EFFECTIVE EXPLORATION VIA INTRINSIC MOTIVATION IN REINFORCEMENT LEARNING

BERKAY EREN

Thesis for the M.Sc. Program in Computer Engineering

Graduate School

İzmir University of Economics

İzmir

2026

EFFECTIVE EXPLORATION VIA INTRINSIC MOTIVATION IN REINFORCEMENT LEARNING

BERKAY EREN

THESIS ADVISOR: ASSIST. PROF. DR. ALPER DEMIR

Master's Exam Jury Members

Assoc. Prof. Dr. Kaya OĞUZ

Assoc. Prof. Dr. Selma TEKİR

Assist. Prof. Dr. Alper DEMİR

A M.Sc. Thesis

Submitted to

the Graduate School of İzmir University of Economics

the Department of Computer Engineering

İzmir

2026

Approval of the Graduate School

Prof. Dr. Muhittin Hakan Demir

I certify that this thesis satisfies all the requirements as a thesis for a M.Sc. degree.

Prof. Dr. Yusuf Murat Erten

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for a M.Sc. degree.

Assist. Prof. Dr. Alper Demir

M.Sc. Exam Jury Members

Assoc. Prof. Dr. Kaya Oğuz
Dept. of Computer Engineering, İUE

Assoc. Prof. Dr. Selma Tekir
Dept. of Computer Engineering, İZTECH

Assist. Prof. Dr. Alper Demir
Dept. of Computer Engineering, İUE

ETHICAL DECLARATION

I hereby declare that I am the sole author of this thesis and that I have conducted my work in accordance with academic rules and ethical behaviour at every stage from the planning of the thesis to its defence. I confirm that I have cited all ideas, information and findings that are not specific to my study, as required by the code of ethical behaviour, and that all statements not cited are my own.

Name, Surname: Berkay EREN

Date: 07.01.2026

Signature:

ABSTRACT

EFFECTIVE EXPLORATION VIA INTRINSIC MOTIVATION IN REINFORCEMENT LEARNING

Eren, Berkay

M.Sc. Program in Computer Engineering

Advisor: Assist. Prof. Dr. Alper Demir

December, 2025

Reinforcement learning agents often struggle in sparse-reward environments where feedback is limited and appears only after a sequence of correct actions. In partial-observable navigation tasks, simple exploration strategies are often insufficient. This thesis investigates intrinsic motivation mechanisms, specifically focusing on the "Don't Do What Doesn't Matter" (DoWhaM) method, which rewards rare but effective actions. To address its limitations in spatial tasks, we propose Area-aware DoWhaM Adaptation (ADA). This method extends action-usefulness with spatial novelty bonuses to encourage expanding the visible area. We evaluate ADA against DoWhaM and a Count-Based baselines in various MiniGrid environments. Results indicate that ADA improves sample efficiency in the early stages of training. In dynamic environments where the layout changes in every episode, ADA significantly outperforms the Count-Based baseline and learns faster than DoWhaM. These findings suggest that combining action-usefulness with spatial novelty provides a robust heuristic for exploration in procedurally generated tasks.

Keywords: Reinforcement Learning, Intrinsic Motivation, DoWhaM, Sparse Rewards, Sample Efficiency.

ÖZET

PEKİŞTİRMELİ ÖĞRENMEDE İÇSEL MOTİVASYON YOLUYLA ETKİLİ KEŞİF

Eren, Berkay

Bilgisayar Mühendisliği Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi. Alper Demir

Aralık, 2025

Pekiştirmeli öğrenme etmenleri, geri bildirimin sınırlı olduğu ve yalnızca bir dizi doğru eylemden sonra ortaya çıktığı seyrek ödüllü ortamlarda sıklıkla güçlük çekmektedir. Kısmi gözlemlenebilir navigasyon problemlerinde, basit keşif stratejileri genellikle yetersiz kalmaktadır. Bu tez, özellikle nadir ancak etkili eylemleri ödüllendiren "Önemsiz Olanı Yapma" (DoWhaM) yöntemine odaklanarak içsel motivasyon mekanizmalarını incelemektedir. Bu yöntemin uzamsal görevlerdeki sınırlamalarını ele almak amacıyla, Alan-duyarlı DoWhaM Adaptasyonu (ADA) önerilmektedir. Bu yöntem, görülebilir alanı genişletmeyi teşvik etmek için eylem-yararlılığını uzamsal yenilik bonusları ile genişletir. ADA, çeşitli MiniGrid ortamlarında DoWhaM ve Sayaç-Tabanlı (Count-Based) bir temel yöntem ile karşılaştırmalı olarak değerlendirilmiştir. Sonuçlar, ADA'nın eğitimin erken aşamalarında örneklem verimliliğini artırdığını göstermektedir. Düzenin her bölümde değiştiği dinamik ortamlarda ADA, Sayaç-Tabanlı yöntemi önemli ölçüde geride bırakmakta ve DoWhaM'dan daha hızlı öğrenmektedir. Bu bulgular, eylem-yararlılığının uzamsal yenilik ile birleştirilmesinin, prosedürel olarak oluşturulmuş görevlerde keşif için sağlam bir sezgisel yaklaşım sağladığını ortaya koymaktadır.

Anahtar Kelimeler: Pekiştirmeli Öğrenme, İçsel Motivasyon, DoWhaM, Seyrek Ödüller, Örneklem Verimliliği.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Alper Demir, for his continuous support, guidance, and encouragement throughout this thesis study. His feedback and academic direction were essential in shaping the research questions, the experimental design, and the overall structure of the work.

Also I dedicate this thesis to my wife, who shared every burden of this difficult year with patience, strength, and unwavering support. I also dedicate it to our beloved daughter, Ada. Although we never had the chance to meet her, she will always be part of our family, and her memory and love will remain with us.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 : INTRODUCTION

1.1 *Motivation*

Reinforcement learning (RL) provides a general framework where an agent learns to make decisions by interacting with an environment. While interacting with the environment, agent receives a feedback in the form of rewards (Sutton and Barto, 1998). In many practical problems, these feedbacks are not immediately collected or dense enough to define a clear policy π. Instead, rewards are *sparse* and appear only after a long sequence of actions. Navigation tasks, puzzle-like environments or environments that require collecting items in a specific order before reaching the goal are examples of tasks with *sparse* rewards. In such environments, simple exploration techniques are often not sufficient, and the agent may spend a large amount of time wandering around without making a progress.

Grid-based environments with goal-oriented tasks, such as MiniGrid (Chevalier-Boisvert et al., 2023), are frequently used as a platform for such hard-exploration problems. Within each step, agent receives a small local window so called a *partial-observation* of the environment. Then using this limited information, agent must discover doors, keys or other important objects before it can finally reach the goal. Even in these relatively simple visual domains, finding a good exploration strategy is non-trivial. ε-greedy exploration is a widely used baseline , where agent selects a random action with probability ε and acts greedily otherwise (Sutton and Barto, 1998). In principle, this kind of random exploration can eventually find a rewarding behavior, but in practice, it usually requires a very large number of episodes to train a sufficient policy.

To improve exploration, modern deep RL research makes heavy use of *intrinsic motivation* signals (Ladosz et al., 2022). For hard-exploration problems, relying only on the external reward from the environment is not enough. To solve such environments, an agent needs to be motivated with internal bonuses which we call *intrinsic rewards*. These intrinsic rewards encourage agent to visit novel states. Popular approaches include prediction-error based bonuses such as Random Network Distillation (RND) (Burda et al., 2018) and Count-Based exploration where the agent gets higher rewards for rarely visited state-action pairs. These methods have shown

promising results in different sparse-reward setups, but they also come with trade-offs in terms of stability, implementation complexity and environment dependence.

Don't Do What Doesn't Matter (DoWhaM) is a more recent intrinsic motivation method which focuses on *action-usefulness* (Seurin et al., 2021). Instead of purely focusing on a state novelty, DoWhaM tries to identify and reward the actions that rarely but effectively change the state of the environment. DoWhaM has shown strong results where there are many possible actions and only a small subset of them are actually useful.

In some environments, the agent must explore large areas, open several doors in sequence or collect items in different rooms before reaching the final goal. In such environments, relying only on a pure action-usefulness strategy is not always enough in tasks where *spatial* coverage is crucial. In these environments, agent should not only try to discover rare but effective actions, but it also needs to be encouraged to widen its visible area and step into parts of the map it has not visited before.This creates the need for intrinsic reward mechanism that bring together action-usefulness and spatial novelty.

This thesis focuses on intrinsic motivation for exploration in navigational environments where agent location is included in the state definition. We first will adapt the original DoWhaM mechanism. Then on top of the original method, we will introduce an extended version with spatial novelty bonus as we call it **A***rea-aware* **D***oWhaM* **A***daptation* (**ADA**). Both variants are evaluated against a Count-Based exploration baseline in several environments with different levels of difficulty. The main goal is to see in which kinds of tasks these *spatial bonuses* in ADA give a clear benefit compared to the original formulation and the simpler Count-Based method.

1.2 *Research Questions*

The work in this thesis is guided by the following research questions:

1. To what extent does the ADA mechanism improve sample-efficiency compared to the original DoWhaM and Count-Based baselines across different sparse-reward environments? In this research, we aim to investigate if the addition of spatial components helps the agent learn faster and reduce episode lengths during the training process.

2. How does the performance of ADA, original DoWhaM, and Count-Based exploration compare when environment layouts and object positions are randomized in every episode? This study also investigates if ADA provides a more stable exploration strategy in environments where the agent cannot rely on memorizing fixed positions.

By answering these questions, we aim to obtain a clearer picture of the strengths and weaknesses of action-usefulness with spatial bonuses, and to see if the additional bonuses can help in complex environments.

1.3 *Contributions*

The main contributions of this thesis can be summarised as follows:

- We implement the original DoWhaM intrinsic motivation mechanism with PPO and experiment in partial observable MiniGrid (Chevalier-Boisvert et al., 2023) environments.
- We propose ADA by enhancing the original action-usefulness of DoWhaM with a spatial novelty components. These components includes an *expansion* bonus for revealing new parts of the map and *achievement* bonus for visiting the previously unseen positions in the environment. Both bonuses are implemented to work together with original action bonus and normalised by episodic state visitation counts.
- We conduct a empirical evaluation of DoWhaM, ADA and a Count-Based exploration baseline across several MiniGrid environments. The experiments are run with multiple random seeds, and we use bootstrap analysis to compare sample-efficiency.
- We provide a qualitative analysis of the learned behaviours by examining episode length distributions. This helps to illustrate how the different intrinsic rewards influence exploration patterns.

1.4 *Thesis Outline*

The rest of this thesis is organised as follows. Chapter 2 introduces the necessary background on Markov Decision Processes (MDP), Reinforcement Learning (RL), policy gradient methods and Proximal Policy Optimization (PPO). It also reviews exploration methods in deep reinforcement learning, with a focus on intrinsic motivation, Count-Based exploration and the original DoWhaM method.

Chapter 3 presents the intrinsic motivation mechanisms proposed in this work. We first describe the DoWhaM formulation in section 2.7.2 and then introduce our method ADA. We will discuss the action bonus, the spatial bonuses and the final intrinsic reward function together with implementation details.

Chapter 4 describes the experimental setup. We introduce the MiniGrid environments used in the experiments, including empty, crossing, four-room maps, multi-room layouts and multi-room with key variants. We also explain the observation and action spaces, the PPO agent configuration and the hyper-parameters used in training.

Chapter 5 reports the empirical results. We compare DoWhaM, ADA and the Count-Based baseline in different environments using episode lengths, reward and percentage visited metrics. We also discuss behavioral differences and episode statistics.

Finally, Chapter 6 provides a general discussion of the findings, outlines limitations of the current work and suggests directions for future research.

CHAPTER 2 : BACKGROUND AND RELATED WORK

2.1 *Markov Decision Processes*

A Markov Decision Process (MDP) is a formal framework used in reinforcement learning (RL) to model sequential decision-making problems where an agent interacts with an environment under uncertainty (Sutton and Barto, 1998). At each time step, the agent observes the current state s, chooses an action a, and then receives a reward r and a new state s' from the environment. The future of the system depends only on the current state and action, which is known as the Markov property.

Formally, an MDP is usually described by the tuple (S, A, P, R, γ), where:

- S is the *state space*, the set of all possible states in which the agent can be,
- A is the *action space*, the set of all possible actions the agent can take,
- $P(s' \mid s, a)$ is the *transition function*, which defines the probability of moving to next state $s' \in S$ when the agent is in state $s \in S$ and takes action $a \in A$,
- $R(s, a)$ is the *reward function*, which assigns a scalar reward $r \in \mathbb{R}$ to each state–action pair (s, a),
- $\gamma \in [0, 1)$ is the *discount factor*, which controls how much future rewards are taken into account compared to immediate rewards.

The agent's behavior is described by a policy π. This policy can be deterministic or stochastic. In the stochastic case, the policy is a distribution $\pi(a \mid s)$ over actions given a state s. The main objective in RL is to learn a policy that maximizes the expected discounted return $\sum_{t=0}^{\infty} \gamma \cdot r_t$, i.e. the expected cumulative reward that the agent collects over time (Sutton and Barto, 1998). When the transition function P is unknown, the agent must estimate the value of states or state–action pairs directly from experience.

To achieve this, many RL methods rely on value functions such as the action-value function $Q^{\pi}(s, a)$. The policy decides how good it is to be in a state, or to take an action in a state. If P and R were known, these value functions could be computed by solving the Bellman equations (Bellman, 1957). However in practice, agent often does not know the dynamics of the environment and it must extract these data with incrementally sampling the transitions.

2.2 *Partially Observable Markov Decision Process (POMDP)*

In many scenarios, the agent does not have access to the full state information. To address this, Partially Observable Markov Decision Process (POMDP) framework is used. Following the standard formalism by Kaelbling et al. (1998), POMDP model is defined by the tuple $\langle S, A, P, R, \gamma, \Omega, \mathcal{O} \rangle$ by extending MDP with:

- Ω: The set of possible observations.
- $O : S \times A \times \Omega \to [0,1]$: The observation function defining the conditional probability $P(o \mid s, a)$.

The optimization goal is defined by the discount factor $\gamma \in [0,1)$, where the agent seeks to maximize the expected discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$. Unlike a standard MDP, the agent does not observe state s directly. An agent must rely on the history of observations and actions to resolve state ambiguity.

2.3 *Temporal Difference*

Temporal Difference (TD) learning provides a way to update value estimates. At each time step, new estimates are computed based on previously learned estimates together (Sutton, 1988). At time step t, the agent observes a transition (s_t, a_t, r_t, s_{t+1}) and compares its current prediction $V(s_t)$ to the TD target

$$r_t + \gamma V(s_{t+1}), \tag{2.1}$$

where $\gamma \in [0,1)$ is the discount factor. The difference between this target and the old estimate is called the TD error,

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{2.2}$$

and the value function is updated in the direction of this error:

$$V(s_t) \leftarrow V(s_t) + \alpha\, \delta_t, \tag{2.3}$$

where $\alpha \in [0,1]$ is the learning rate. This enhances the ability of the agent to learn and update its policy while iteracting with the environment. Since agent does not

need to wait until the end of the episode to evaluate the desired actions, it provides a computational efficiency because agent is updating its predictions online. This kind of update is actually important to solve such environments where reaching a goal requires a sequence of correct actions. For example, in door-based navigation tasks, agent might open the first door which is neccessary for reaching a goal but might fail to open second one so that the episode might end with *termination*. In such sparse reward setting, agent might not understand that opening a first door was actually a good decision because the sequence of actions did not led agent to a successful outcome. TD helps to propagate the final reward signal backwards, so that earlier decisions can receive some credit even when the reward appears only at the end. TD-based methods form the basis of several modern RL algorithms that are used throughout this thesis.

2.4 *Reinforcement Learning*

Reinforcement learning (RL) is a branch of machine learning such that the decision maker, called an agent, interacts with and environment and learns how to achieve a goal by taking actions that maximize its cumulative reward over time. Within each step, using the partial (partial-observation) or global (fully-observation) perception, received from the environment, the agent takes an action and receives a scalar reward from the environment that provides a feedback which can be rewards or penalties based on the agent's actions. Over many episodes, the agent tries to learn a policy π that is maximizes the total reward over an episode or sequence of interactions (Sutton, 1988).

There are several approaches to organizing reinforcement learning methods. First distinction is between *model-based* and *model-free* approaches (Kaelbling et al., 1996). In model-based approaches, model is formed that reflects the dynamics of the environment. Then the agent simulates future trajectories and performs planning to choose its actions. Model-free methods do not aim to estimate the transition probabilities. Instead, they learn the policy or value functions by interacting with the environment. In this thesis, we will focus on the model-free methods.

Another common distinction is between *value-based*, *policy-based* methods (Arulkumaran et al., 2017). *Value-based* methods aim to learn a value function that estimates how good it is to taking a certain action a in the given state s. A typical example is Q-learning. Given a state s and action a , Q-learning implements an off-policy algorithm

that learns an action-value function $Q(s,a)$. Agent then seeks to approximate the optimal value function $Q^*(s,a)$ without needing to follow the optimal policy during learning (Watkins, 1989). Once $Q(s,a)$ is learned, the agent can act greedily with respect to these values. On the other hand, *policy-based* methods focuses on learning the policy $\pi_\theta(a \mid s)$ and directly maps the states s to actions s. This method is adjusting the network weights using the policy parameters θ in the direction that increases the expected return.

Actor-critic methods combine these two ideas. They maintain both a policy (the actor) and a value function (the critic). The critic estimates how good the current policy is and learns an approximate value function. The actor (policy) is using this information to adjust its behavior. Then TD error (Eq. 2.2) is generated between observed value and the predicted value and used to updating the actor's policy parameters. Konda and Tsitsiklis analyse the convergence of such actor-critic architectures and they show that they can provide stable and efficient learning under suitable assumptions (Konda and Tsitsiklis, 1999).

In modern deep reinforcement learning, these ideas are combined with function approximation. In small RL problems, it is possible to store the action-value function $Q(s,a)$ in a table for each state-action pair (s,a). On the other hand, for high dimensional spaces like the image inputs, it is not realistic to build a table that contains an entry for every possible pair. To solve such limitation, deep RL methods are using neural networks as a function approximator for value functions and policies. This allows RL methods to handle more complex observations. The possible downside is, this makes training more unstable and sensitive to hyper-parameters. Among these methods, Proximal Policy Optimization (PPO, see section 2.5) (Schulman et al., 2017) has became widely used because its relatively simple implementation and stable performance in many benchmarks. In this thesis, we use PPO as our underlying model-free actor–critic algorithm.

2.5 *Proximal Policy Optimization (PPO)*

Proximal Policy Optimization (PPO) is a model-free actor–critic algorithm that belongs to the family of policy gradient methods (Schulman et al., 2017). It was proposed as a practical alternative to Trust Region Policy Optimization (TRPO)

(Schulman et al., 2015).

TRPO introduces a trust-region style update where the new policy π' is required to stay closed to old policy π under the KL-divergence constraint. By using this *hard constraint*, TRPO ensures that every update is inside a trust region otherwise taking a large step from old policy to new policy can easily break learning. This behavior leads to a stable learning but it is relatively complicated to implement. PPO keeps the same method of limiting how much the policy can change in a single update, but instead of enforcing an explicit KL constraint, it defines a *clipped surrogate objective*. This objective can be treated as a standard loss and it can be optimised with several epochs of stochastic gradient ascent on mini-batches of collected data. This makes PPO easier to implement and tune in practice, while still providing relatively stable policy updates.

At a high level, PPO alternates between two phases. In the first phase, the current policy interacts with the environment and collects trajectories of transitions (s_t, a_t, r_t, s_{t+1}) for a fixed number of time steps. In the second phase, this batch of data is used to update the policy and value function by performing several epochs of stochastic gradient ascent (for the policy) and gradient descent (for the value function) on mini-batches of the collected experience. This structure follows the usual policy-gradient framework, but the key difference is the surrogate objective that PPO optimizes.

Let $\hat{A}_t$ denote an estimate of the advantage function at time step t. Then a standard policy-gradient objective can be written as

$$L_{\mathrm{PG}}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]. \tag{2.4}$$

In practice, performing many gradient steps on this objective using the same batch of data can lead to very large changes in the policy and therefore to unstable learning. TRPO addresses this problem by explicitly constraining the average KL divergence between the old and new policies, which defines a trust region around the current policy (Schulman et al., 2018). PPO instead modifies the objective itself so that large changes in the policy are automatically penalised.

Let

$$\kappa_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t \mid s_t)} \tag{2.5}$$

be the probability ratio between the new and old policies for the sampled action a_t. The clipped PPO objective is defined as

$$L_{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(\kappa_t(\theta) \hat{A}_t,\ \text{clip}\big(\kappa_t(\theta), 1-\varepsilon, 1+\varepsilon\big) \hat{A}_t \right) \right], \tag{2.6}$$

where ε is a small hyper-parameter (for example 0.1 or 0.2), and clip$(\cdot)$ truncates the ratio to remain in the interval $[1-\varepsilon, 1+\varepsilon]$. The term $\kappa_t(\theta)\hat{A}_t$ corresponds to the usual policy-gradient objective: when $\hat{A}_t > 0$, increasing $\kappa_t(\theta)$ above 1 increases the objective and makes the chosen action more likely, and when $\hat{A}_t < 0$ the opposite happens. The clipped term replaces $\kappa_t(\theta)$ by a value that is no smaller than $1-\varepsilon$ and no larger than $1+\varepsilon$. By taking the minimum of these two expressions, the objective discourages updates that would push $\kappa_t(\theta)$ too far away from 1. In other words, PPO allows improvements in the policy while preventing overly aggressive changes that could destroy previously learned behaviour.

In practice, PPO is implemented in an actor–critic architecture. A neural network (or two closely related networks) outputs both the policy $\pi_\theta(a_t \mid s_t)$ and a value estimate $V_\phi(s_t)$ for the current state s_t and action a_t at time step t. The full loss combines the clipped policy objective, a squared-error loss for the value function and an entropy term that encourages exploration:

$$L(\theta, \phi) = \hat{\mathbb{E}}_t \left[L_{\text{CLIP}}(\theta) - c_1 \big(V_\phi(s_t) - V_t^{\text{targ}}\big)^2 + c_2\, \mathcal{E}\big(\pi_\theta(\cdot \mid s_t)\big) \right], \tag{2.7}$$

where c_1 and c_2 are scalar coefficients, V_t^{targ} is a target for the value function, and $\mathcal{E}(\pi_\theta)$ is the entropy of the policy. The entropy bonus keeps the policy sufficiently random during training, which can improve exploration and prevent the policy from getting stuck in local optima.

The advantage estimates $\hat{A}_t$ are typically computed using Generalised Advantage Estimation (GAE) (Schulman et al., 2018). GAE uses a discounted sum of temporal-difference errors, controlled by an additional parameter λ, to provide a compromise between low variance and low bias. In the PPO paper, this combination of clipped objective and GAE is shown to give robust performance across a range of continuous control and Atari benchmarks (Schulman et al., 2017).

In this thesis, PPO is used as the underlying model-free actor–critic algorithm. The

networks approximate both the policy and the value function from observations, and the intrinsic motivation mechanisms (DoWhaM, ADA and Count-Based exploration) are added as extra reward terms on top of the environment reward. The main focus is not to modify the PPO update itself, but to study how these intrinsic rewards interact with PPO and how they influence the exploration behaviour of the agent in the considered environments.

2.6 *Intrinsic Motivation*

Intrinsic motivation (IM) is a concept used in Reinforcement Learning, where agents obtain internal rewards derived from the satisfaction of their own actions, regardless of external objectives. Elements such as curiosity, novelty, and surprise can elicit this intrinsic motivation (Singh et al., 2010). Recent research has demonstrated that intrinsic rewards, alongside external rewards, enhance agent's learning performance (Barto, 2013). Intrinsic rewards driven by perceptual novelty have been shown to promote more effective exploration and learning (Bellemare et al., 2016; Ostrovski et al., 2017), while predefined intrinsic rewards have been found to improve learning efficiency in hierarchical tasks with sparse external feedback (Kulkarni, 2016). In hierarchical RL, intrinsic motivation is also utilized to steer agents toward specific goals (Li et al., 2021). Furthermore, it is applied in multi-agent settings to foster social awareness (Sequeira et al., 2011), enable coordinated exploration (Iqbal and Sha, 2019), shape social influence (Jaques et al., 2019), and support the acquisition of cooperative behaviors (Hong and Lee, 2021).

2.7 *Exploration*

In reinforcement learning, *exploration* refers to the agent's attempt to search for and discover rewarding behaviors. The agent interacts with the environment, observes states and rewards, and must balance exploiting what it already knows with trying out unfamiliar actions to gather new information. This becomes especially critical in *sparse reward* problems, where meaningful feedback is *rare* and may appear only after a long sequence of actions. A simple and widely used exploration strategy is ε-greedy, where the agent selects a random action with probability ε and acts greedily with respect to its current value estimates otherwise (Sutton and Barto, 1998). In

principle, such strategies can eventually discover rewarding behaviour even in sparse-reward settings, but in practice the number of episodes required is often large. Simple reward shaping is another option, but it is difficult to design by hand. Shaping signals that are too strong or misaligned may lead to discouraging behaviours, such as the agent refusing to move or wandering around the goal.

Ladosz et al. provide a structured overview of modern exploration methods in deep reinforcement learning and organise them into several categories (Ladosz et al., 2022). A first group rewards *novel states*, typically via intrinsic motivation signals based on prediction error (Pathak et al., 2017; Burda et al., 2018), Count-Based bonuses (Bellemare et al., 2016; Tang et al., 2017), or memory-based novelty (Savinov et al., 2019; Badia et al., 2020). A second group rewards *diverse behaviours*, where the objective is to discover a wide range of qualitatively different skills rather than to chase external reward directly (Eysenbach et al., 2018; Gregor et al., 2016). Other families include goal-based exploration where agents learn from achieved goals (Andrychowicz et al., 2018), probabilistic methods which act optimistically or reduce uncertainty based on value or model distributions (Osband et al., 2016), imitation-based approaches which use demonstrations to "kick-start" exploration (Hester et al., 2017).

2.7.1 Spatial Exploration

In sparse reward environment, an agent cannot take any rewards until sequence of an action is taken. This causes a problem that agent cannot take a reward to learn from until task is finished. Especially high-dimensional or visual state spaces common in spatial tasks, there needs to be exploration technique beyond simple random exploration. To address this, exploration methods either generalize counts or focus on intrinsic rewards derived directly from spatial change or position. Spatial bonuses generally refer to intrinsic reward mechanisms that specifically encourage an agent to explore new locations, view new areas, or execute actions that result in significant spatial transitions.

2.7.1.1 *Count-based Exploration*

Count-based exploration is a classical intrinsic-motivation technique that encourages an agent to visit novel or infrequently visited states (Bellemare et al., 2016; Ostrovski et al., 2017).

Given a state-action pair (s_t, a_t) at time t, the agent maintains a visitation count $N(s_t, a_t)$ that tracks the number of times this particular pair has been encountered during training episodes. At each transition (s_t, a_t, r_t, s_{t+1}), the count is updated as:

$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1 \tag{2.8}$$

The intrinsic reward is then defined as an inverse function of this count, which encourages the agent to explore rarely visited state-action pairs. Formally, the Count-Based intrinsic reward is given by:

$$r_t^{\text{count}} = \frac{1}{\sqrt{N(s_t, a_t)}} \tag{2.9}$$

This formulation provides higher intrinsic rewards for novel state-action pairs. In practice, this means the intrinsic bonus starts at a very high value when the agent first encounters new states. As training progresses and the agent repeatedly visits these areas, the reward signal decays and eventually converges to a lower value, effectively shifting the agent's focus from exploration to exploitation.

2.7.2 *Don't Do What Doesn't Matter (DoWhaM)*

Don't Do What Doesn't Matter (DoWhaM) (Seurin et al., 2021) is an intrinsic motivation-based exploration technique that encourages reinforcement learning agents to discover *rare but effective* actions. Unlike the traditional intrinsic motivation methods which encourage an agent based on novelty in state visitation, DoWhaM identifies actions that rarely but effectively alter the state. Given history of transitions $\mathcal{H} = (s_h, a_h, s_{h+1})_{h=0}^{H}$, DoWhaM tracks following features to identify these effective actions;

For every action a, track the *usage count* of an action;

$$U^{\mathcal{H}}(a) = \sum_{h=0}^{H} \mathbf{1}_{\{a_h=a\}} \tag{2.10}$$

For every action a_i, track the *usage effectiveness count* of an action if a changes the state s_h;

$$E^{\mathcal{H}}(a) = \sum_{h=0}^{H} \mathbf{1}_{\{a_h=a\}} \times \mathbf{1}_{\{s_h \neq s_{h+1}\}} \tag{2.11}$$

where $\mathbf{1}$ is the *indicator function*.

The intrinsic action bonus for a given action a_t, is proportional to how frequently the action is effectively changed to the state of the environment where η is a hyper-parameter of the decay of the ratio;

$$B(a_t) = \frac{\eta^{\left(1-\frac{E^{\mathcal{H}}(a_t)}{U^{\mathcal{H}}(a_t)}\right)} - 1}{\eta - 1} \tag{2.12}$$

The final intrinsic motivation reward is given only if the state s_t at time t changes and is divided by an episodic state visitation count to prevent repetitive exploitation;

$$r_t^{\text{DoWhaM}}(s_t, a_t, s_{t+1}) = \begin{cases} \dfrac{B(a_t)}{\sqrt{N_\tau(s_{t+1})}}, & \text{if } s_t \neq s_{t+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

The results of DoWhaM are compared by RND (Burda et al., 2018) and Count-Based exploration (subsubsection 2.7.1.1) in a minigrid (Chevalier-Boisvert et al., 2023) environment. DoWhaM performs better showcasing is practical benefits over other methods. However, DoWhaM is not without limitations. Its reliance on action effectiveness may introduce exploration biases especially in environments densely populated with interactive but non-critical elements.

CHAPTER 3 : AREA-AWARE DOWHAM ADAPTATION (ADA)

The original Don't Do What Doesn't Matter (DoWhaM) algorithm is an action-based intrinsic motivation method. It is designed to identify and rewards actions that are *rarely effective* while down-weighting the value of exploring states that have been frequently visited (Seurin et al., 2021). In the original version, statistics are collected globally over actions. For each action in a given observation, the algorithm counts how often it is used and how often it changes the state. The intrinsic bonus is calculated as a function of the ratio between these counts, combined with an episodic state-count penalty to discourage the agent from visiting the same states repeatedly.

In spatial navigation tasks such as *Minigrid* (Chevalier-Boisvert et al., 2023), this global view is often not enough. However, the effectiveness of an action is often strictly tied to the local context. An action that triggers a meaningful state change in one specific area may be redundant or ineffective in another. Moreover, the original DoWhaM does not inherently distinguish between actions that keep the agent within already explored areas and actions that push agent into new regions.

To address these issues, we propose **A***rea-aware* **D***oWhaM* **A***daptation* (**ADA**). ADA extends DoWhaM to state-conditioned statistics and with two additional bonuses that are directly related to spatial novelty.

ADA builds on the core action-usefulness mechanism of the original DoWhaM. To address the limitations of purely action-based exploration in spatial tasks, it introduces two additional intrinsic reward components. These are designed to encourage specific spatial behaviors.

- **Expansion Bonus (Section 3.2.1):** This component encourages the agent to reveal new parts of the map that were previously outside of its field of view.
- **Achievement Bonus (Section 3.2.2):** This component rewards the agent for moving into regions that were previously observed (seen) but not yet actually visited.

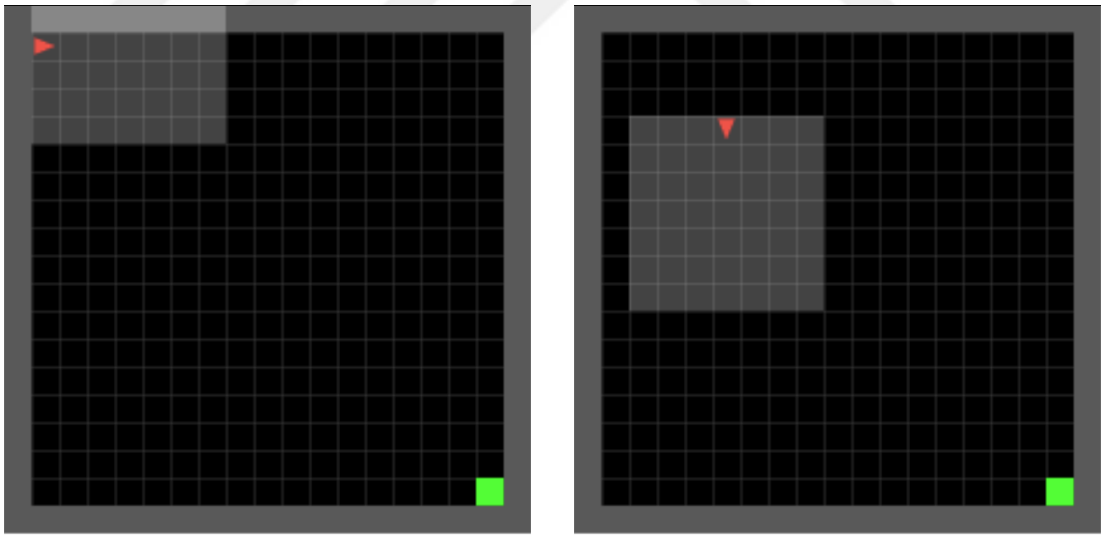
The final intrinsic reward is computed by combining these three bonuses (action-usefulness, expansion, and achievement) and dividing them by an episodic state-count term, following the normalization strategy of the original method.

3.1 *Observation Encoding and State–Action Statistics*

In this section, we describe the technical components required to implement the proposed method ADA. We will discuss the hashing mechanism for encoding partial observations, the statistical counters for tracking interactions, and the spatial memory sets used to distinguish between visited and unseen regions.

3.1.1 Observation hashing

In realistic settings, an RL agent's perception is limited. The agent is unable to access the current state but gets observations. These observations can be coordinates of the agent, RGB images, vector representations, or textual representations, depending on the description of the environment. Agent then decide an action using these observations. In this work, we assume the environment is *partially-observable*, meaning that the agent receives an egocentric view. Figure 2 shows that the gray area shown in grid world is what agent receives as an observation in a MiniGrid environment.

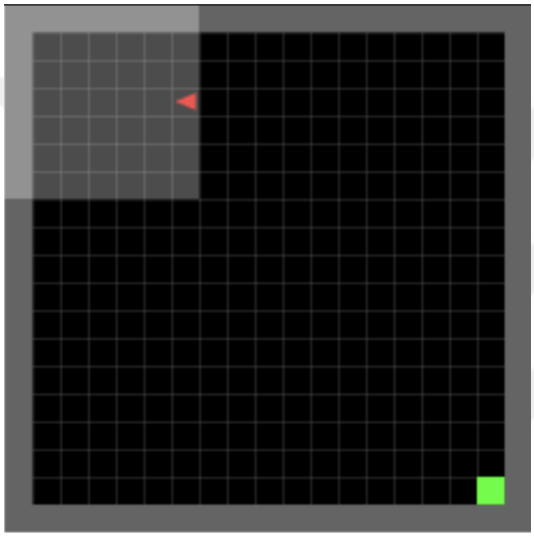


(a) Observation hash $0fe08d3cbecb36e4$ (b) Observation hash $6b6a12e48dbf9edc$

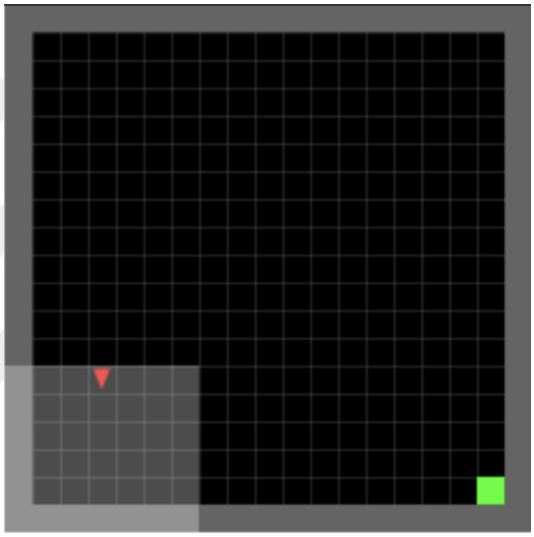
Figure 1. Partial-observation demonstration in `Empty` environment of *MiniGrid*. Agent only receive perception of the gray area as a observation.

Since observation might vary, we needed a function that maps this observation into numeric identifier. The aim is to implement the proposed reward functionality to any environment or observation by just defining hash function suitable for the environment. To obtain a compact and reproducible index for each observation, ADA uses a simple

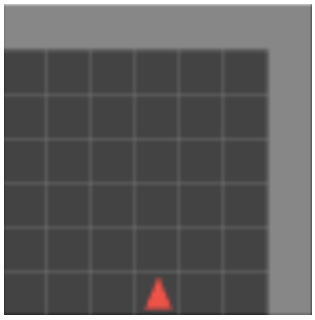
hash function $h(\cdot)$ that maps the observation o_t at time t into an integer code. The exact choice of $h(\cdot)$ is not critical. The only requirement is that calculation should be deterministic. If two observations are identical, they should receive the same code across whole training (should not be effected by episode reset). For example, Figure 2 shows that in a large empty environment of *MiniGrid* even though the agent is in different corners of the environment, it receives exactly same partial observation. As shown in figure 2c, both agent sees a long wall on the agent's right-hand side and another wall directly ahead, so these partial-observations are mapped to the same hash even though the global position changes.



(a) Observation hash $4e43f6188dd22fc9$



(b) Observation hash $4e43f6188dd22fc9$



(c) Egocentric partial observation received in both scenarios.

Figure 2. Both agents will generate the same hash because the partial observation colored in gray is the same.

Hash collisions are possible but rare, and in practice they have negligible effect on our approach.

From the algorithm's point of view, $h(o_t)$ is simply a discrete label for the current

partial observation. Whenever the same view appears again, the same label is used and the corresponding counters are updated.

3.1.2 State–action statistics

Same as the original method, ADA also maintains three types of counters indexed by the observation hash $h(o_t)$ and the action a:

- $U(h(o_t), a)$: how many times action a was taken in observation $h(o_t)$,
- $E(h(o_t), a)$: how many of these uses actually changed the observation,
- $N_\tau(h(o_t))$: how many times $h(o_t)$ was visited in the current episode τ.

For each transition (o_t, a_t, r_t, o_{t+1}) we compute $h(o_t)$ and $h(o_{t+1})$ and update as following.

$$U(h(o_t), a_t) \leftarrow U(h(o_t), a_t) + 1, \tag{3.1}$$

$$E(h(o_t), a_t) \leftarrow E(h(o_t), a_t) + \mathbf{1}_{\{o_t \neq o_{t+1}\}}, \tag{3.2}$$

$$N_\tau(h(o_{t+1})) \leftarrow N_\tau(h(o_{t+1})) + 1. \tag{3.3}$$

Conditioning the statistics on $h(o_t)$ makes the algorithm sensitive to where an action is used in given observation. Turning left in front of a door and turning left in the middle of an empty room are treated as different and can have different effectiveness ratios.

At the beginning of each episode we reset all episodic counts $N_\tau(\cdot) = 0$. The global usage and effectiveness counts U and E are kept across episodes so that the method can reuse information from earlier episodes.

3.1.3 Spatial memory under partial observability

The spatial bonuses that we introduce use simple set of variables that are maintained by the algorithm.

Let $\mathcal{H}_t = (s_0, a_0, s_1, \ldots, s_t)$ denote the history of states and actions up to time t. Let $\mathrm{p}(s_t)$ be a function that extracts the agents position from state s_t, $\mathrm{d}(s_t)$ be a function

that extracts the agents direction from state s_t. Then we set;

$$P_t \leftarrow \mathrm{p}(s_t), \tag{3.4}$$

$$D_t \leftarrow \mathrm{d}(s_t). \tag{3.5}$$

let $\mathcal{V}(o_t, P_t, D_t)$ be the visibility operator returning the set of observable coordinates at time t.

The spatial memory sets are defined as follows:

- $P_t^{visited}$: The set of positions that the agent has *visited* up to time t in the current episode:

$$P_t^{visited} = \bigcup_{k=0}^{t} \{\mathrm{p}(s_k)\}$$

- P_t^{unseen}: The set of positions that have been *observed* in the history of partial observations but have not yet been visited by the agent:

$$P_t^{unseen} = \left(\bigcup_{k=0}^{t} \mathcal{V}(o_k, P_k, D_k) \right) \setminus P_t^{visited}$$

- P_{t+1}: The agent's next position after executing a_t:

$$P_{t+1} = \mathrm{p}(s_{t+1})$$

- $\mathcal{V}_t$: The positions that are currently visible in the partial observation at time t:

$$\mathcal{V}_t = \mathcal{V}(o_t, P_t, D_t)$$

- $\mathcal{V}_{t+1}$: The positions that will be visible at time $t+1$ after executing a_t:

$$\mathcal{V}_{t+1} = \mathcal{V}(o_{t+1}, P_{t+1}, D_{t+1})$$

At the beginning of each episode we initialise $P_0^{visited} = \{P_0\}$, set $P_0^{unseen} = \emptyset$.We do *not* assume that agent has access to the full map and we never pre-fill P_t^{unseen}. Instead,

both $P_t^{visited}$ and P_t^{unseen} are built incrementally from what the agent has actually seen (observed) so far.

Given the new observation o_{t+1} and position P_{t+1}, the algorithm updates these sets as follows; First, it extracts positions that are visible in the partial observation. These local coordinates need to be transferred to global ones to be able to keep the track. The set of newly visible positions is updated as $\mathcal{V}_{t+1} = \mathcal{V}(o_{t+1}, P_{t+1}, D_{t+1})$.

Then the three sets are updated by;

$$P_{t+1}^{visited} \leftarrow P_t^{visited} \cup \{P_{t+1}\}, \tag{3.6}$$

$$P_{t+1}^{unseen} \leftarrow \left(P_t^{unseen} \cup \mathcal{V}_{t+1}\right) \setminus P_{t+1}^{visited}, \tag{3.7}$$

In other words, every *new position* that appears in the current egocentric view is added to the frontier set, and any position that the agent actually steps on is moved from the frontier to the visited set. In this way P_t^{unseen} contains positions that are known to exist but have not yet been reached. The algorithm builds these sets using only the partial observation combined with the agent's current position and direction, without relying on the underlying global map layout.

3.2 *Action bonus*

The action bonus in ADA follows the same functional form as in DoWhaM. For a given state hash $h(o_t)$ and action a_t we compute the local effectiveness ratio

$$\rho_t = \frac{E(h(o_t), a_t)}{U(h(o_t), a_t)}. \tag{3.8}$$

This quantity is close to 0 when the action almost never changes the observation in this view, and close to 1 when it is almost always effective.

Using a decay parameter $\eta > 1$, the action bonus is

$$B(h(o_t), a_t) = \begin{cases} 1.0, & \text{if } U(h(o_t), a_t) = 1 \text{ and } E(h(o_t), a_t) = 1, \\ \dfrac{\eta^{(1-\rho_t)} - 1}{\eta - 1}, & \text{otherwise.} \end{cases} \tag{3.9}$$

The first case gives the maximum bonus 1.0 when an action is tried once and it is immediately effective. Otherwise, the bonus decays smoothly as ρ_t increases from 0

to 1.

3.2.1 Expansion bonus

In this work, we introduce *expansion bonus* which encourages the agent to increase the visible area of the environment. We define the set of newly seen positions as

$$\mathcal{S}_{\text{new}} = \left\{ p \in \mathcal{V}_{t+1} \ : \ p \notin \mathcal{V}_t \text{ and } p \notin P_t^{visited} \right\}. \tag{3.10}$$

So $\mathcal{S}_{\text{new}}$ contains positions that become visible for the first time at time $t+1$ and that were never visited before. If this set is non-empty, we set

$$I_{\text{exp}} = \mathbf{1}_{\{|\mathcal{S}_{\text{new}}|>0\}} \cdot B(h(o_t), a_t) \cdot \ln\big(1 + |\mathcal{S}_{\text{new}}|\big). \tag{3.11}$$

The logarithm keeps the bonus at a reasonable scale when an action suddenly reveals a large room, but still makes a difference between revealing one and many positions.

3.2.2 Achievement bonus

We also propose *achievement bonus* which focuses on actual movement into *unexplored* areas. It is triggered when the agent steps into a position that belongs to the unseen set:

$$I_{\text{ach}} = \mathbf{1}_{\{P_{t+1} \in P_t^{unseen}\}} \cdot B(h(o_t), a_t). \tag{3.12}$$

In other words, this occurs whenever agent reaches positions previously seen in its partial view. For example, when agent first enters the new room, it will get I_{exp} since there are new positions that agent never seen before. Each time agent steps into these locations, it will get achievement bonus. Even these positions were seen before, they are never visited. The idea is that, this behavior can encourage the agent to move into seen states, which might eventually triggers the expansion bonus again.

3.3 *Final intrinsic reward*

The intrinsic reward in ADA is only defined when the observation changes. If $o_t = o_{t+1}$, there is no bonus:

$$r_t^{\text{ADA}}(o_t, a_t, o_{t+1}) = \begin{cases} \dfrac{B(h(o_t), a_t) + I_{\text{exp}} + I_{\text{ach}}}{\sqrt{N_\tau(h(o_{t+1}))}}, & \text{if } h(o_t) \neq h(o_{t+1}), \\ 0, & \text{otherwise.} \end{cases} \tag{3.13}$$

The final intrinsic reward combines the three terms described above. Even though we define *expansion* and *achievement* bonuses separately, they complement each other to drive the agent toward the boundary of its knowledge. The normalization term includes the episodic count $N_\tau(h(o_{t+1}))$, which reduces the bonus when the same observation hash is visited many times in one episode. This follows the same idea as the state-count normalization in the original DoWhaM and in Count-Based exploration.

3.4 *Overall Algorithm*

Algorithm 1 summarises the bonus mechanism defined in ADA. Given the current and next observation after action is taken, ADA first updates the state-action usage and effectivenes counts for to calculate action-usefulness. It then identifes newly visible frontier positions and newly reached unseen positions to calculate expansion and achievement bonuses. Finally, these three bonuses are combined and normalized by episodic state count to produce final intrinsic reward.

Algorithm 1 ADA intrinsic reward under partial observability

Require:

Intrinsic counters U, E, N_τ, Decay parameter $\eta > 1$, initial observation o_0 and initial agent position P_0

$N_\tau \leftarrow 0$
$P_0^{visited} \leftarrow \{P_0\}$
$P_0^{\text{unseen}} \leftarrow \emptyset$
for $t = 0, 1, 2, \ldots$ until episode *terminates* or *finishes* **do**
  o_t and P_t are given by the environment at time t
  $a_t \leftarrow \pi(o_t)$ {action selection (outside ADA)}
  Calculate $h(o_t)$
  Execute a_t in the environment to obtain o_{t+1}, next position P_{t+1} and extrinsic reward r_t^{env}
  Calculate $h(o_{t+1})$
  $U(h(o_t), a_t) \leftarrow U(h(o_t), a_t) + 1$
  if $h(o_t) \neq h(o_{t+1})$ **then**
    $E(h(o_t), a_t) \leftarrow E(h(o_t), a_t) + 1$
  end if
  $N_\tau(h(o_{t+1})) \leftarrow N_\tau(h(o_{t+1})) + 1$
  $\mathcal{V}_t \leftarrow$ visible_positions(o_t)
  $\mathcal{V}_{t+1} \leftarrow$ visible_positions(o_{t+1})
  $P_{t+1}^{visited} \leftarrow P_t^{visited} \cup \{P_{t+1}\}$
  $P_{t+1}^{unseen} \leftarrow \left(P_t^{unseen} \cup \mathcal{V}_{t+1}\right) \setminus P_{t+1}^{visited}$
  $\mathcal{S}_{\text{new}} \leftarrow \{p \in \mathcal{V}_{t+1} \mid p \notin \mathcal{V}_t \wedge p \notin P_t^{visited}\}$
  if $h(o_t) = h(o_{t+1})$ **then**
    $r_t^{ADA} \leftarrow 0$
    continue to next time step $t+1$
  end if
  $\rho_t \leftarrow \dfrac{E(h(o_t), a_t)}{U(h(o_t), a_t)}$
  if $U(h(o_t), a_t) = 1$ **and** $E(h(o_t), a_t) = 1$ **then**
    $B \leftarrow 1.0$
  else
    $B \leftarrow \dfrac{\eta^{(1-\rho_t)} - 1}{\eta - 1}$
  end if
  if $|\mathcal{S}_{\text{new}}| > 0$ **then**
    $I_{\text{exp}} \leftarrow B \cdot \ln\left(1 + |\mathcal{S}_{\text{new}}|\right)$
  else
    $I_{\text{exp}} \leftarrow 0$
  end if
  if $P_{t+1} \in P_{\text{unseen},t}$ **then**
    $I_{\text{ach}} \leftarrow B$
  else
    $I_{\text{ach}} \leftarrow 0$
  end if
  $r_t^{ADA} \leftarrow \dfrac{B + I_{\text{exp}} + I_{\text{ach}}}{\sqrt{N_\tau(h(o_{t+1}))}}$
  return $r_t^{ADA} + r_t^{\text{env}}$ to the learning algorithm
end for

CHAPTER 4 : EXPERIMENT SETUP

4.1 *Evaluation Strategy*

In this work, we aim to answer two research questions with empirical results. The first question focuses on comparing sample-efficiency of ADA, DoWhaM and Count-Based exploration.The second question aims to show and compare the performance of ADA under the environments that their layout and object positions changed with every episode reset (see Section 1).

To answer these, there are three different metrics to compare the results. *Episode Length Mean* metric shows how many steps the agent takes to reach the goal on average over evaluation episodes. A faster reduction in episode length indicates higher *sample-efficiency* during the early stages of training. *Percentage Visited* metric shows how much of the grid is covered by the agent within an episode, helping to show the broadness of exploration before convergence. *Episode Reward Mean* represents the total reward collected by the agent within an episode, which includes both the time-dependent extrinsic reward (see Eq. 4.3) and the intrinsic reward generated by the specific motivation method. Since this metric includes intrinsic reward, higher value does not necessarily mean a superior performance. While high reward with shorter episodes can mean that the agent solves and receives a higher *extrinsic* reward, high reward with long episodes can imply that the agent lingers in the environment to collect *intrinsic* rewards.

For each experiment, we used 10 random seeds. In every 3 training iterations, we evaluated the current policy for 10 episodes. Then, the training algorithm returns the average results of these 10 episodes. The results presented in this section represent the policy's evaluation performance measured at specific checkpoints, rather than the online training data. Following the Ray RLLib framework (Liang et al., 2018), we use training iterations as the primary metric. Since episode length varies depending on the performance of the intrinsic motivation method, using iterations ensures consistent evaluation points across different runs. The total number of iterations is always fixed for each environment. Finally, we bootstrapped the evaluation metrics of these 10 seeds to create results using 95% bootstrap confidence intervals (Tibshirani and Efron, 1993). This ensures that results are not just bounded to lucky seeds but instead show

the mean performance as well as the upper and lower bounds.

In the following sections, we describe the five different MiniGrid (Chevalier-Boisvert et al., 2023) environments designed to evaluate the proposed method. We first introduce the common configuration shared across all tasks, including the action space and reward function, followed by a detailed description of each environment.

4.2 *General Configuration*

All environments share a common grid dimension of 19×19. The maximum step limit $T_{\max}$ defines the timeout threshold. The environment will be *terminated* if the agent fails to achieve the goal after this many steps. Let $\mathcal{W}$ denote the grid width:

$$T_{\max} = 4 \times \mathcal{W}^2 \tag{4.1}$$

For a grid size of 19:

$$T_{\max} = 4 \times 19^2 = 4 \times 361 = 1444 \tag{4.2}$$

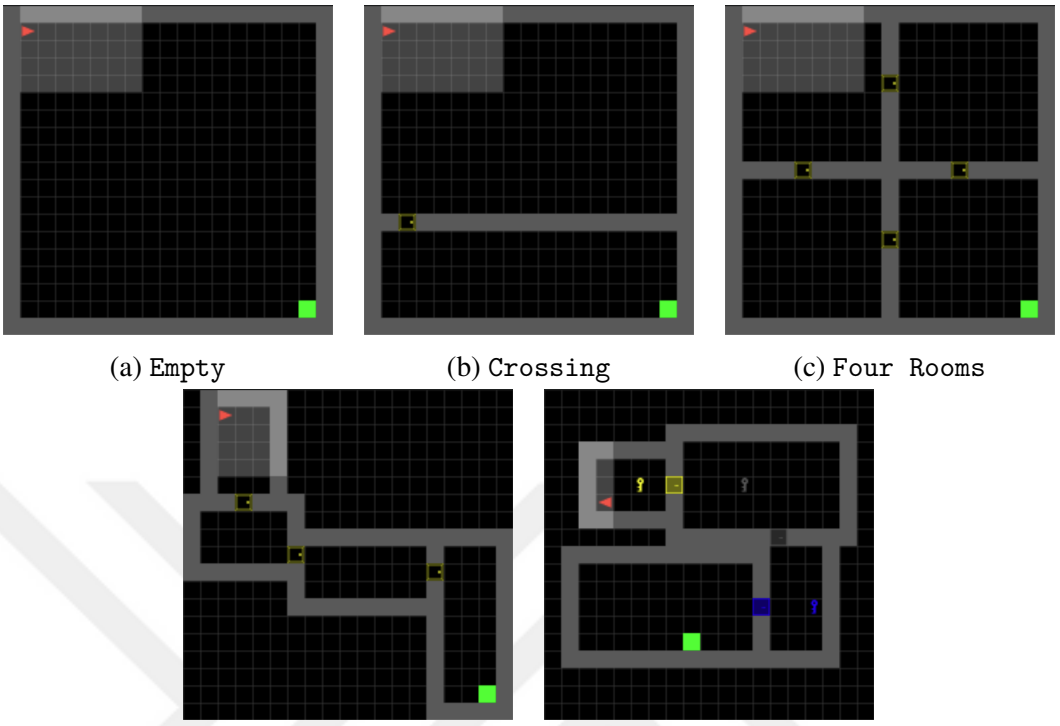
If the agent reaches the goal within the maximum step limit $T_{\max}$, then the episode is *done*.

Environments provide time dependent *extrinsic* reward that encourages agent to solve the tasks as quickly as possible. The agent receives no *extrinsic* reward unless it reaches to the goal state. Assuming that the agent reaches the goal at time step t, the *extrinsic* reward is calculated as

$$r_t^{\text{env}} = 10 - 9 \cdot \frac{t}{T_{\max}} \tag{4.3}$$

This shaping scheme returns reward in the interval $[1, 10]$. If agent reaches the goal early in the episode receive a reward close to 10, if it reaches the goal near the time limit, reward will get close to 1.

4.3 *Environments*



(a) `Empty` (b) `Crossing` (c) `Four Rooms`

(d) `Multi-room` (e) `Multi-room Key`

Figure 3. Visual representation of the environments used in the experiments. The agent is represented by a red triangle, the goal by a green square.

These environments were chosen to systematically evaluate the performance of intrinsic motivation techniques from relatively simple to progressively complex navigation and exploration tasks. For all setups, environments are partially observable. Agent also cannot see through the walls or behind doors unless door is opened.

There is seven different discrete actions that agent can use. *left* and *right* rotates agent's direction by 90 degrees, *forward* moves agent a one cell in current direction, *pickup* collects an object (*key* in our setup), *drop* allows the agent to release the carried object to interact with others (*key* in our setup), *toggle* triggers an interaction with the front object (*door* in our setup), and finally *done* is used for a finish signal when successfully reaching the goal.

4.3.1 Empty Environment

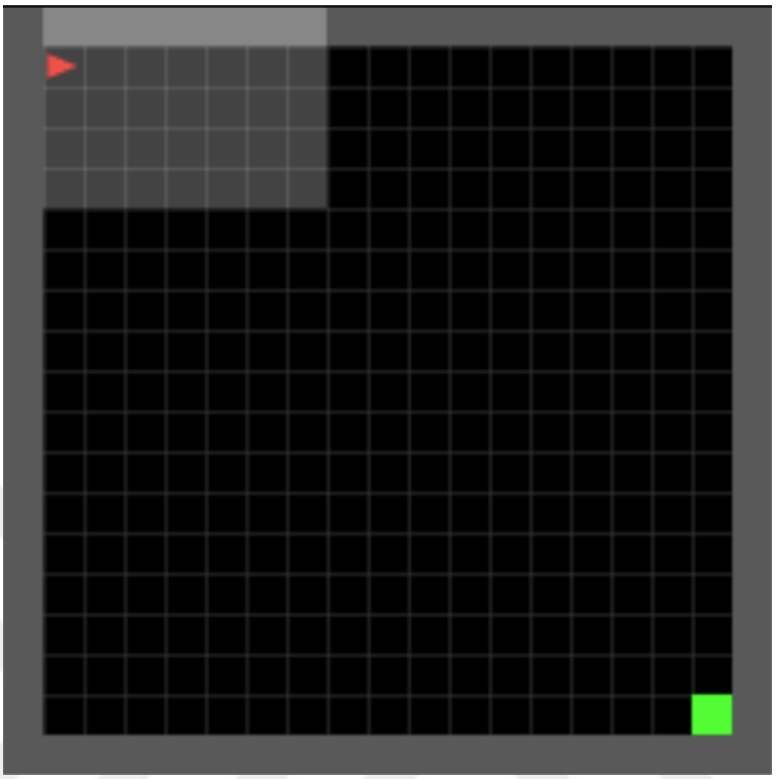


Figure 4. `Empty` environment layout. The agent starts in top left corner and must reach the goal at bottom right.

The `Empty` environment is the simplest one in our set. There is 19×19 empty grid without any object or obstacles. Agent always start at top left corner and goal is always at bottom right corner. Once path to the goal is discovered, the task becomes trivial and there is little need for sophisticated exploration.

This environment serves as a baseline for **RQ1**. It allows us to measure the raw sample-efficiency of the proposed method in the absence of obstacles, ensuring that the intrinsic bonuses accelerate learning even when sophisticated exploration is not strictly necessary.

4.3.2 Crossing Environment

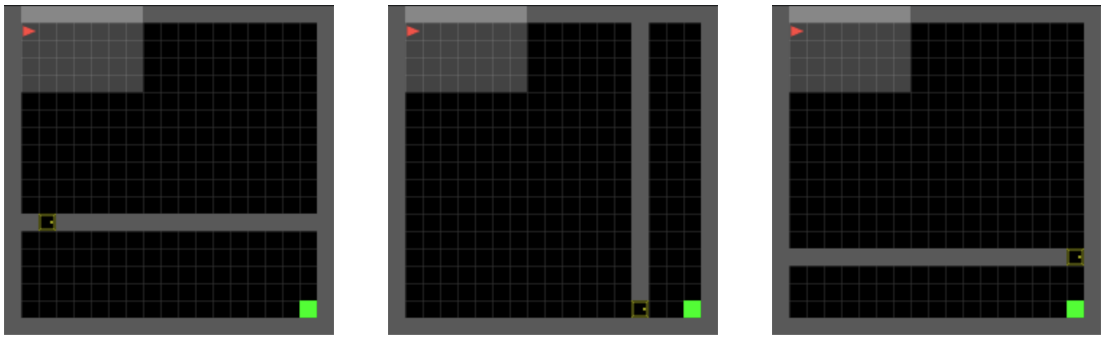


Figure 5. Different environment setups for `Crossing` environment that changes every episode reset.

The `Crossing` environment adds an extra layer of difficulty on top of the previously discussed Empty setup. At every episode reset, the grid is split into two regions by a wall, which can be either vertical or horizontal. A single yellow door is placed somewhere along this wall. To reach the goal located in the bottom–right corner, the agent must first discover this door, toggle it, and then enter the second room before navigating to the goal.

This setup addresses **RQ2** by introducing randomized obstacles. Since the wall and door positions change at every reset, the agent cannot rely on memorizing a fixed trajectory. Instead, it must learn a robust exploration strategy to locate the door and navigate to the goal.

4.3.3 Four Rooms Environment

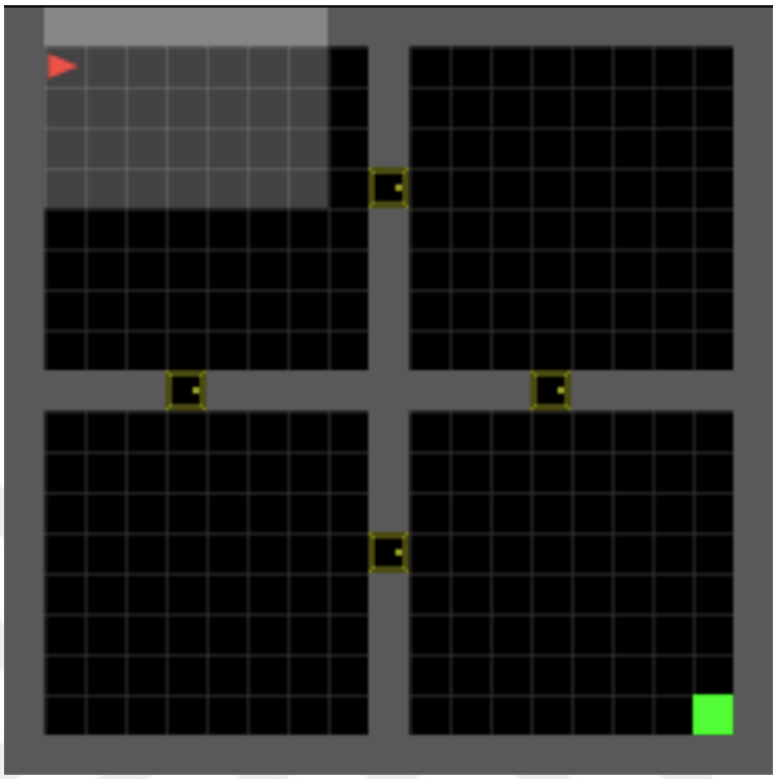


Figure 6. Layout for `Four Rooms` Environment

`Four Rooms` environment is diving the 19×19 grid both vertically and horizontally to create four rooms. Along these walls, there are 4 yellow doors. The starting position of an agent and the goal position are fixed. Agent must open at least two doors to navigate the goal before timeout. This scenario evaluates the agent's ability to explore and navigate efficiently through multiple rooms to locate the goal.

This environment targets **RQ1** by evaluating exploration performance in a structured, static layout. It tests the agent's ability to navigate multiple connected rooms, providing a benchmark for sample-efficiency in medium-complexity tasks where simple random exploration is often insufficient.

4.3.4 Multi-room

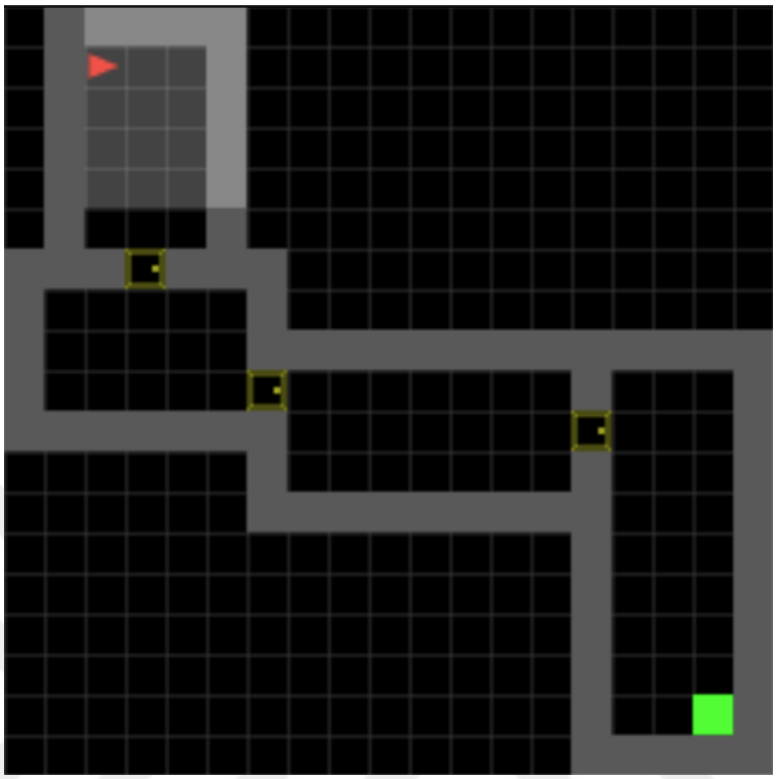


Figure 7. Layout for `Multi-room` Environment

`Multi-room` consists of a sequence of rooms connected by doors in a fixed layout. Here the geometry remains static across all episodes. This environment evaluates the agent's ability to memorize a long-horizon path in a stable setting.

This environment continues the investigation of **RQ1** in a long-horizon setting. It evaluates whether the spatial bonuses accelerate learning in tasks that require deep exploration, while also highlighting the convergence differences in static layouts.

4.3.5 *Multi-room with Key*

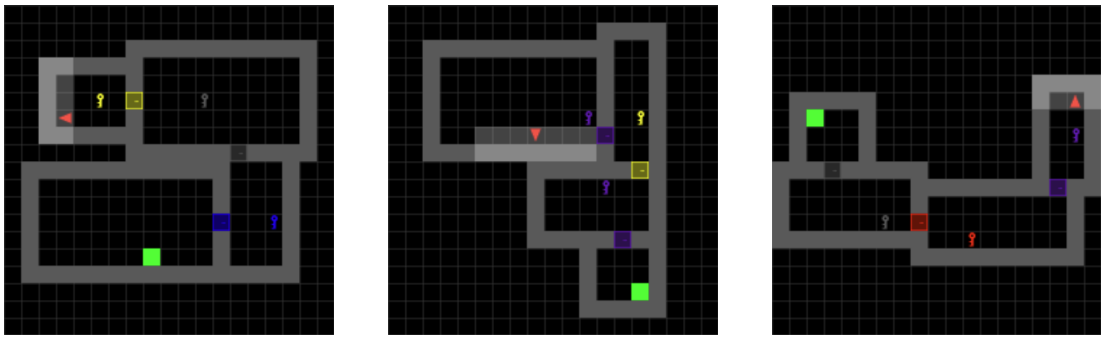


Figure 8. Different environment setups for `Multi-room with Key` environment

The final environment `Multi-room with Key` is designed as the most challenging environment in this work. At every environment reset, the start position and direction of the agent, the goal position, the layout, the size of the rooms, as well as the colors and locations of doors and keys are all randomized (see Figure 8). There are always 4 interconnected rooms. In this configuration, the room size consists of a minimum of 4 and a maximum of 12 tiles. To progress through the environment, agent must first navigate to the correct key, and then locate and open the door whose color matches the acquired key. In this environment, simple navigation is not sufficient. Agent must figure out about sequence of sub-tasks so that it can effectively plan the horizon. Since the layout of the environment is also changing, memorizing the interactable object locations is useful.

This environment is central to **RQ2**. By randomizing the layout and object positions in every episode, it tests the **robustness** of the exploration strategy. It allows us to investigate if the ADA heuristic remains effective in dynamic environments where the agent cannot rely on memorizing state-visitation counts.

4.4 *Hyper-parameters*

All experiments in this work were implemented using the Ray RLlib framework (Liang et al., 2018; Wu et al., 2021) using Torch (Paszke et al., 2019). RLlib is an open-source library that provides implementations of standard reinforcement learning algorithms and exposes a comprehensive set of hyper-parameters for PPO. In deep reinforcement

Table 1. PPO hyper-parameters used in all experiments.

Parameter	Value
Algorithm	PPO
Discount factor γ	0.99
GAE parameter λ	0.95
Use critic / GAE / KL	True / True / True
KL coefficient	0.2
KL target	0.01
Learning rate lr	2.5×10^{-4}
Train batch size	16384 (per learner)
Mini-batch size	2048
SGD epochs per update	6
Policy clip parameter	0.3
Value-function loss coefficient	0.5
Value-function clip parameter	10.0
Entropy coefficient schedule	$(0, 0.006) \rightarrow \left(\frac{1}{2}N, 0.002\right) \rightarrow (N, 0.0)$

learning, the choice of hyper-parameters is critical, as the agent's convergence and stability are often highly sensitive to these values.

Table 2. Policy and value network architecture.

Parameter	Value
Base MLP hidden sizes	[512, 512]
Base MLP activation	`tanh`
Post-MLP hidden sizes	[512]
Post-MLP activation	`relu`
Value / policy layer sharing	Actor-Critic
Observation dim	88
Grayscale	False
Zero-mean normalisation	True
Framestack	False
Recurrent layer	LSTM
LSTM cell size	256
Max sequence length	64
Use previous action	True
Use previous reward	True
Attention	Disabled

Table 1 and 2 summarize the key hyper-parameters used in our experiments. These settings remain consistent across all training runs. For intrinsic motivation, decay parameter η is set to 40 and intrinsic reward scale is set to 0.05. We denote total number of environment time-steps for the entire training process as N.

CHAPTER 5 : RESULTS

5.1 *Empty Environment*

We evaluate DoWhaM, the Count-Based bonus and ADA in this setting to check that these methods behave sensibly and to compare their sample efficiency in a controlled scenario.

For each intrinsic motivation technique, we train for 10 seeds for 1 million environment steps.

5.1.1 Episode Length Analysis

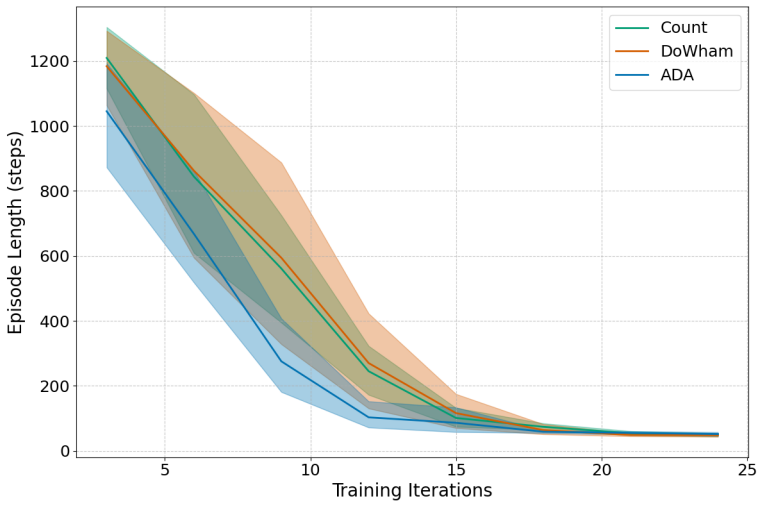


Figure 9. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Empty` environment in terms of episode length.

At iteration 3, all three methods are still close to random and episodes are long. The bootstrap mean is 1183.66 steps for DoWhaM, 1209.15 for Count and 1045.08 for ADA. Already here ADA achieves about an 11.7% reduction in episode length relative to DoWhaM, but at this point the policy is still mostly random.

The advantage becomes clearer over the next few iterations. At iteration 6, the means drop to 861.90 (DoWhaM), 843.69 (Count) and 667.32 (ADA), which corresponds to reductions of roughly 22.6% compared to DoWhaM and 20.9% compared to Count-Based. By iteration 9, the difference is large: DoWhaM still spends on average 594.22 steps per episode and Count-Based 561.20 steps, while ADA is down to 275.50 steps (around 53.6% and 50.9% reductions). By iteration 12, the mean

episode length of ADA is 103.08 steps, compared to 270.13 for DoWhaM and 245.07 for Count-Based. This corresponds to a 61.8% reduction relative to DoWhaM and a 57.9% reduction relative to Count-Based. These numbers show that ADA approaches its final performance level much earlier in training than the two baselines, which we interpret as a clear gain in sample-efficiency in this environment.

The confidence intervals in Figure 9 support the same pattern. At iteration 9 the 95% interval for ADA is $[181.03, 407.47]$, while DoWhaM has $[328.24, 887.50]$ and Count-Based has $[394.67, 725.14]$. The entire band of ADA lies well below the means of the baselines and far below their upper bounds. At iteration 12, the interval for ADA is $[71.79, 152.22]$ compared to $[130.48, 422.75]$ for DoWhaM and $[172.33, 322.81]$ for Count-Based. Across seeds, ADA consistently reaches much *shorter* episodes earlier in training, which is a sign that it converges faster in this environment.

After this initial phase, the curves flatten and the ordering changes. Around iteration 15, all three methods have essentially solved the environment, and the episode lengths vary in a relatively narrow range (for example, at iteration 15 the means are 116.11 for DoWhaM, 101.17 for Count-Based and 86.35 for ADA). Further into training, the episode lengths of DoWhaM and Count-Based become slightly shorter on average than those of ADA. At iteration 30, DoWhaM and Count-Based are at 44.32 and 46.75 steps, whereas ADA remains at 49.90 steps; at iteration 60, the means are 50.49 (DoWhaM), 40.01 (Count) and 43.68 (ADA). In this converged regime the confidence intervals for all three methods are narrow and overlapping, and the differences are small compared to the earlier gap.

5.1.2 Episode Reward Analysis

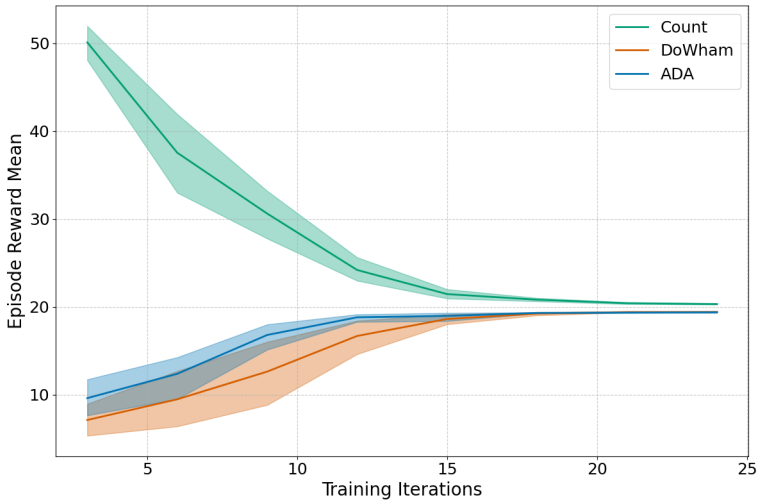


Figure 10. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Empty` environment in terms of episode reward.

In the empty environment, episode returns quickly stabilize for all three exploration strategies. This reflects the relatively simple structure of the task. The Count-based baseline starts with a very high mean return of roughly 50 at iteration 3. However, it decreases and converges to about 20 by iteration 15, remaining close to this level up to iteration 60. This behavior is consistent with an intrinsic signal that is initially strong but fades once most states have been visited.

Both ADA and DoWhaM exhibit a more gradual learning curve. DoWhaM starts with substantially lower returns, around 7 at iteration 3. It increases steadily, reaching approximately 18.6 by iteration 15 and converging to a narrow band around 19.3–19.5 between iterations 18 and 60. ADA follows a similar pattern but with slightly higher returns in the early and mid training phase. It starts around 9.6 at iteration 3 and reaches roughly 18.8 by iteration 12. Afterwards, it stabilizes around 19.4–19.5. Across iterations 18–60, the confidence intervals of DoWhaM and ADA overlap strongly, and both are only slightly below the Count-based baseline in terms of total reward.

Overall, in the empty environment, the three methods converge to almost the same final episode reward. The spatial bonuses introduced by ADA do not lead to large gains here. This is expected given the lack of meaningful structure in the environment. Once the agent has learned to move efficiently through an empty space, there are few opportunities where frontier or achievement intrinsic signals can provide additional

guidance beyond what Count-based exploration already offers.

5.1.3 Percentage Visited Analysis

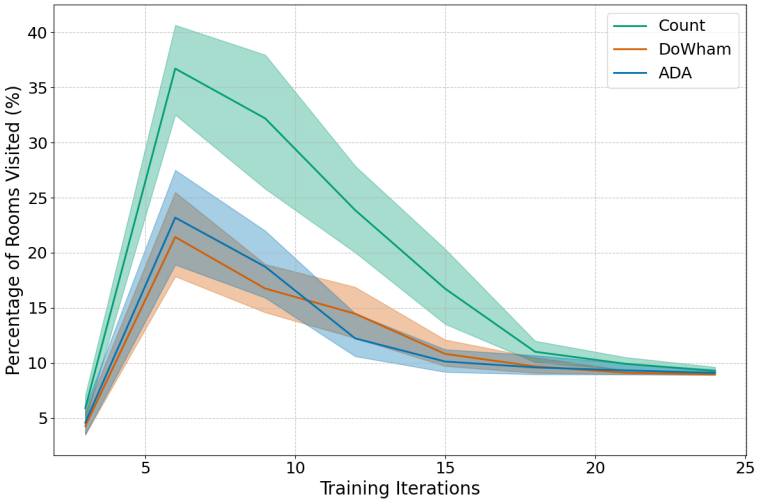


Figure 11. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Empty` environment in terms of percentage visited.

In the empty environment, all three methods show a similar pattern of rapid early exploration followed by convergence to a steady state coverage. The Count-based baseline initially visits only about 5.9% of the map at iteration 3. However, it then rapidly expands its coverage to approximately 36.7% at iteration 6 and 32.2% at iteration 9. As the novelty bonus vanishes, this behavior stabilizes. The percentage of visited positions declines towards roughly 9% by iteration 18 and remains in a narrow band of 8.9%–9.1% up to iteration 60.

The original DoWhaM and ADA follow the same qualitative trend but with a more moderate peak in coverage. DoWhaM reaches about 21.4% visited positions at iteration 6 and 16.8% at iteration 9. It then gradually converges to about 9.0%–9.3% from iteration 18 onwards.

On the other hand, ADA exhibits slightly stronger early exploration. It attains roughly 23.2% at iteration 6 and 18.7% at iteration 9. However, it also converges to a visited fraction of about 8.9%–9.1% in the later iterations. Overall, in this environment without obstacles, the spatial bonuses of ADA show only a small increase in early coverage compared to the original DoWhaM. All methods eventually settle on a very similar long-run percentage of visited positions.

5.1.4 *Conclusion: Empty Environment*

More concretely, all methods very quickly learn a near-optimal path to the goal and then stay there. The interesting part is the early phase. There, ADA drives episode length down much faster than both DoWhaM and the Count-Based baseline and does so with clearly lower bootstrap means and tighter confidence intervals. The Count-Based method, on the other hand, tends to visit a larger portion of the grid in the first few iterations and achieves relatively high rewards early, but with longer episodes. Once learning has stabilised, the three methods converge to very similar episode lengths, coverage and rewards.

5.2 *Crossing Environment*

For `Crossing` environment, we train 1M environment steps with 10 random seeds and results are bootstrapped over these seeds.

5.2.1 *Episode Length Analysis*

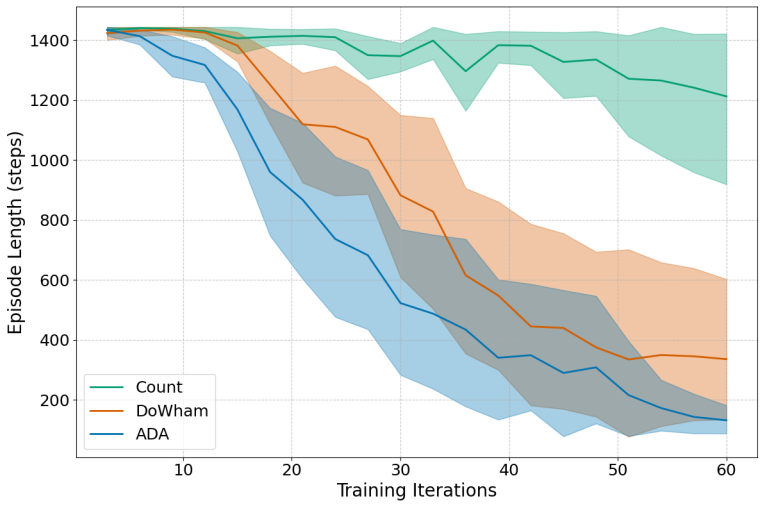


Figure 12. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Crossing` environment in terms of episode length.

For the Count-Based, agent fails to solve the task within one million time-steps. For the entire run, episode length mean is near the timeout. It starts with 1434.42 steps in iteration 3 and still 1212.77 steps in iteration 60 which shows that there is no significant findings to support that agent is learned to solve the task efficiently.

Both DoWhaM and ADA make clear progress. In iteration 15, the episode length mean has dropped to 1382.30 steps for DoWhaM and 1168.95 for ADA. This corresponds to a reduction of approximately 15.4% for ADA relative to DoWhaM and about 16.9% relative to Count-Based. By iteration 30 the gap is much stronger: DoWhaM needs on average 882.88 steps per episode, Count-Based still 1347.12 steps, while ADA has already reached 522.80 steps. In relative terms, ADA produces episodes that are about 40.8% shorter than DoWhaM and about 61.2% shorter than Count-Based at this point.

The bootstrap intervals in Figure 12 support the same picture. At iteration 30, the interval for ADA is $[282.98, 769.21]$, whereas DoWhaM lies in $[609.34, 1150.05]$ and Count-Based in $[1295.58, 1390.05]$. The interval of ADA is entirely below the Count-Based interval (no overlap) and mostly below the DoWhaM interval, which indicates that the improvement is not due to one or two lucky seeds.

Towards the end of training, ADA continues to improve and stabilises at much shorter episodes. At iteration 60, the mean episode length is 131.50 steps for ADA, compared to 335.66 for DoWhaM and 1212.77 for Count-Based. This corresponds to an approximate reduction of 60.8% relative to DoWhaM and about 89.2% relative to Count-Based. The confidence intervals at iteration 60 are $[87.57, 181.55]$ for ADA, $[133.81, 602.64]$ for DoWhaM, and $[918.86, 1421.77]$ for Count-Based. Again, the band of ADA is well separated from the Count-Based and clearly shifted downward compared to DoWhaM.

5.2.2 Episode Reward Analysis

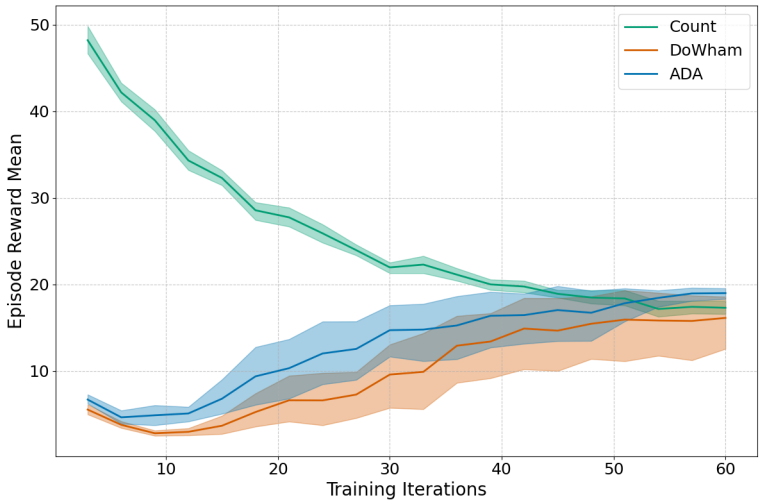


Figure 13. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Crossing` environment in terms of episode reward.

Naturally, the count method starts with very high reward as reward is decaying when agent is visited the same positions over and over again. The Count-Based starts with a high reward (mean 48.21 at iteration 3) but then steadily decreases to 17.31 by iteration 60. DoWhaM and ADA on the other hand, start with low rewards (5.55 and 6.70 at iteration 3) and increase over time as they learn to reach the goal (16.13 DoWhaM, 17.31 Count-Based, and 18.99 ADA).At the end of the training ADA achieves about 17.7% higher reward than DoWhaM and 9.7% higher reward than Count-Based. The 95% confidence intervals overlap, but ADA consistently lies at the top of the cluster in the later iterations, which matches reduction in episode length.

5.2.3 *Percentage Visited Analysis*

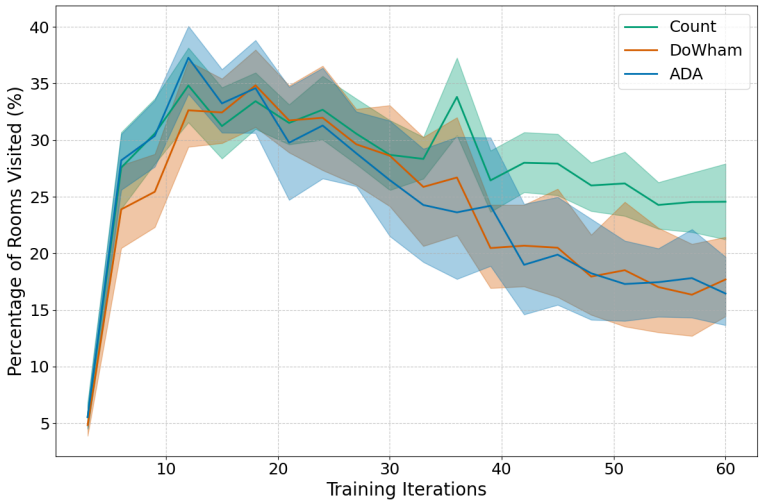


Figure 14. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Crossing` environment in terms of percentage visited.

For percentage visited metrics, all three methods show similar pattern. First there is a high percentage of environment coverage that follows the gradual decreases as the agent learns and policy becames stable.

Around iteration 12, ADA visits on average 37.27% of the environment, compared to 34.82% for Count-Based and 32.63% for DoWhaM. This means that at this stage ADA explores about 14% more than DoWhaM and roughly 7% more than Count-Based.

After that point, for all three methods, percentage is declining as they start to prefer shorter paths to the door and the goal instead of visiting the entire map.By iteration 60, Count-Based maintains the highest coverage (24.56%) despite failing to solve the task. However, this high value must be considered alongside with the failure in Episode Length (Figure 12). Since the agent consistently fails to reach the goal, this high coverage does not represent progress through the first room. This behavior of Count-Based implies that agent is stuck in the local optima by doing exhaustive local search. In contrast DoWhaM and ADA settle around 17.69% and 16.45% while successfully solving the environment by breaking this local optima with balancing exploration versus explotation.

For this `Crossing` task, it not is neccessary or critical to visit every part of the

map. Once the agent has learned to locate the door and move through it, additional exploration mainly adds extra steps without improving the return.

5.2.4 Conclusion: Crossing Environment

In the `Crossing` environment, ADA provides a clear benefit. It reaches shorter episodes much earlier than both Count-Based and DoWhaM and converges to the shortest trajectories among the three. Episode rewards also supports the same behavior. Although Count-Based starts with a high reward, it decays over time, and ADA achieves the highest return in the later part of training. The percentage–visited metric shows that ADA initially explores at least as broadly as the count and dowham but later focuses on more direct routes to the goal. Together, these results suggest that the additional spatial components in ADA help the agent to discover door–crossing behaviour in a more sample–efficient way than the original DoWhaM formulation and the Count–Based baseline.

5.3 *Four Rooms Environment*

For `Four Rooms` environment, we train 1.8M environment steps with 10 random seeds and results are bootstrapped over these seeds.

5.3.1 Episode Length Analysis

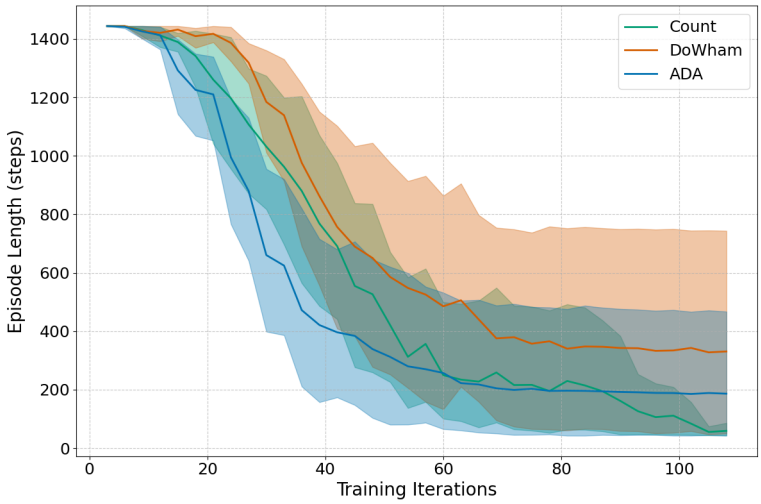


Figure 15. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Four Rooms` environment in terms of episode length.

At the beginning of the training, all three methods behave like a random policy. The differences start to appear once learning progresses. By iteration 15, Count-Based and DoWhaM are still close to the maximum episode length with means of 1389.44 and 1432.01 steps whereas ADA has already reduced the mean to 1292.27 steps. The gap widens over the next few iterations. At iteration 24, the mean episode lengths are 1386.44 for DoWhaM, 1196.97 for Count-Based and 994.82 for ADA. In other words, ADA produces episodes that are about 28% shorter than DoWhaM and roughly 17% shorter than Count-Based at this point in training. The confidence intervals also support this ordering. At iteration 24, the confidence interval of ADA ($[766.04, 1195.97]$) lies entirely below the interval of DoWhaM ($[1323.29, 1440.66]$). Furthermore, it overlaps only with the lower part of the Count-Based interval ($[954.95, 1405.79]$). This indicates that the proposed method achieves a significantly lower episode length compared to the original DoWhaM at this stage of training. The strongest advantage in terms of sample-efficiency appears around iteration 30. At this iteration, DoWhaM still requires an average of 1184.56 steps per episode and the Count-based method needs 1031.65 steps. In contrast, ADA has already reduced the mean episode length to 660.52 steps. This corresponds to approximately a 44% reduction relative to DoWhaM and around a 36% reduction relative to Count-Based which indicates a higher sample-efficiency in the early stages of training.

Following the initial acceleration phase, all methods continue to improve but the relative performance shifts. The Count-based method steadily reduces the episode length and eventually overtakes ADA. Around iteration 54, the mean episode lengths are 312.65 for Count-Based, 548.17 for DoWhaM, and 280.09 for ADA, with wide and overlapping confidence bands for the Count-based method and ADA.

By iteration 60, the Count-based method achieves a mean of 249.78 steps, slightly better than the 257.33 steps of ADA, while DoWhaM still lags behind at 485.41 steps. In the later stages of training, the gap between the Count-based method and ADA becomes more clear. At the end of training (iteration 108), the Count-based method reaches an average episode length of 59.29 steps, whereas ADA and DoWhaM remain at 186.50 and 330.90 steps. The bootstrap intervals at iteration 108 are $[43.77, 86.21]$ for Count-Based, $[42.52, 467.22]$ for ADA, and $[50.16, 743.67]$ for DoWhaM. This indicates that the Count-based method consistently achieves the shortest episodes once

training has converged.

In conclusion, ADA clearly accelerates the reduction in episode length compared to both DoWhaM and the Count-based method in the early stages. However, the Count-based method eventually catches up and outperforms ADA in the long run, whereas DoWhaM demonstrates the weakest performance among the three.

5.3.2 Episode Reward Analysis

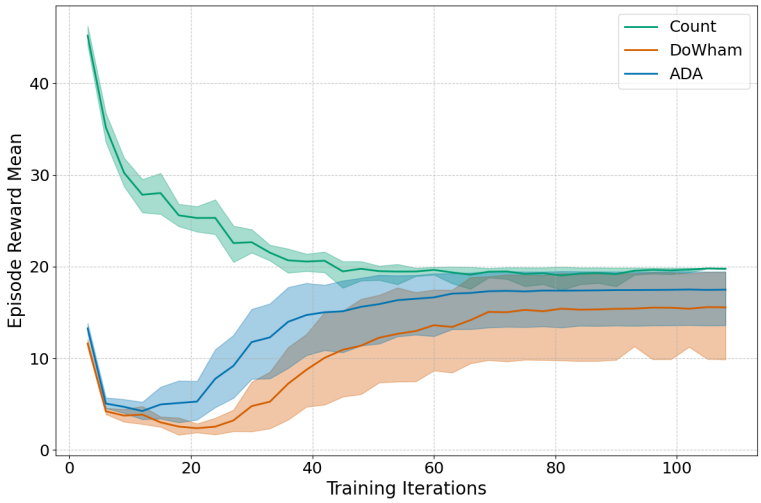


Figure 16. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Four Rooms` environment in terms of episode reward.

At the beginning of training (iteration 3), Count-Based starts with a relatively high episode reward of 45.19, while DoWhaM and ADA are at 11.62 and 13.26. For Count-Based, the mean reward then decreases and stabilises around 19.7–19.8 from iteration 90 onward (e.g. 19.80 at iteration 105 and 19.78 at iteration 108). In contrast, DoWhaM moves from 11.62 (iteration 3) to a stable band around 15.4–15.6 after iteration 90 (e.g. 15.56 at iteration 108), while ADA climbs from 13.26 and stabilises around 17.4–17.5 (e.g. 17.49 at iteration 108).

Because the absolute reward scales evolve differently across methods, we mainly focus on internal consistency with the episode length. For Count-Based, the decrease in reward is accompanied by a strong reduction in episode length (from 1444 steps down to about 59 steps at iteration 108), which is consistent with a shaping scheme where shorter path receive a more favourable return. For DoWhaM and ADA, the increase in episode reward goes hand in hand with shorter episodes (DoWhaM from 1444 to

330.90 steps, ADA from 1444 to 186.50 steps at iteration 108). In all three cases, the reward curves become flat after approximately 1.2–1.4M time steps, indicating that the policies have reached a stable behaviour in this environment.

In general, episode reward analysis aligns with the findings of the episode length analysis. The results indicate that the Count-based method converges to a distinct performance level. On the other hand, ADA consistently outperforms the original DoWhaM method, yet it does not reach the final performance of the Count-based baseline in this specific environment.

5.3.3 Percentage Visited Analysis

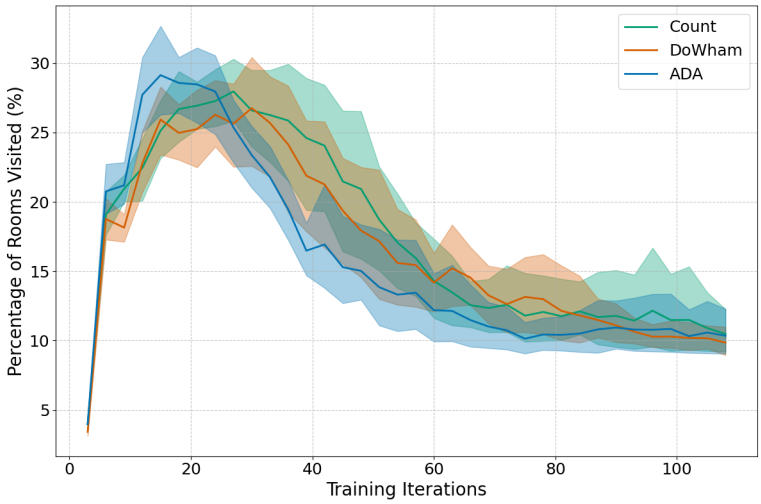


Figure 17. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Four Rooms` environment in terms of percentage visited.

The percentage visited metric shown in Figure 17 measures how much of the grid is covered by agent within episode. For this environment, all three methods shows the similar pattern. They start with high percentage in the beginning which is followed by a decay as the policy learns to solve the task efficiently.

In terms of the percentage of the environment visited, all three methods follow a similar trend. For the Count-based method, the mean percentage visited grows from 3.98% at iteration 3 to a peak around 28% between iterations 24 and 30 (e.g., 27.97% at iteration 27), before stabilizing around 10.48% in the final part of training. Similarly, DoWhaM increases from 3.42% to a peak of 26.76% at iteration 30, then gradually decreases to 9.86% at iteration 108. ADA follows this same pattern, starting

at 3.97%, reaching a maximum of 29.14% at iteration 15, and steadily dropping to 10.35% at iteration 108. The confidence intervals of all three methods overlap significantly throughout training, suggesting that they visit broadly similar portions of the environment once a good policy has emerged.

5.3.4 Conclusion: Four Rooms

In `Four Rooms` environment with no key, the three intrinsic motivation methods show a clear seperation in terms of final performance. From the episode length perspective, Count-Based method shows strongest convergence with tighter confidence interval. ADA shows better performance then original DoWhaM. At final iteration of 108, DoWhaM achives mean of 330.90 while ADA is at 186.5 which represents a reduction of approximately 43.6% in episode length.

These results suggest that the additional spatial components proposed in ADA offer an improvement over the original DoWhaM. However, since the structure of this task appears simple enough that counting visited states provides a strong shaping signal, the Count-based baseline generates better results in this environment. ADA is optimized for rapid discovery in sparse domains, while pure state-counting may allow for better fine-tuning once the environment is fully explored.

5.4 *Multi-Room*

For `Multi-Room` environment, we train 1M environment steps with 10 random seeds and results are bootstrapped over these seeds.

5.4.1 *Episode Length Analysis*

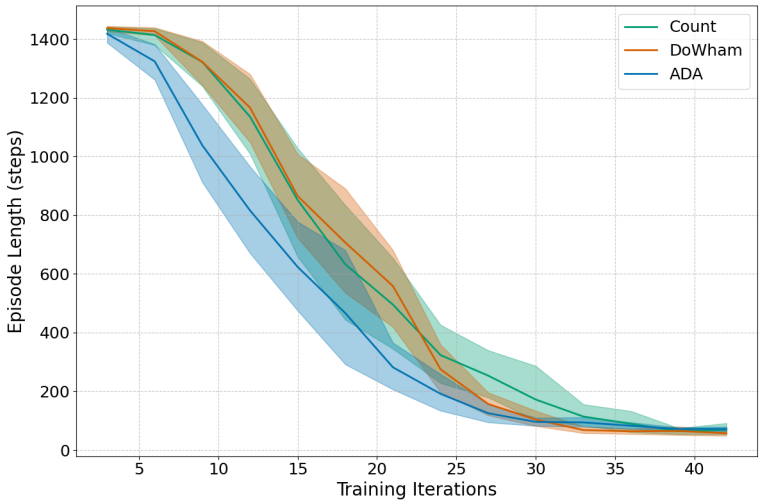


Figure 18. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Multi-Room` environment in terms of episode length.

In `Multi-Room` environment, all three methods perform similarly. At iteration 3, the bootstrap means are 1431.97 steps for Count-Based, 1437.25 for DoWhaM, and 1417.94 for ADA. All confidence intervals are tightly concentrated around the maximum episode length which is 1444 (Eq 4.2). As the training continues, differences are started to show. By iteration 9, Count-Based is around 1321.86 and DoWhaM is around 1322 where as the ADA is already at 1037.57. This represents an efficiency gain of approximately 21.5% compared to both baselines. At iteration 12, this efficiency gain increases significantly. ADA achieves a mean episode length of 816.03 steps, whereas the Count-based method is at 1135.22 steps and DoWhaM is at 1166.41 steps. This corresponds to episodes that are 28.1% shorter than the Count-based baseline and approximately 30.0% shorter than DoWhaM. This advantage persists through the middle of the training phase. At iteration 24, ADA reaches a mean of 191.31 steps ($[133.72, 258.45]$), while the Count-based method and DoWhaM remain at 323.28 steps ($[228.23, 426.08]$) and 274.75 steps ($[196.30, 359.08]$). By iteration 30, ADA further reduces the length to 95.74 steps, whereas DoWhaM is at 103.52 steps and the Count-based method is still at 171.74 steps. At this point, the gap to the Count-based baseline is large. Even compared to DoWhaM, the mean episode length of ADA is clearly lower.

After approximately 30 iterations, all three methods have essentially solved the navigation task and converged. From iteration 33 onward, DoWhaM achieves the shortest episodes on average. At iteration 36, DoWhaM is at 63.76 steps versus 89.19 for Count-Based and 82.65 for ADA. Towards the end of training the differences are small. At iteration 60, the means are 55.07 for Count-Based, 54.07 for DoWhaM and 69.07 for ADA.Overall, ADA provides a clear reduction in episode length in the early and mid phases, while Count-Based and DoWhaM catch up and slightly overtake it once the agents converge to near-optimal policies.

5.4.2 Episode Reward Analysis

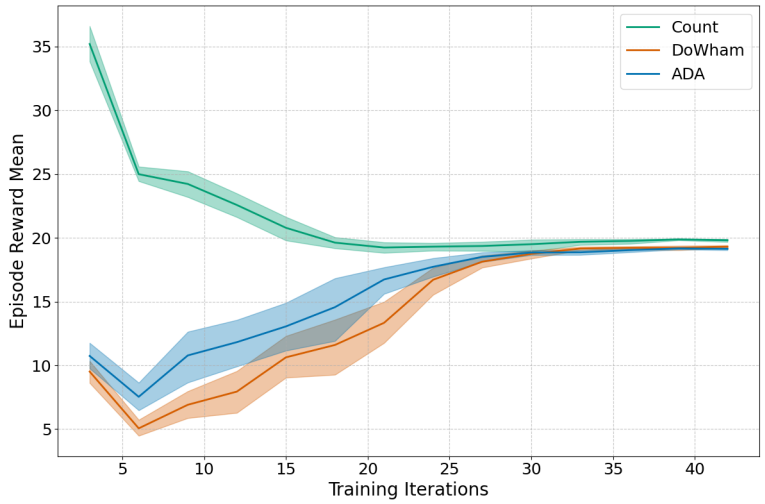


Figure 19. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Multi-room` environment in terms of episode reward.

In multi-room with fix position environment, Count-based method is again starting with high reward by design. At iteration 3 Count-Based starts with 35.20 while DoWhaM and ADA begin much lower at 9.51 and 10.74. As iteration continues Count-Based starts to drop and DoWhaM and ADA starts to close the gap. At iteration 15 count is dropped to 20.79 while DoWhaM increased to 10.63 and ADA is at 13.06. From iteration 30, all three methods are converged and there is a minimal difference between them. At iteration 30 Count-Based reaches 19.50 and increases it to 19.82 by the end of the training. Same small increase is also observed in DoWhaM that 18.74 at iteration 30 to 19.32 points in iteration 60. For ADA, this increase starts at 18.84 to 19.16 at the end of the training. All three methods after iteration 30 has a very tight

confidence intervals.

In other words, in terms of episode reward, ADA reaches same reward regime at the same time as DoWhaM but the main difference is how quickly the methods reduce episode length.

5.4.3 Percentage Visited Analysis

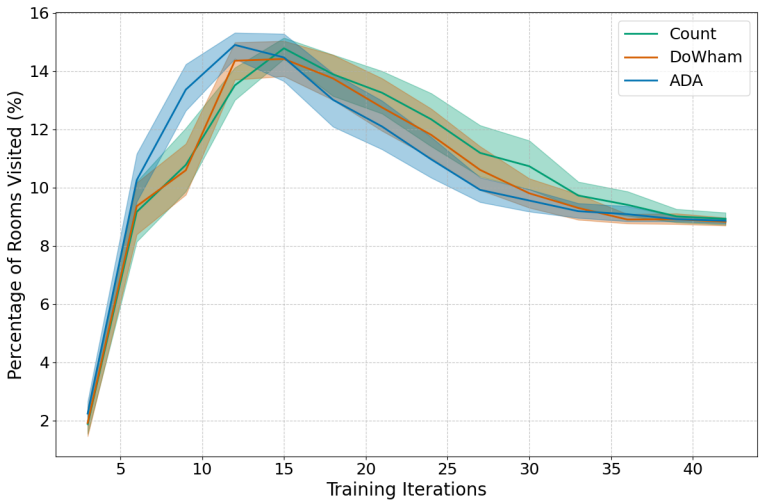


Figure 20. The performance of the proposed method ADA compared to DoWhaM and Count-based exploration methods in `Multi-room` environment in terms of percentage visited.

Percentage visited metrics show again that for all of the methods there is an initial increase corresponding to broad exploration, followed by a gradual decrease as the agents find a direct route.

At iteration 6 Count-Based and DoWhaM reach around 9% of the environment (9.17 and 9.36). At that point ADA has a slightly higher percentage with 10.25%. This difference is more shown at iteration 9, where ADA covers 13.36% of the grid, compared to 10.78% for Count-Based and 10.59% for DoWhaM.

After around the first quarter of the training, for all of the methods, percentage steadily declines as policies become more focused. At iteration 60, they converged to almost identical values of 8.67% for Count-Based, 8.69% for DoWhaM and 8.75% for ADA. These results suggest that all three methods visit roughly the same fraction of the environment while following near-optimal paths.

5.4.4 Conclusion: Multi-room

In the `Multi-Room` with fix position environment, ADA displays a clear sample-efficiency in terms of episode length. During first 30 iterations, it consistantly achieves much shorter episodes than both DoWhaM and Count-Based. Confidence intervals also support this and shows that before convergence, ADA provides lower episode lengths also suggest that this is not a single lucky seed. The percentage visited metric indicate that this improvement comes from a more targeted exploration of the rooms and doors.

Overall, in this environment ADA proves to be more beneficial in terms of sample-efficiency. However, as the training horizon extends, the performance gap between the methods diminishes as they all converge to similar levels. In practical settings where environment interactions are expensive or training is stopped once a target performance threshold is reached, results indicates that ADA will achieve desired performance way before the baselines.

5.5 *Multi-room with Key*

For the `Multi-room with Key` environment, we trained three different methods but Count-Based method is not be able to solve the problem. This behavior is expected for Count-Based method because counting the states will not be beneficial to agent since layout changes completely and this historical data will not be effective to use. It suggest that simple bonus for visiting the new states is not sufficient for this environment. Therefore we compare results of DoWhaM and ADA, which are trained for 10M time-steps with 10 different seeds.

Before analyzing the specific metrics, it is important to note the high variance observed in the results for this environment. Unlike the previous environments, the `Multi-room with Key` environment is procedurally generated. At every episode reset, the algorithm randomly constructs a new layout of rooms within the grid boundaries. Consequently, the difficulty of the task varies naturally from one episode to the next, causing fluctuations in the recorded metrics.

5.5.1 Episode Length Analysis

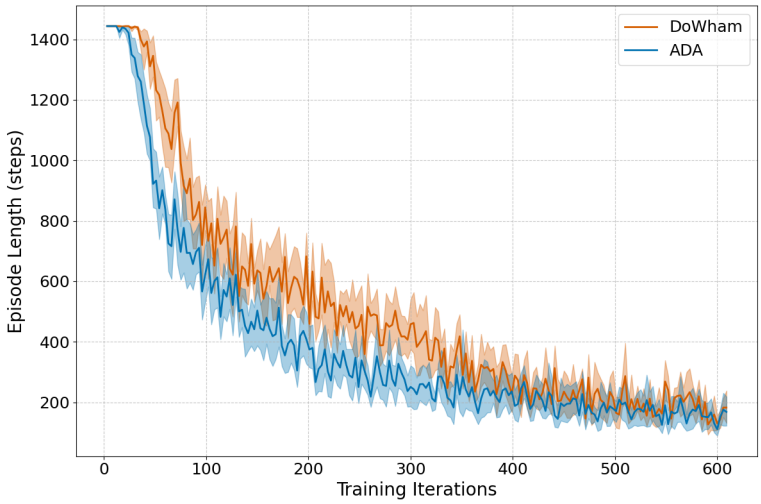


Figure 21. The performance of the proposed method ADA compared to DoWhaM in `Multi-room with Key` environment in terms of episode length.

The episode length analysis highlights a clear advantage for ADA in terms of sample-efficiency and convergence speed. In very early part of the training, including the iteration 12, both DoWhaM and ADA stay at 1444 which indicates that agent never reaches the goal before timeout. This behavior is expected given the complexity of the environment. After this initial phase, upper bound starts to decline at iteration 15 for ADA and much later at iteration 36 for DoWhaM. Although there seems to be improvement, the progress is relatively modest and still around 1300-1400 step range. Once learning starts to progress, ADA consistently achieves shorter episodes over the most of the training horizon than the original DoWhaM with a few spots that is on the side of original method. At iteration 60, ADA reduces the mean episode length to 837.62 steps. On the other hand the original DoWhaM method is at a mean of 1105.74 steps. Comparing these means, ADA provides an improvement of approximately 24.2% in sample-efficiency at this stage. Crucially, the confidence intervals do not overlap (upper bound of ADA 883.83 $<$ 978.06 lower bound of DoWhaM), which provides strong empirical evidence that the performance gap is not due to random seed variance.

This performance divergence reaches its peak during the mid-training phase. At iteration 300, ADA achieves a mean episode length of 250.81 steps. Meanwhile, DoWhaM still requires an average of 458.12 steps to solve the task. This corresponds

to a 45.2% reduction in episode length. The non-overlapping confidence bands at this iteration further confirm that ADA is significantly more robust in discovering the necessary sequence of sub-goals of finding the key, opening the door, and navigating to the goal.

Towards the end of the training at iteration 600, both methods converge to successful policies, although ADA maintains a slight lead. ADA reaches a final mean of 110.54 steps, compared to 129.65 steps for DoWhaM, representing a 14.7% advantage. However, unlike the earlier phases, the confidence intervals at this stage overlap (DoWhaM: $[107.98, 155.73]$ vs. ADA: $[89.23, 135.92]$). This suggest that while ADA learns much faster, both methods are capable of solving the task.

Overall, the ADA in *Multi-room with Key* setting shows a better performance in terms of sample-efficiency. In such hard exploration problem, this reduction in episode length translates directly into faster acquisition of competent behavior and more efficient use of experience.

5.5.2 Episode Reward Analysis

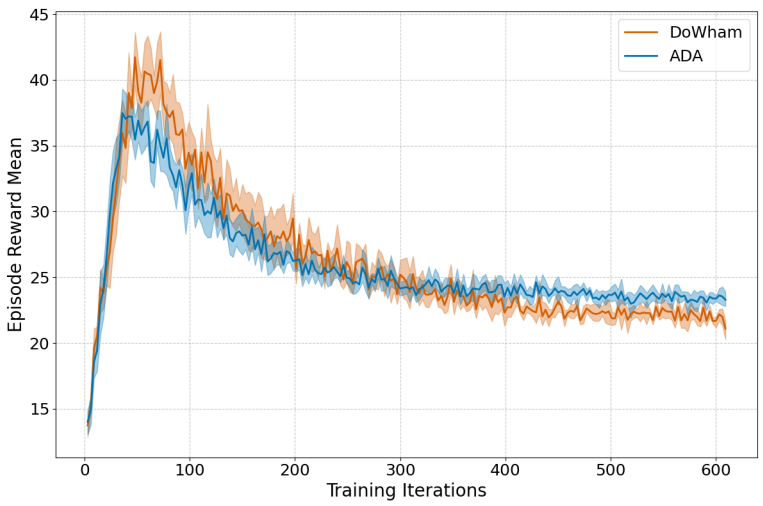


Figure 22. The performance of the proposed method ADA compared to DoWhaM in `Multi-room with Key` environment in terms of episode reward.

In episode reward perspective both DoWhaM and ADA are starting from very similar performance levels. In very early stage, both methods have approximately same rewards and the confidence intervals largely overlap, indicating there is no strong evidence that either method is clearly superior. In next iterations, both methods

continue to improve and then reach their highest returns. The original DoWhaM reaches its highest reward at iteration 48, with a mean return of approximately 41.7 and a 95% confidence interval of $[39.6, 43.6]$. In contrast, ADA reaches its highest values earlier in this window, between iterations 36–45 with 37.18 at iteration 45. Then after this phase, both methods are started to slowly decline in terms of episode reward.Until nearly the mid training (iteration 300), DoWhaM generates high rewards than ADA which is actually consistent with longer episode lengths in this training window. This suggest that DoWhaM lingers in the environment to collect action-usefulness bonus without actually solving the task.

The stable differences emerge in the late training between iteration 540-609. For instance, at iteration 546 the original DoWhaM achieves a mean return of about 22.0 with a confidence interval of $[21.8, 22.3]$, whereas ADA reaches approximately 23.4 with an interval of $[23.0, 23.7]$. A similar pattern appears at iteration 552, where the original DoWhaM mean is around 22.4 ($[21.84, 23.07]$) and ADA reaches about 23.5 ($[23.28, 23.66]$). By the final evaluation at iteration 609, the original DoWhaM has decayed to a mean return of 21.08 ($[20.27, 21.7]$), while ADA ends at about 23.28 ($[22.8, 23.88]$). This late-stage gap is statistically significant and indicates that ADA preserves higher episodic returns when training is extended.

In summary, ADA demonstrates better long term performance. This difference in final return is consistent with the episode length analysis, where ADA achieves shorter, more efficient paths compared to DoWhaM. The original DoWhaM generates higher maximum return during the early training but its performance gradually declines. On the other hand, ADA reaches slightly lower peak values but displays more stable late training behavior which ultimately converging to consistanly higher reward in the final part of the training.

5.5.3 *Percentage Visited Analysis*

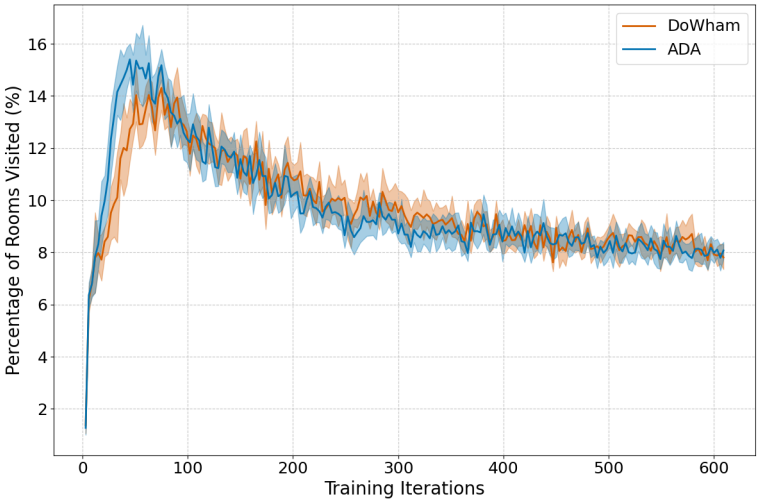


Figure 23. The performance of the proposed method ADA compared to DoWhaM in `Multi-room with Key` environment in terms of percentage visited.

In this procedurally randomized environment, where the map layout and object positions change every episode, the agent must explore sufficiently to find the key and the goal, but excessive exploration can lead to inefficiency.

ADA demonstrates a more sample-efficient exploration compared to the original DoWhaM. The ADA method reaches its peak exploration coverage of 15.40% at iteration 45. In contrast, DoWhaM reaches its peak significantly later, achieving 14.31% coverage at iteration 75. This earlier and slightly higher peak indicates that ADA motivates agent to visited and expand its frontier aggressively then gradually focuses more on goal as policy converges. By the end of the training, both agents converge to a similar level of coverage, with ADA visiting approximately 8.04% of the grid and DoWhaM visiting 7.8%. This convergence sugggest that once the task is learned, both methods are following the similar trajectories of the neccessary sequence of actions. However ADA allows agent to reach this optimal behavior with fewer total environment interactions.

5.5.4 *Conclusion: Multi-room with Key*

The overall conclusion for the `Multi-room with Key` environment is that ADA provides a clear and robust benefit over the original DoWhaM formulation in this highly challenging, dynamic environment. The combination of the action-usefulness

and spatial novelty bonuses in ADA enables the agent to discover the necessary sequence of sub-goals significantly faster and more reliably. The coverage curves show that this additional exploration components is not forcing agent to a random exploration. Early in training, ADA visits a clearly higher fraction of distinct positions than the original DoWhaM, which is consistent with the intended effect of the expansion and achievement bonuses. The agent is encouraged to reveal new parts of the layout and to move into previously unseen positions until it reliably discovers the key and the door sequence. Once a good route is established, both methods gradually reduce unnecessary visitations and converge to similar asymptotic episode lengths and rewards. Overall, the *Multi-room with Key* results indicate that the spatial extensions of DoWhaM are particularly helpful in tasks where progress depends on uncovering a sparse sequence of positional changes.

CHAPTER 6 : DISCUSSION AND CONCLUSION

6.1 *Discussion*

In this section, we discuss the main findings of the experiments across the five MiniGrid environments. The goal is to show how ADA behaves when the environments are gradually becomes more difficult to solve. To demonstrate that, we start from a simple empty environment and move towards long horizon key-door navigation tasks.

The `Empty` environment has the simplest settings. Agent only needs to move in a large open space. The results show that all three methods can solve the problem but they show different learning dynamics. Normally, simple tasks does not require a complex exploration yet ADA demonstrated significant sample-efficiency in the early training phase. It reduced the episode length much faster than both DoWhaM and the Count-based baseline. This suggest that spatial novelty components we introduced motivates agent to discover the goal trajectory earlier while encouraging agent to cover empty space more systematically. Once the optimal path is found, all methods are converged to a similar performance showing that additional spatial bonuses diminishes after the initial discovery phase.

In the `Crossing` environment, which introduces a simple wall and door that can be either vertical or horizontal, the difference becomes more clear. Since structure of the environment is changing in every episode reset, Count-Based exploration is struggling to find an optimal path and maintaining a high episode length. Both DoWhaM and ADA succesfully solves the task but ADA achieves a higher episode reward and lower episode length. Also in early training, ADA converges faster and has a lower confidence intervals meaning that combination of action-usefulness and spatial novelty in ADA helps the agent navigate through the door without getting stuck in local optima.

Static environments like `Four Rooms` and `Multi-room`, we observed a clearer distinction between the methods that prioritize rapid exploration and those that favor long-term convergence. In these settings, ADA consistently demonstrated the fastest initial learning curve while achieving a significant reduction in episode lengths within the first 30 to 50 iterations. This result confirms that the *expansion* and *achievement* bonuses successfully encourage the agent to quickly move beyond known areas and discover new parts of the map. However the Count-Based exploration proved to be

more robust for long term convergence. While it starts slower, it eventually surpassed ADA and achieved the shortest final episode lengths. This suggests that ADA serves as a strong accelerator for discovering the environment layout. However, the simple mechanism of counting visits appears more effective in fixed layouts where the state space remains constant across episodes. The performance of the original DoWhaM varied significantly between these two settings. In the `Four Rooms` environment, it showed the weakest performance, likely due to the difficulty of traversing rooms without spatial guidance. Conversely, in the `Multi-room` environment, DoWhaM proved to be robust, achieving the shortest final episode lengths and performing competitively with the Count-based baseline.

The `Multi-room with Key` environment presents the most significant challenge in this study. In this setting, the layout and object positions change in every episode. This dynamic structure prevents the agent from memorizing a fixed path or object locations. Results show that the Count-based baseline failed completely in this task. This is likely because state-visitation counts become less meaningful when the state space is constantly changing and the agent rarely sees the exact same configuration twice.

In contrast, ADA provides a clear benefit over the original DoWhaM. It discovers the necessary sequence of sub-goals significantly faster. Furthermore ADA maintains a higher episodic return in the final stages of training, where the performance of the original DoWhaM declines. This suggests that in dynamic environments where the agent cannot rely on memorized positions, the spatial bonuses of ADA provide a generalized heuristic for exploration. The ability to distinguish between *seen* and *visited* positions appears to be particularly important when the agent must actively search for objects in a procedurally generated maze.

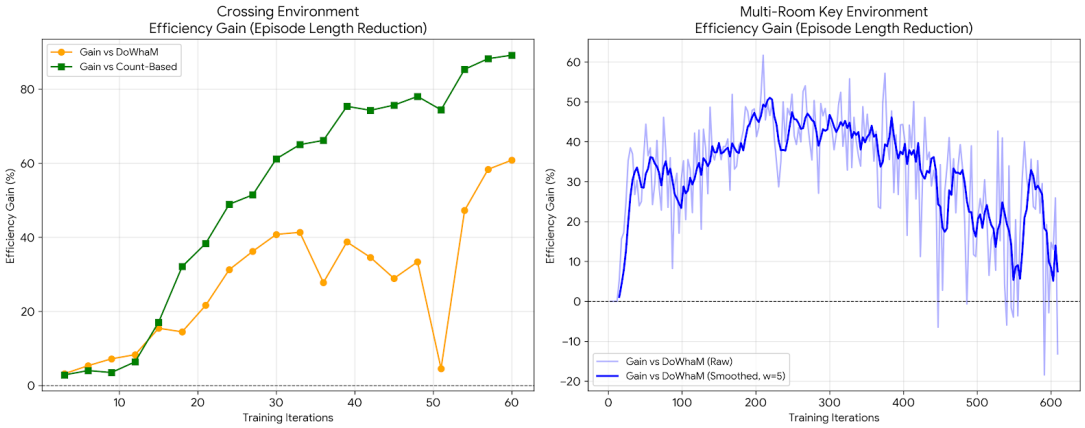


Figure 24. Sample efficiency gains of the proposed ADA method in procedurally generated environments.A positive efficiency gain indicates that ADA produces shorter episodes than the baseline at that training iteration

The advantage of the proposed method is visible at most in procedurally generated environments, such as `the Multi-room with Key` (see 5.5) and `Crossing` (see 5.2), as shown in the Figure 24, where the layout and object positions change in every episode. In such cases, the Count-Based baseline fails since state-visitation statistics become less meaningful under constantly changing configurations, ADA provides a clear benefit over DoWhaM by discovering the required sub-goals faster and maintaining stronger performance in later training.

Across the experiments, the results indicate that ADA behaves as a sample-efficient exploration mechanism. It consistently achieves shorter episode lengths and lower confidence intervals in every environment, particularly in the early stages of training. This suggests that ADA acts as a clear accelerator compared to the baselines. In static environments, the Count-based method can eventually achieve a more optimal policy when given sufficient time. However, in environments where the agent cannot rely solely on memorizing the optimal path, ADA shows better performance than both the Count-based method and its predecessor, DoWhaM.

This implies that the combination of action-effectiveness and spatial novelty provides a robust heuristic for navigation when the agent cannot rely on memorized state-visitation counts. Therefore, ADA is best suited for tasks where the environment structure is complex or dynamic, and where minimizing the number of interactions required to learn a policy is a priority.

6.2 *Limitations*

In this study, we demonstrated that ADA improves sample-efficiency in dynamic exploration tasks. However, these results come with several limitations that should be considered.

This work only evaluates the method performance in Minigrid (Chevalier-Boisvert et al., 2023) environment. Minigrid provides a 2D grid-world with discrete action spaces. The proposed method uses two helper functions to map observation to hash and extract visible states from partial observation using the current position and direction of an agent which are provided by the environment in every step. In MiniGrid, these functions can be defined cleanly because the environment exposes grid coordinates. Even we demonstrate the sample-efficiency in 2D grid world, we cannot generalize this to a complex 3D environments. In high-dimensional visual space, observation hashing, defining *visited* and *unseed* positions can be challenging and may require additional work. The implementation of spatial bonuses relies on these heuristics. In different observation conditions, like first person 3D views, calculating these positions accurately might be computationally expensive or noisy.

6.3 *Future Work*

There are several direction that this work can be extended. In the thesis, we focused on grid-based environment with discrete action spaces. A natural next step is to try ADA on high-dimensional 3D environment such as VizDoom (Wydmuch et al., 2019), Atari (Machado et al., 2018) or Mujoco (Todorov et al., 2012). In such environments, defining *unseen* positions is not straightforward so that further investigation need for such domains.

For all experiments, we used PPO (Section 2.5) for our underlying learning algorithm. The effectiveness of the methods can vary with different algorithms or optimization methods. We can further analyze the results with distributed algorithms like IMPALA (Importance Weighted Actor-Learner Architecture) (Espeholt et al., 2018). IMPALA is designed to generate high-throughput data. In our experiments, computational constraints are limited by the hardware, so utilizing such parallelism is not an option.

Also the hyper-parameters (see Section 4.4) we used in this work is same accross all experiments. This makes the comparison between Count-Based exploration, DoWhaM, and ADA more controlled. Even-though this shows the robustness of the method, with additional hyper-parameter search might improve the method's performance further. Future work is needed to analyze the sensitivity of the spatial bonuses to these parameters.

Another important area to cover to prove the robustness of the method is investigating ADA performance in a phenomenon known as *noisy-tv* problem. In our experiment setup, the environment dynamics was deterministic. However, many RL algorithms often struggle to solve the task when state transitions are noisy. Future work should investigate whether the combination of action-usefulness with spatial novelty bonuses allows the agent to distinguish between meaningful progress and random noise.

Additionaly, we only evaluate the methods where agent starts to learn from scratch. Given that ADA provides better sample-efficiency, it would be valuable to further investigate its performance in curriculum learning scenarios where we can *transfer learning*. Future work could analyze whether pre-trained agent with spatial bonus components can adapt simple maps to a larger or complex maps significantly faster then baselines.

6.4 *Conclusion*

In this thesis, we investigated intrinsic motivation mechanisms for sparse-reward, partially observable navigation tasks in MiniGrid. We focused on the original DoWhaM method and proposed *Area-aware DoWhaM Adaptation (ADA)*, which extends action-usefulness with two spatial novelty components. We proposed *expansion* bonus to encourage agent to reveal new areas and *achievement* bonus to move agent into previously unseen regions in grid-world. The experiments across multiple environments show that these spatial bonuses successfully push the agent beyond already known areas and accelerate exploration during early training while providing the sample-efficiency.

The results indicate that ADA behaves as a sample-efficient exploration mechanism and acts as a clear accelerator compared to DoWhaM and the Count-Based baseline, particularly in the early stages of training. In static environments, the Count-Based

method can be more robust for long-term convergence and can eventually achieve shorter final episode lengths, suggesting that simple state-counting remains effective when environment can be solved with simple state-visitation mechanisms. The performance of the original DoWhaM also depends on the environment structure, where it can be weak in layouts that require spatial guidance but can still be competitive in some multi-room setups.

Overall, this work shows that combining action-effectiveness with spatial novelty provides a more robust heuristic for exploration in complex or procedurally generated navigation tasks, especially when minimizing the number of interactions required to learn a competent policy is a priority.

REFERENCES

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P. and Zaremba, W. (2018), *Hindsight Experience Replay*.
URL: *https://arxiv.org/abs/1707.01495*

Arulkumaran, K., Deisenroth, M. P., Brundage, M. and Bharath, A. A. (2017), *Deep Reinforcement Learning: A Brief Survey*, IEEE Signal Processing Magazine 34(6), p. 26–38.
URL: *http://dx.doi.org/10.1109/MSP.2017.2743240*

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A. and Blundell, C. (2020), *Never Give Up: Learning Directed Exploration Strategies*.
URL: *https://arxiv.org/abs/2002.06038*

Barto, A. G. (2013), *Intrinsic motivation and reinforcement learning*, *in* 'Intrinsically Motivated Learning in Natural and Artificial Systems', Springer, pp. 17–47.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D. and Munos, R. (2016), *Unifying Count-Based Exploration and Intrinsic Motivation*, NIPS .

Bellman, R. (1957), *Dynamic Programming*, Princeton University Press, Princeton, NJ.

Burda, Y., Edwards, H., Storkey, A. and Klimov, O. (2018), *Exploration by random network distillation*, arXiv preprint arXiv:1810.12894 .

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S. and Terry, J. (2023), *Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks*, *in* 'Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)', Neural Information Processing Systems (NeurIPS).
URL: *https://github.com/Farama-Foundation/Minigrid*

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S. and Kavukcuoglu, K. (2018), *IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures*.

Eysenbach, B., Gupta, A., Ibarz, J. and Levine, S. (2018), *Diversity is All You Need: Learning Skills without a Reward Function*.
URL: *https://arxiv.org/abs/1802.06070*

Gregor, K., Rezende, D. J. and Wierstra, D. (2016), *Variational Intrinsic Control*.
URL: *https://arxiv.org/abs/1611.07507*

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z. and Gruslys, A. (2017), *Deep Q-learning from Demonstrations*.
URL: *https://arxiv.org/abs/1704.03732*

Hong, S.-J. and Lee, S.-K. (2021), *Learning Cooperative Intrinsic Motivation in Multi-Agent Reinforcement Learning*, *in* '2021 International Conference on Information and Communication Technology Convergence (ICTC)', IEEE, pp. 1697–1699.

Iqbal, S. and Sha, F. (2019), *Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning*, arXiv preprint arXiv:1905.12127 .

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P., Strouse, D., Leibo, J. Z. and De Freitas, N. (2019), *Social influence as intrinsic motivation for multi-agent deep reinforcement learning*, *in* 'International conference on machine learning', PMLR, pp. 3040–3049.

Kaelbling, L. P., Littman, M. L. and Cassandra, A. R. (1998), *Planning and acting in partially observable stochastic domains*, Artificial Intelligence 101(1), pp. 99–134.
URL: *https://www.sciencedirect.com/science/article/pii/S000437029800023X*

Kaelbling, L. P., Littman, M. L. and Moore, A. W. (1996), *Reinforcement Learning: A Survey*.
URL: *https://arxiv.org/abs/cs/9605103*

Konda, V. and Tsitsiklis, J. (1999), *Actor-Critic Algorithms*, *in* S. Solla, T. Leen and K. Müller, eds, 'Advances in Neural Information Processing Systems', Vol. 12, MIT Press.

Kulkarni, T. D. (2016), *Deep Reinforcement Learning with Temporal Abstraction and Intrinsic Motivation*, *in* 'Advances in neural information processing systems', Vol. 48, pp. 3675–3683.

Ladosz, P., Weng, L., Kim, M. and Oh, H. (2022), *Exploration in Deep Reinforcement Learning: A Survey*, arXiv preprint arXiv:2205.00824 . aDepartment of Mechanical Engineering, Ulsan National Institute of Science and Technology (UNIST), 50 UNIST-gil, Ulsan, Republic of Korea; bOpenAI LP, 3180 18th St, San Francisco, CA 94110.

Li, R., Cai, Z., Huang, T. and Zhu, W. (2021), *Anchor: The achieved goal to replace the subgoal for hierarchical reinforcement learning*, Knowledge-Based Systems 225, p. 107128.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I. and Stoica, I. (2018), *RLlib: Abstractions for Distributed Reinforcement Learning*, *in* 'International Conference on Machine Learning (ICML)'.
URL: *https://arxiv.org/pdf/1712.09381*

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M. J. and Bowling, M. (2018), *Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents*, Journal of Artificial Intelligence Research 61, pp. 523–562.

Osband, I., Blundell, C., Pritzel, A. and Roy, B. V. (2016), *Deep Exploration via Bootstrapped DQN*.
URL: *https://arxiv.org/abs/1602.04621*

Ostrovski, G., Bellemare, M. G., Van Den Oord, A. and Munos, R. (2017), *Count-based exploration with neural density models*, *in* 'ICML 2017', Vol. 6, JMLR. org, pp. 4161–4175.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019), *PyTorch: An Imperative Style, High-Performance Deep Learning Library*.
URL: *https://arxiv.org/abs/1912.01703*

Pathak, D., Agrawal, P., Efros, A. A. and Darrell, T. (2017), *Curiosity-driven Exploration by Self-supervised Prediction*.
URL: *https://arxiv.org/abs/1705.05363*

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T. and Gelly, S. (2019), *Episodic Curiosity through Reachability*.
URL: *https://arxiv.org/abs/1810.02274*

Schulman, J., Levine, S., Moritz, P., Jordan, M. I. and Abbeel, P. (2015), *Trust Region Policy Optimization*, CoRR abs/1502.05477.
URL: *http://arxiv.org/abs/1502.05477*

Schulman, J., Moritz, P., Levine, S., Jordan, M. and Abbeel, P. (2018), *High-Dimensional Continuous Control Using Generalized Advantage Estimation*.
URL: *https://arxiv.org/abs/1506.02438*

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017), *Proximal Policy Optimization Algorithms*.
URL: *https://arxiv.org/abs/1707.06347*

Sequeira, P., Melo, F. S. and Paiva, A. (2011), *Emotion-based intrinsic motivation for reinforcement learning agents*, *in* 'Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)', Vol. 6974 LNCS, Springer, pp. 326–336.

Seurin, M., Strub, F., Preux, P. and Pietquin, O. (2021), *Don't do what doesn't matter: Intrinsic motivation with action usefulness*, arXiv preprint arXiv:2105.09992 .

Singh, S., Lewis, R. L., Barto, A. G. and Sorg, J. (2010), *Intrinsically motivated reinforcement learning: An evolutionary perspective*, IEEE Trans. on Auto. Mental Dev. 2(2), pp. 70–82.

Sutton, R. S. (1988), *Learning to predict by the methods of temporal differences*, Machine Learning 3(1), pp. 9–44.

Sutton, R. S. and Barto, A. G. (1998), *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning, MIT Press.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., Turck, F. D. and Abbeel, P. (2017), *Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning*.
URL: *https://arxiv.org/abs/1611.04717*

Tibshirani, R. J. and Efron, B. (1993), *An introduction to the bootstrap*, Monographs on statistics and applied probability 57(1), pp. 1–436.

Todorov, E., Erez, T. and Tassa, Y. (2012), *MuJoCo: A physics engine for model-based control*, *in* '2012 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 5026–5033.

Watkins, C. (1989), Learning from Delayed Rewards, Ph.d. thesis, Cambridge University.

Wu, Z., Liang, E., Luo, M., Mika, S., Gonzalez, J. E. and Stoica, I. (2021), *RLlib Flow: Distributed Reinforcement Learning is a Dataflow Problem*, *in* 'Conference on Neural Information Processing Systems (NeurIPS)'.
URL: *https://proceedings.neurips.cc/paper/2021/file/2bce32ed409f5ebcee2a7b417ad9beed-Paper.pdf*

Wydmuch, M., Kempka, M. and Jaśkowski, W. (2019), *ViZDoom Competitions: Playing Doom from Pixels*, IEEE Transactions on Games 11(3), pp. 248–259. The 2022 IEEE Transactions on Games Outstanding Paper Award.