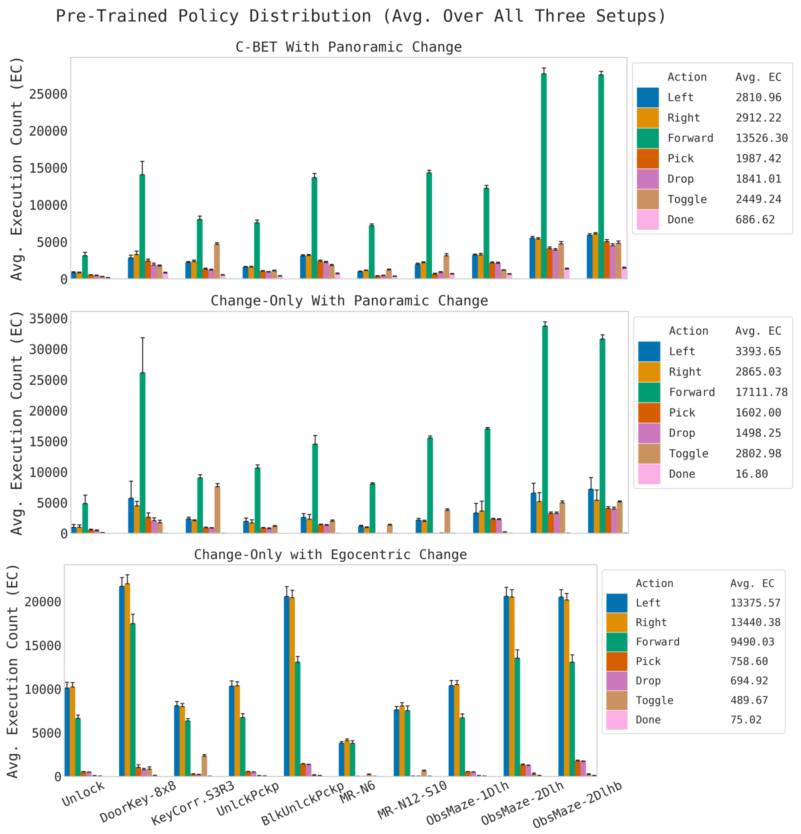


Figure 20: Plots show how many times pre-trained policies executed an action during 100 episodes when transferred to new environments. Counts are averages over both SingleEnvs and MultiEnv transfers. The table reports the resulting action probability. Last column shows the normalized entropy of the distribution (a random distribution has entropy 1). 'Panoramic change-only' overfits to moving forward, while 'egocentric change-only' to turns. On the contrary, C-BET moves less, interacts more, and overfit less to any action as shown by its higher entropy.

B.5 Counts: No Resets vs Random Resets

Questions: Are count random resets necessary for intrinsic-only learning?
Discussion: Figure 22 and its table show that not resetting counts achieves lower success rate. In particular, transfer is significantly worse when the agent is pre-trained on DoorKey. The reason for the worse performance –especially in DoorKey– is shown in Figure 21. As the agent explores, intrinsic rewards decay due to counts growth. The decay is more prominent in DoorKey because of its sparse changes and similar states, but it is noticeable also in KeyCorridor and MultiEnv. Interestingly, 'state-count only' (purple) achieves a better success rate without resets when pre-trained in MultiEnv. However, Figure 22 also shows that the policy overfits to training environments, not showing any transfer to MultiRooms and ObstructedMazes. On the contrary, despite the overall success rate, C-BET (blue) still transfers well to all environments from MultiEnv.

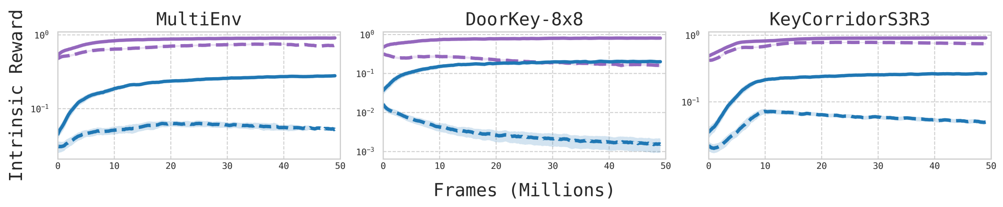


Figure 21: Log scale intrinsic rewards trend at pre-training. Without resets, rewards decay preventing further learning. This is prominent in DoorKey, where states are similar and changes sparse.

$\frac{1}{\sqrt{N(s')}}$, Resets	$\frac{1}{N(s')+N(c)}$, Resets	$\frac{1}{\sqrt{N(s')}}$, No Resets	$\frac{1}{N(s')+N(c)}$, No Resets
28.35%	**33.67%**	27.21%	18.73%

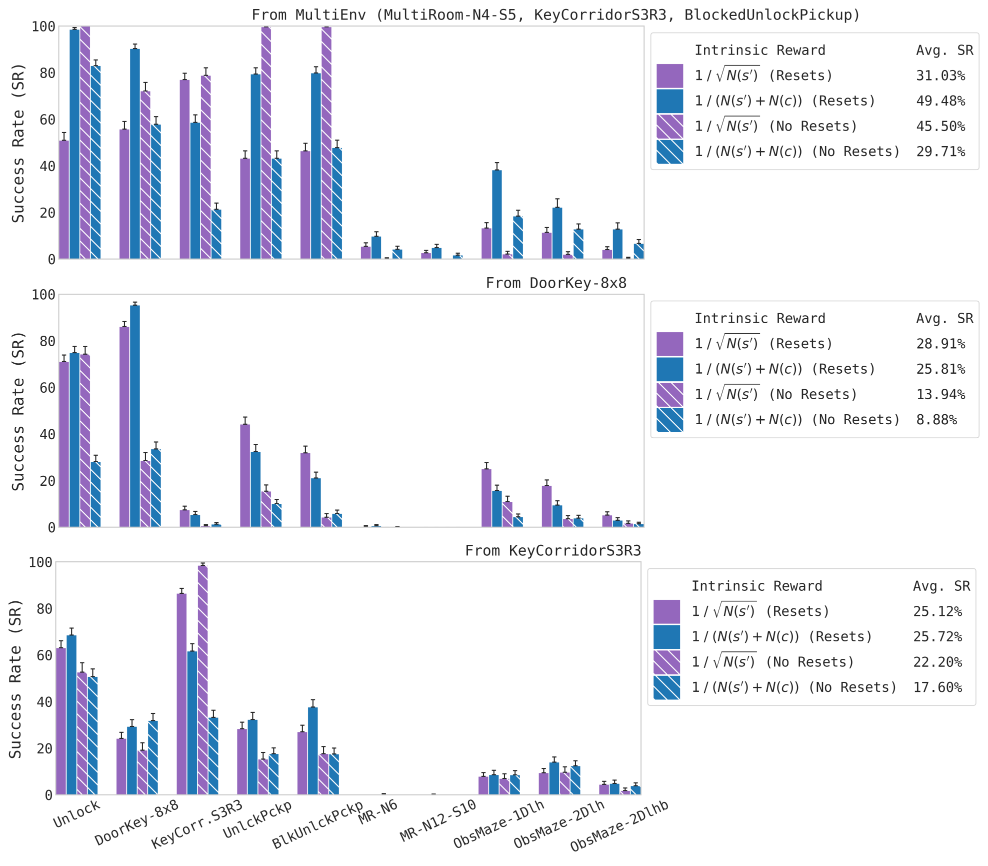


Figure 22: Due to intrinsic rewards decay, policies cannot be properly trained, and do not transfer well. Yet, C-BET still transfers well to all environments when pre-trained in MultiEnv.

B.6 Conclusion of the Study

The ablation study above has provided the answers to the fundamental questions we asked at the beginning.

1. In Appendix B.1 and B.4, state-only and change-only counts with panoramic views did not perform well, thus *panoramic views, alone, are not sufficient for learning good exploration policies*.
2. In Appendix B.2 and B.3, combining egocentric views for counting states and panoramic views for counting changes performs best, thus *the most important contribution of C-BET is the combination of agent-centric (egocentric states) and environment-centric (panoramic changes) exploration*. The design of the reward is also important, albeit to a lesser extent.
3. In Appendix B.5, resetting counts prevents the vanishing of intrinsic rewards, thus *random resets are also a crucial contribution of C-BET*.
4. In all experiments, policies transfers better when pre-trained in MultiEnv –especially C-BET– thus *pre-training by exploring multiple environments allows better generalization*. Similarly, transfer is harder when pre-training environments have similar states or sparse changes.

C MiniGrid Pre-Training Supplemental Results

In this paper, we argued that interacting with the environment while looking for rare changes helps finding extrinsic rewards faster. Thus, we evaluated policies on the number of interactions with the environment and on their success rate. Here we further elaborate the results presented in Section 4.1.

C.1 Unique Interactions per Episode at Pre-Training and at Offline Transfer

Figures 23 and 24 show that C-BET intrinsic reward encourages to interact with the environment more than all baselines. The more C-BET explores at pre-training, the more it interacts with it producing **unique** changes. At transfer, C-BET behavior transfers well to new environments, even the ones unknown dynamics (e.g., boxes in ObstructedMazes needs to be toggled to reveal keys). Of the baselines, only Count performs similarly to C-BET, but it does not generalize as well as C-BET. Most of its interactions, indeed, are in ObstructedMazes after training in MultiEnv.

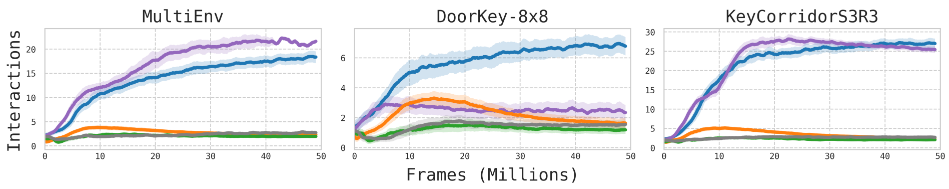


Figure 23: Trend of unique interactions per episode, i.e., picks/drops/toggles resulting in unseen changes. For instance, repeatedly picking and dropping the same key in the same cell results in only two interactions. Only C-BET interacts more with all pre-training environments as it explores.

C-BET	Count	RIDE	Curiosity	RND	Random
26.92	24.72	8.32	7.18	7.34	7.40

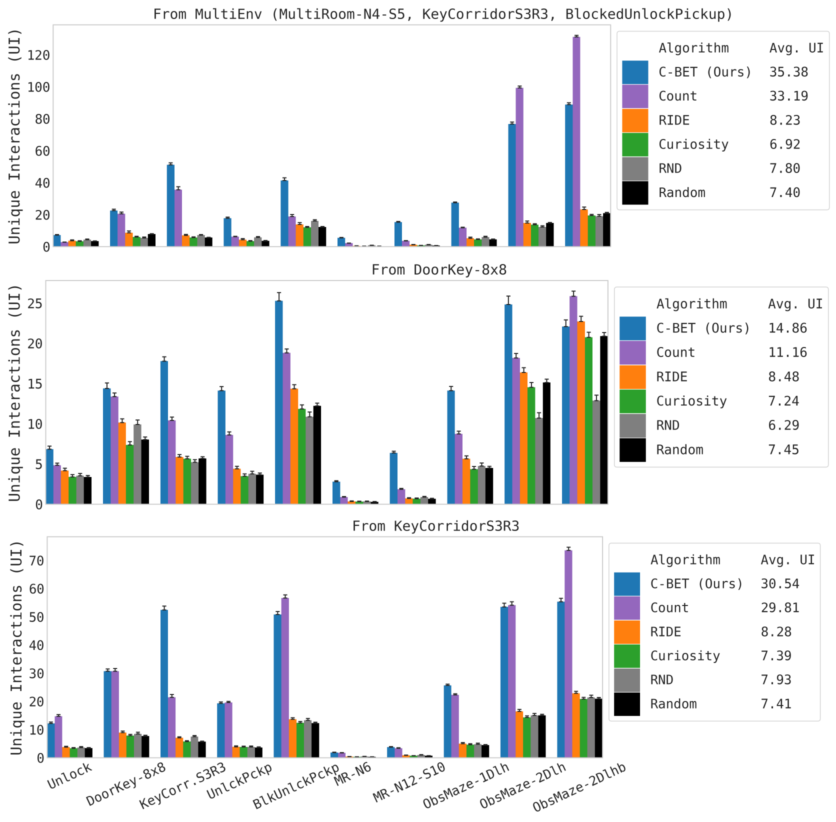


Figure 24: C-BET also outperforms baselines when we compare unique interactions at offline transfer. Not only it interacts the most as shown by the recap table, but also interacts in all environments. Clearly, it interacts more in environment with many keys/balls/boxes to pick (KeyCorridor, BlockedUnblockPickup, ObstructedMazes), and less if there is nothing to pick (MultiRooms).

C.2 Extrinsic Return at Pre-Training and Success at Offline Transfer

Already at pre-training, C-BET finds goal states thanks to its better exploration, as shown by the increasing trend of the extrinsic return[6] (Figure 25). Similarly, at the beginning of transfer –i.e., without further training with extrinsic rewards– C-BET already succeeds in many environments, especially when pre-trainined on MultiEnv (Figure 26). Among the baselines, only Count achieves success in some environments, but it clearly overfits to the pre-training environments and does not generalize nearly as well as C-BET.

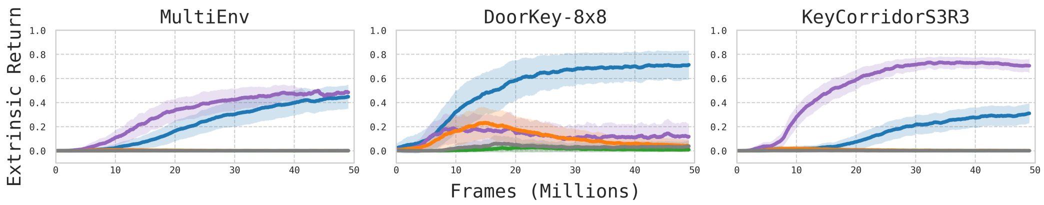


Figure 25: Trend of extrinsic return at pre-training. Only C-BET shows increasing return in all setups, showing that its better exploration brings it often to goal states.

C-BET	Count	RIDE	Curiosity	RND	Random
33.64%	27.21%	1.8%	0.83%	2.52%	0.9%

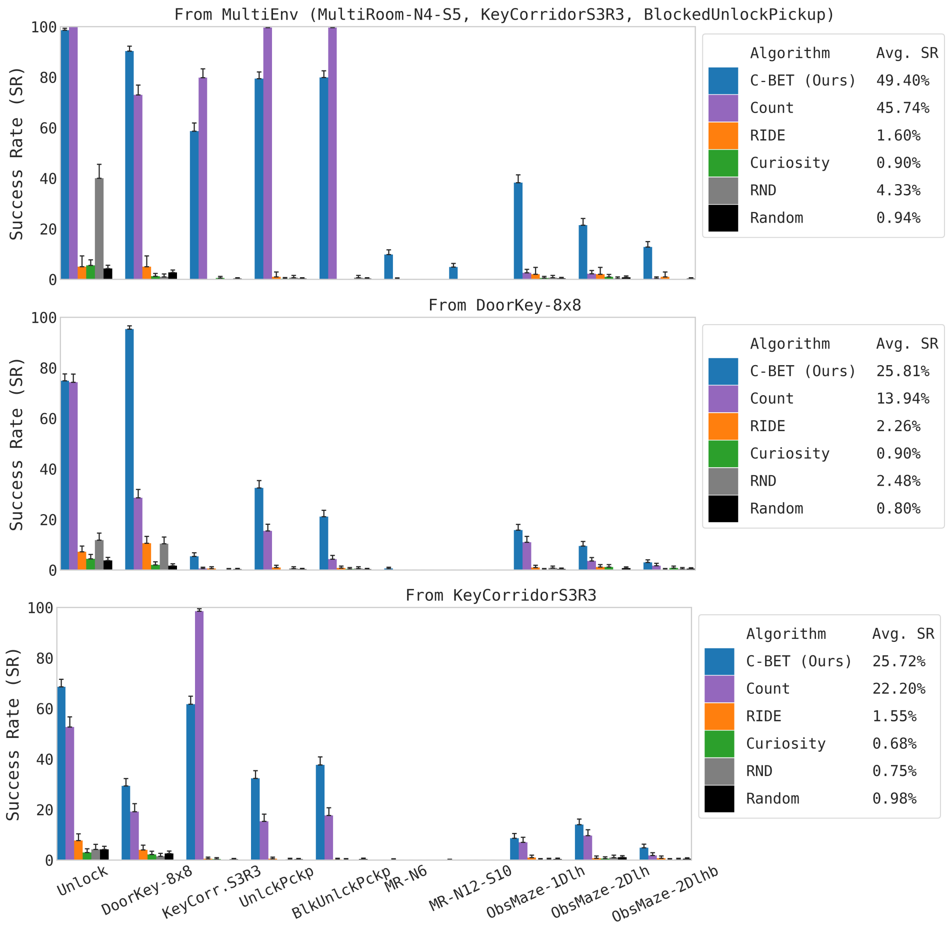


Figure 26: Success rate of pre-trained policies at the beginning of transfer. Not only C-BET achieves the highest rate, but also generalizes to most environments, especially when pre-trained on MultiEnv.

[6]We recall that at pre-training agents do not get extrinsic rewards. We only record them as proxy for success.

C.3 The Problem of Vanishing Intrinsic Rewards at Pre-Training

In the previous two sections, we have shown that C-BET outperforms baselines. Its agent interacts more with the environment while looking for rare and unique changes, and the resulting behavior allows it to discover extrinsic rewards more often. But why do baselines –especially RIDE, Curiosity and RND– perform poorly? As discussed in Section 3.2, classic intrinsic rewards decrease over time as the agent explores, to the point that they vanish to zero given enough samples, preventing further learning. On the contrary, thanks to count resets C-BET reward never decays, and allows the agent to always get meaningful feedback.
This is shown in Figure 27, where intrinsic rewards based on model errors (RIDE, Curiosity, RND) quickly decay as models are learned with ease. Even Count, that does not use any model, suffer from vanishing rewards, albeit to a lesser extent and mostly when trained in environments with similar states like DoorKey. On the contrary, C-BET intrinsic reward increases over time the more agent learns to interact with the environment. But does C-BET outperform Count only thanks to count resets? As we have shown in Appendix B.5, even with resets Count does not generalize well to unseen environments, and overfits to the training environments. C-BET, instead, achieves success even in unseen environments.

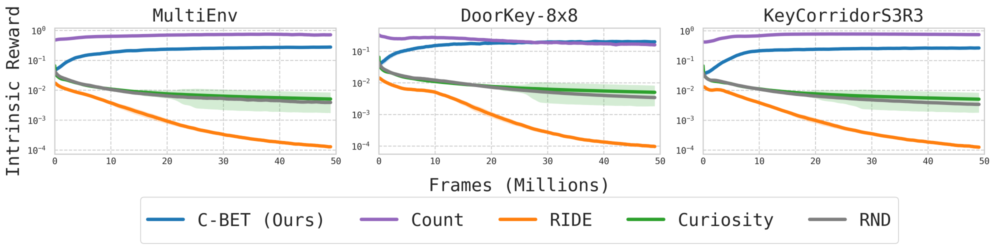


Figure 27: Log-scale trend of intrinsic rewards at pre-training. Baselines rewards decay over time, preventing learning. The problem is prominent in model-based rewards (RIDE, Curiosity, RND). On the contrary, C-BET rewards increase thanks to count resets.

D Noisy Environment Supplemental Results

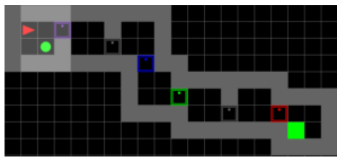


Figure 28: Random instance of the MultiRoomNoisyTV-N7-S4 environment used in our experiments. It has seven rooms of maximum size four.

So far, we conduced experiments in environments with deterministic dynamics. It is known, that stochastic dynamics can be problematic for intrinsic rewards based on model prediction errors [35], as the agents it attracted to sources of noise. This may affect C-BET as well, especially if the noise influences the environment (e.g., the change) rather than agent (e.g., the state). In this section, we test C-BET ability to deal with stochasticity. In particular, we want to see if either its change-based reward or its count resets may negatively affect learning. Therefore, for this evaluation we compare C-BET against the Count baseline –i.e., state-count reward only– with and without count resets on two versions of the same environments (with and without noise)[7]. The environment is MultiRoomNoisyTV, developed by Raileanu and Rocktäschel [36] and depicted in Figure 28. In this environment, a ball is placed in the first room, where the agent spawns at the start of an episode. Anytime the agent does the 'drop' action the ball randomly changes color, regardless of the agent position. Like any MultiRoom, there is nothing that can be picked, and the ball itself cannot be picked.

Table 1 shows that both C-BET and Count perform worse without resets. In particular, Table 2 shows that their learned policies are essentially random. This further confirms what we showed in Appendix B.5: without resets, intrinsic rewards vanish and any policy –even a random one– is optimal. This problem is prominent in MultiRooms due to their scarce state diversity, as there are only empty rooms with already unlocked doors.

[7]We also tested RIDE, RND and Curiosity, but they showed the same poor performance of previous experiments. Despite the noise, their model error still decays quickly.

Table 1: **Policy success rate** averaged over all ten transfer environments in different pre-training setups. In both cases, not resetting counts perform worse. However, C-BET is negatively affected by the NoisyTV, albeit to a limited extend, while Count benefits from it.

	C-BET (Resets)	C-BET (No Resets)	Count (Resets)	Count (No Resets)
Default	11.37%	1.68%	9.72%	5.92%
NoisyTV	9.76%	4.0%	**17.90%**	4.04%

Unsurprisingly, when the noisy ball is added C-BET performance slightly decreases. However, Count's almost doubles. To explain these behaviors, we need to look at the policy distributions after pre-training, shown in Table 1. Let's consider distributions after training without noise first. In this case, both policies assign high probability to 'toggle'. 'Toggle', indeed, is the only interesting action to do because there is nothing to pick and all doors are already unlocked (and 'toggle' opens/closes doors). The only significant difference is that C-BET moves more and turns less. The reason is that C-BET's panoramic changes give little reward to turns, as discussed in Appendix B.1 and B.4. Overall, when trained without noise C-BET transfer slightly better than Count, as shown in Table 1.

Let's now consider distributions after training with noise. As expected, both policies assign much higher probability to 'drop', the action randomly changing the ball color. Yet, other actions probabilities are significantly different between C-BET and Count.
Indeed, C-BET assigns slightly lower probability to all other actions, but overall its distribution does not change much. This explains why its success rate in Table 1 is similar with/without noise.
On the contrary, Count's 'toggle' probability drastically decreases (from 24.7% to 6.8%) but 'forward' increases. Therefore, Count's policy toggles less and moves more. Thanks to this, the resulting policy explores better, despite the higher 'drop' probability.

Finally, we can summarize the findings of this investigation as follows.

- Count resets are important for learning exploration policies, especially if the training environment lacks diversity.
- A policy 'toggling too much' performs poorly.
- In the presence of noise, C-BET performs worse, albeit to a limited extent.
- The presence of noise actually helps the Count baseline over 'overfit less' to some actions (in this case, the policy learns not to 'toggle too much').

Table 2: **Policy distributions** after pre-training in different setups. Without resets (gold rows) both C-BET and Count policies are essentially random regardless of the noise, as shown by their entropy. With resets and no noise, both C-BET and Count assign low probability to 'drop' and high to 'toggle' (highlighted in red), as there are only unlocked doors. With resets and noise, both assign much higher probability to 'drop', the action triggering the noisy ball to change color.

From Default MultiRoom

	Left	Right	Forward	Pick	**Drop**	**Toggle**	Done	Entropy
C-BET (Resets)	4.0%	4.3%	48.6%	5.3%	**6.0%**	**26.1**%	5.6%	0.75
C-BET (No Resets)	13.6%	13.6%	16.0%	14.1%	14.1%	14.4%	14.0%	0.99
Count (Resets)	12.4%	9.2%	33.2%	5.2%	**5.6%**	**29.0**%	5.0%	0.86
Count (No Resets)	14.3%	14.8%	23.7%	12.7%	10.6%	12.6%	11.1%	0.98

From MultiRoom with NoisyTV

	Left	Right	Forward	Pick	**Drop**	**Toggle**	Done	Entropy
C-BET (Resets)	2.9%	3.0%	48.4%	5.3%	**12.8%**	**24.7**%	2.9%	0.74
C-BET (No Resets)	11.5%	11.3%	20.3%	13.0%	15.8%	15.7%	12.3%	0.99
Count (Resets)	10.7%	11.6%	42.3%	4.6%	**22.2%**	**6.8**%	2.1%	0.82
Count (No Resets)	10.9%	12.1%	16.5%	9.3%	32.1%	11.3%	7.6%	0.94

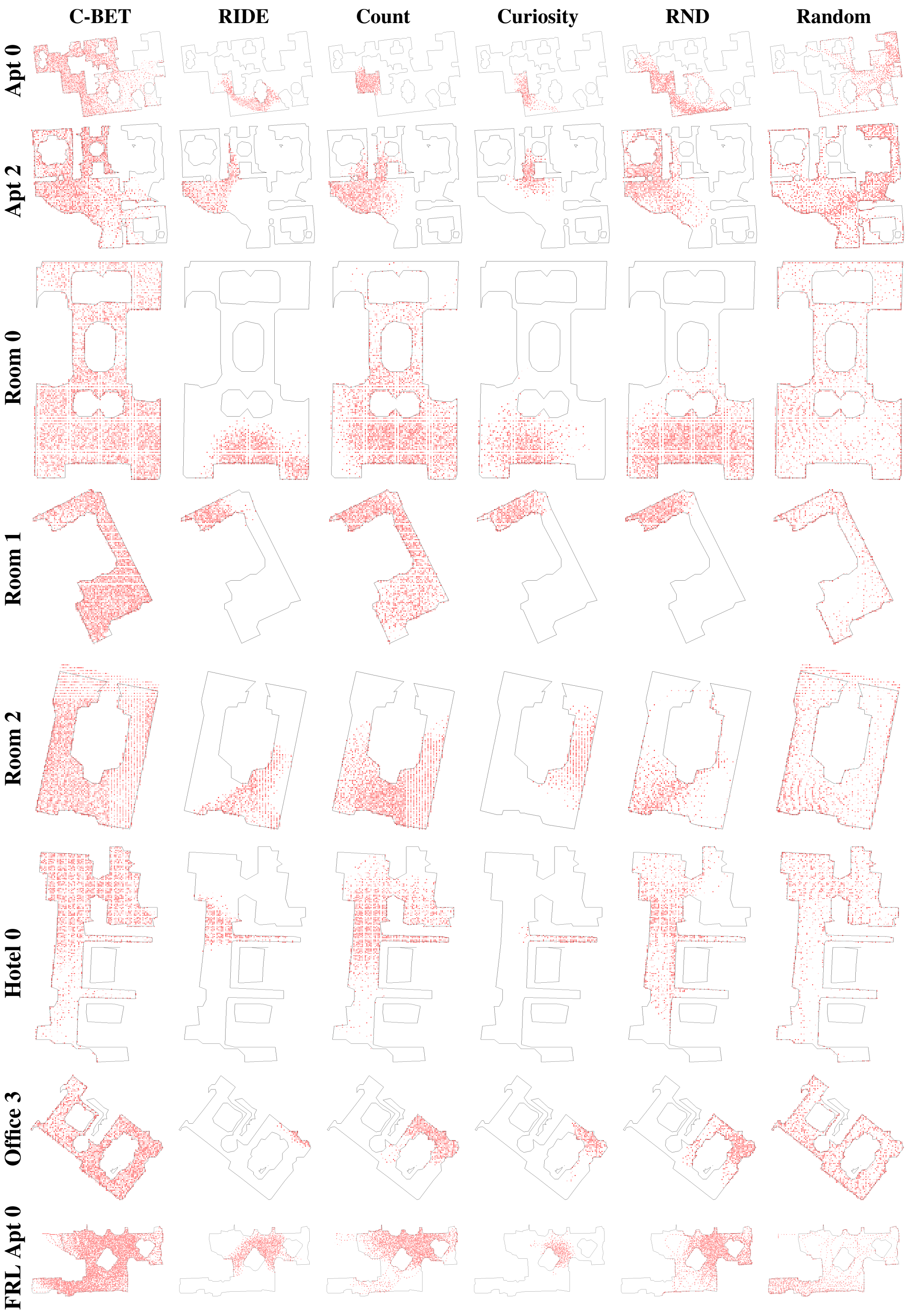


Figure 29: **States discovered by exploration policies during one episode (500 steps) at offline transfer to new scenes.** Darker red cells denote higher visitation rates. C-BET outperforms baselines and exhibits great transfer to all scenes. It is the only algorithm visiting smaller scenes completely (Room 0, Room 1, Room 2, Office 3, FRL Apt. 0), and most of the bigger ones (Apt. 0, Apt. 2, Hotel 0) within only one episode. Furthermore, its visitation count is uniform in all scenes.