Policy Optimisation and Generalisation for Reinforcement Learning Agents in Sparse Reward Navigation Environments

by

Asad Jeewa

Submitted to the School of Mathematics, Statistics and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

at the

UNIVERSITY OF KWAZULU-NATAL

January 2021

Author .....
School of Mathematics, Statistics and Computer Science
January 2021

Certified by .....
Mr Anban W. Pillay
Lecturer
Thesis Supervisor

Certified by .....
Dr Edgar Jembere
Lecturer
Thesis Supervisor

Abstract

Sparse reward environments are prevalent in the real world and training reinforcement learning agents in them remains a substantial challenge. Two particularly pertinent problems in these environments are policy optimisation and policy generalisation. This work is focused on the navigation task in which agents learn to navigate past obstacles to distant targets and are rewarded on completion of the task. A novel compound reward function, Directed Curiosity, a weighted sum of curiosity-driven exploration and distance-based shaped rewards is presented. The technique allowed for faster convergence and enabled agents to gain more rewards than agents trained with the distance-based shaped rewards or curiosity alone. However, it resulted in policies that were highly optimised for the specific environment that the agents were trained on, and therefore did not generalise well to unseen environments. A training curriculum was designed for this purpose and resulted in the transfer of knowledge, when using the policy "as-is", to unseen testing environments. It also eliminated the need for additional reward shaping and was shown to converge faster than curiosity-based agents. Combining curiosity with the curriculum provided no meaningful benefits and exhibited inferior policy generalisation.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Mr Anban Pillay, for his continuous support, guidance and invaluable feedback throughout the completion of this research, as well as my co-supervisor, Dr Edgar Jembere, for advising and guiding me whenever needed. I would also like to thank my family for supporting and inspiring me during this journey, as well as God for being a source of peace and solace throughout my life.

Sincere thanks goes to the the Council for Scientific and Industrial Research (CSIR) and the Department of Science and Innovation for funding this research, as well the Centre for High Performance Computing (CHPC), for providing access to their hardware to run these experiments. Furthermore, various members of the Centre for Artificial Intelligence Research (CAIR) have given me additional feedback and direction that guided this work.

Preface

The research contained in this dissertation was completed by the candidate while based in the Discipline of Computer Science, School of Mathematics, Statistics and Computer Science of the College of Agriculture, Engineering and Science, University of KwaZulu-Natal, Westville, South Africa. The research was financially supported by the Council for Scientific and Industrial Research (CSIR) and the Department of Science and Innovation (DSI).

The contents of this work have not been submitted in any form to another university and, except where the work of others is acknowledged in the text, the results reported are due to investigations by the candidate.

Declaration

Declaration: Plagiarism

I, Asad Jeewa, declare that:

(i) the research reported in this dissertation, except where otherwise indicated or acknowledged, is my original work;

(ii) this dissertation has not been submitted in full or in part for any degree or examination to any other university;

(iii) this dissertation does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons;

(iv) this dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a) their words have been re-written but the general information attributed to them has been referenced;

b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced;

(v) where I have used material for which publications followed, I have indicated in detail my role in the work;

(vi) this dissertation is primarily a collection of material, prepared by myself, published as journal articles or presented as a poster and oral presentations at conferences. In some cases, additional material has been included;

(vii) this dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

Declaration: Publications

1. Asad Jeewa, Anban Pillay, and Edgar Jembere. Learning to Generalise in Sparse Reward Navigation Environments. In Aurona Gerber, editor,*Artificial Intelligence Research*, volume 1342 of *Communications in Computer and Information Science*, pages 85–100, ISBN 978-3-030-66150-2, Springer International Publishing, 2020.

2. Asad Jeewa, Anban Pillay, and Edgar Jembere. Directed curiosity-driven exploration in hard exploration, sparse reward environments. In Marelie H. Davel and Etienne Barnard, editors, *Proceedings of the South African Forum for Artificial Intelligence Research, Cape Town, South Africa, 4-6 December, 2019*, volume 2540 of *CEUR Workshop Proceedings*, pages 12–24. CEUR-WS.org, 2019.

3. Asad Jeewa, Anban Pillay, and Edgar Jembere. Directed Curiosity in Sparse Rewards Environments. Poster presented at: Postgraduate Research and Innovation Symposium; 2019 Oct 17; Durban, South Africa

4. Asad Jeewa, Anban Pillay, and Edgar Jembere. Leveraging Demonstrations in Sparse Rewards Environments Poster presented at: Deep Learning Indaba; 2019 Aug 25-30; Nairobi, Kenya

5. Asad Jeewa, Anban Pillay, and Edgar Jembere. Robust Adversarial Inverse Reinforcement Learning. Poster presented at: Deep Learning IndabaX South Africa; 2019 Apr 14-17; Durban, South Africa

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Sparse reward environments are prevalent in the real world and training reinforcement learning agents in them remains a substantial challenge [3]. This work investigated two particularly pertinent problems for developing reinforcement learning agents in sparse reward environments: policy optimisation and policy generalization within the domain of navigation. The work studied curiosity based exploration for policy optimisation and curriculum learning for policy generalisation. We report on combining curiosity-based intrinsic rewards with distance-based shaped extrinsic rewards for learning optimal policies and on a hand-crafted curriculum that was designed for policy generalisation. This chapter provides a brief background to reinforcement learning with a particular focus on sparse reward navigation environments. The problem statement and aims and objectives are detailed.

1.1 Background

1.1.1 Reinforcement Learning

In reinforcement learning (RL), an agent learns how to complete a task or act optimally within an environment by maximising rewards that it obtains from the environment. The agent is able to observe the state of the environment and choose actions from a set of actions. Taking actions modifies the environment and results in rewards

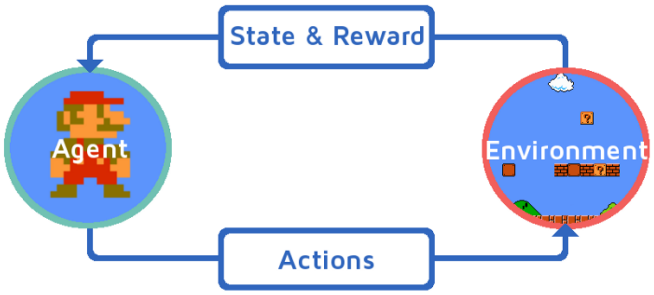


Figure 1-1: Reinforcement learning loop

that serve as positive or negative feedback. The agent seeks to maximise its rewards by learning to make better choices of actions. The choice of actions, as a function of environment states, is referred to as a policy. The agent initially selects actions randomly but learns to adapt its policy by maximising long-term rewards. This process is referred to as the Reinforcement Learning Loop, depicted in Figure 1-1.

The policy can either be learned directly, by optimising the policy parameters to maximise long-term cumulative rewards, or instead, a value function can be learned that represents the expected returns (cumulative rewards) from any given state (when acting under a particular policy) [74]. In this case, the policy is implicitly learned i.e. it is executed by selecting the action with the highest value, from a given state. A hybrid approach is also possible, referred to as Actor-Critic, and was used in this study (detailed in Section 4.2).

Reinforcement learning problems are usually framed as Markov decision processes (MDPs). MDPs are decision frameworks defined by a tuple:

$$(S, A, \mathcal{T}, r, \gamma) \tag{1.1}$$

where:

- S, A are finite sets of environment states and agent actions respectively.
- $\mathcal{T}(s'|a, s)$ is the transition distribution i.e. the dynamics of the environment which is generally defined as the probability $P(s'|a, s)$ where s' is the next state and s and a are the current state and action respectively.

- $r(s, a)$ is the reward function that returns a scalar value given the environment state and action taken by the agent (it is also possible for the reward function to be defined as a function of state only). Since the agent learns desirable behaviour solely through the reward function, it needs to be carefully engineered to associate correct behaviour with maximum rewards.

- $\gamma \in (0, 1)$ is discount factor representing the difference in importance between present and future rewards. If $(\gamma = 0)$, it means the agent is short-sighted and cares only about immediate rewards and if $(\gamma = 1)$, the agent is far-sighted and equally values all future rewards. The policy is learned to maximise the expected discounted rewards.

There are multiple variations on the standard MDP. This work focuses on Partially Observable MDPs (POMDP), where the agent is not able to fully observe the entire environment i.e. the environment state and agent observations are not the same. A POMDP introduces an additional set O: the finite set of observations of an agent. Examples of agent observations are the coordinates of its current position or the output of a camera that allows it to view what is directly in front of it, as opposed a top down view of the entire environment in a fully observable setting.

Furthermore, this work focuses on model-free reinforcement learning where the agent learns a policy through trial and error, by interacting with the environment (and receiving rewards) instead of a model-based approach that is based on planning, given a predictive model of the environment. The transition distribution T is therefore not required.

In many environments, rewards are sparsely distributed, meaning that most timesteps (a single iteration of the RL loop from Figure 1-1) do not return any positive or negative feedback. The reward signal changes sporadically and the majority or all of an agent's actions result in no reward, except for reaching the final goal or completing the task. This is problematic since agents are often required to take a large number of consecutive actions in order to complete a task and if it does not receive any information about whether intermediate actions are good or bad, it becomes very difficult

to decide which action to take in any given state.

These environments, known as sparse reward environments [40, 69], do not provide sufficient feedback for meaningful learning to take place [58]. Sparse reward environments are prevalent in the real-world [58] and training reinforcement learning agents in them remains a challenge [3].

The sparse rewards problem is a broad problem that is well-studied in reinforcement learning. The scope of this research was restricted to navigation tasks which have wide real-world applicability and is an active research area [63].

1.1.2 Reward Engineering

1.1.2.1 Reward shaping

Reward shaping bypasses the sparse rewards problem by reshaping a reward function to ensure that the agent receives sufficient reward feedback to learn how to complete a task. This is possible through augmenting the reward signal with supplemental rewards for intermediate actions that lead to success [23] and is analogous to rewarding an agent for reaching subgoals. A reward function must not only be engineered to reflect the task at hand; it also needs to be carefully shaped [3]. However, this is a difficult process since the shaping function could alter the optimal policy and change the definition of the task [53].

Since the agent crudely optimises the reward function, it learns to exploit any loopholes that may have been unintentionally introduced. It is also common for agents to fall into local optima [78]. There are countless possible reward functions and the process of shaping rewards is often only possible through trial and error, even for experienced engineers [4, 80]. Considering the long training time for RL agents, repeatedly tweaking the reward function, after observing the behaviour of trained agents, is infeasible. It is therefore necessary to investigate the concept of "general reward shaping". In the context of navigation, this refers to the process of using general concepts such as distance measures (distance-based reward shaping) instead of optimising the specific dynamics of any specific environment.

1.1.2.2 Exploration

Closely related to the rewards problem is the issue of exploration in sparse reward environments. Exploration algorithms aim to reduce the uncertainty of an agents understanding of its environment [7]. This is important since it is not possible for an agent to act optimally until it has sufficiently explored the environment and identified all opportunities for reward [75]. Agents are unable to explore large state spaces without an intuitive exploration strategy [3] and it is likely that they will never reach a state that returns a positive reward. Hoping to stumble into a goal state by chance is futile for all but the simplest of environments [58].

ϵ-greedy is a simple exploration strategy where an agent either chooses to explore by selecting a random action with probability ϵ or to exploit the policy by choosing the best action (based on its current knowledge) with probability $1 - \epsilon$. Balancing exploration and exploitation is a common RL challenge [72].

Curiosity-driven exploration is a type of intrinsic reward that encourages an agent to explore the environment and discover "novel" states [58]. Intrinsic rewards, which are generated by the agent itself (as opposed to extrinsic rewards which are from the environment), are particularly useful when extrinsic rewards are sparse. They can be used to augment or even replace the reward function entirely.

Curiosity-driven exploration generates rewards from the error in an agent's ability to predict the consequence of its own actions [58]. During the initial stages of training, the agent explores the environment and is able to gain an understanding of the task by reaching states that return extrinsic rewards (which act as a definition of the task). Over time, the agent's curiosity decreases and it exploits the policy. The algorithm is further detailed in Section 3.1.1. Unlike ϵ-greedy, a curiosity agent explicitly learns exploratory behaviour that enables it to learn how to complete a task.

An environment is referred to as a "hard exploration" environment when rudimentary exploration strategies, such as ϵ-greedy, are insufficient [7] i.e. the probability of reaching a goal state is negligible and more complex exploration strategies such as curiosity-driven exploration are required.

1.1.3 Policy Generalisation

Another fundamental challenge in RL is that of generalisation [13]. It is customary to use the same environments for both training and testing [14]. This means that agents often exhibit breakthrough results on very specific tasks but fail to generalise beyond the training environment [57], even when only slight changes are made [84, 87]. The agents often memorise sequences of actions that lead to success [87] and the notion that agents posses a deep understanding of their environments is often misguided [83]. Policy memorisation refers to a policy that optimises the dynamics of a particular environment by memorising actions that lead to success, resulting in poor performance when changes are made to the environment [84].

Generalisation in RL broadly refers to the capacity of an agent to perform "well" or transfer knowledge to related environments [23]. Furthermore, it refers to learning policies that are robust to environment variations [57] and do not overfit to any particular training environment [83].

This work defined generalisation in sparse reward navigation environments as the ability of agents to navigate to targets in environments that they were not trained on i.e. the task remained the same but the dynamics of the environment were changed by reconfiguring the obstacles. In order to succeed, an agent needs to learn the "skill" of finding a target rather then memorising the optimal path to the target in a specific training environment.

Moreover, policy generalisation relates specifically to the extent to which a policy transfers to unseen environments. In this work, we placed the added condition of evaluating policy generalisation without any additional training or fine-tuning to specific environments (referred to as zero-shot semantic navigation in [49]). This requires an agent to learn a robust policy that enables it to find a distant target in an arbitrary navigation environment. This is a difficult task: it is somewhat counter intuitive since the policy cannot optimise to any specific environments that the agents encounter during training. Furthermore, if the agent learns to optimise for every training environment (which is highly unlikely when there is a large number of train-

ing environments), it is almost certain that overfitting has occurred, resulting in poor performance in unseen environments.

1.1.3.1 Curriculum Learning

This work evaluated curriculum learning as a means to improve policy generalisation. A curriculum may be used to enable agents to learn difficult tasks [51] that are not possible to learn directly [51, 73].

Curriculum Learning refers to the process of training an agent using a learning curriculum that varies the difficulty of the task over time, thereby allowing it to gradually increase its knowledge, by learning how to complete easier versions of the task. The curriculum introduces structure to the training process and is largely the manner in which humans learn [29]: for example, a student first needs to learn basic arithmetic before advancing to algebra and then to calculus.

In this work we considered designing a curriculum to act as a means of bypassing the sparse rewards problem (an environment can be made smaller so that it is possible for the agent to reach the target and gain reward feedback to update its policy) and also to improve policy generalisation by exposing an agent to a diverse set of environments during training [13, 87] thereby deterring it from memorising the dynamics of any particular training environment.

1.2 Problem Statement

It is difficult for agents to learn a task when the reward function is sparse but the process of shaping or augmenting the reward signal is not straightforward [53, 81]. Within the domain of navigation, distance-based reward shaping is a means of increasing the reward feedback but this approach causes agents to get stuck in local optima [21, 78]. Curiosity-driven exploration is an intrinsic reward that has been successfully applied to sparse reward environments in previous studies [11, 58] but while the intrinsic reward function enables agents to explore the environment, it does not incorporate any explicit information about the task. This work therefore investigated

combining curiosity with a distance-based shaped reward function to enable agents to intelligently explore environments i.e. exploring the environment while always keeping the main goal or target in mind. We term this novel compound reward function as *Directed Curiosity* and investigated its effectiveness for policy optimisation in sparse reward navigation environments.

Furthermore, RL agents often learn polices that are optimised to the environments that they were trained on, but fail to generalise beyond the training environment even when subtle changes are made [57]. Instead of learning a single optimal path, it is more desirable for agents to learn a robust policy that enables them to navigate to targets in arbitrary environments. This is a difficult task to learn directly. RL agents often memorise sequences of actions that lead to success during training [87]. This work investigated the effectiveness of curriculum learning as a means of improving generalisation as well as bypassing the sparse rewards problem in navigation tasks.

1.3 Aims and Objectives

This research had two intersecting aims. Firstly, we wished to determine the efficacy of curiosity-driven exploration as a means to enable intelligent exploration and exploitation of sparse reward navigation environments. The second aim was to design and evaluate curricula to enable agents to navigate to targets in previously unseen testing environments.

The specific objectives of this study are:

- To develop a custom suite of sparse reward navigation environments for training and testing.
- To design and implement a curiosity-driven exploration algorithm.
- To determine empirically the effectiveness of curiosity-driven exploration for intelligent exploration and learning optimal polices in sparse reward navigation environments.
- To design and implement a curriculum for learning navigation tasks.

- To evaluate the efficacy of curriculum learning as a means to improve policy generalisation in sparse reward navigation environments.

1.4 Methods

A custom suite of sparse reward navigation environments was designed. In each environment, an agent was required to learn to navigate from its starting point, past obstacles, to a distant target in the shortest possible time (see Section 4.1). The environments were carefully designed to incorporate various high-level features such as dead-ends, bottlenecks and multiple paths to the target. The environments were used in two sets of experiments, one for policy optimisation and the other for policy generalisation.

In the policy optimisation experiments, we evaluated the efficacy of a novel compound reward function, *Directed Curiosity*, against extrinsic reward shaping (Algorithm 1 (Chapter 3) presents a distance-based shaped reward function) and intrinsic rewards, specifically curiosity-driven exploration from [58]. The performance of the agents was evaluated in a sample of five different navigation environments,as detailed in Section 4.3, where agents were trained to learn the shortest path to the target within that particular environment.

In the generalisation experiments, we designed and evaluated a curriculum for sparse reward navigation environments (see Algorithm 3), by analysing its ability to learn policies that generalise to unseen environments, against agents trained with curiosity and a hybrid approach that combined the curriculum with curiosity. The suite of navigation environments was therefore carefully divided into a set of training and testing environments. Details of the experiments are given in Section 4.4.

1.5 Impact and Contributions

This work presents an empirical analysis of reward engineering in sparse reward navigation environments. A novel reward function, *Directed Curiosity*, that combines

curiosity with a distance-based shaped reward function is presented. We show empirically that it enabled agents to learn more robust policies more quickly. A critical evaluation of both reward shaping and curiosity-driven exploration [58] is also presented.

Since *Directed Curiosity* was deliberately restricted to general reward shaping principles in the form of distance measures, the reward function can be fine-tuned and applied to sparse reward tasks in other domains.

This work also presents a study of curriculum learning as a policy generalisation mechanism. A custom training curriculum is presented that resulted in an increased transfer of knowledge to unseen testing environments than a curiosity-based approach and a hybrid approach that combined curiosity with the curriculum. A benefit of this approach is that it does not require any manual reward shaping. The curriculum was designed in a way that can be applied to other RL tasks in different domains, with minimal tweaking.

1.6 Structure of Dissertation

The remainder of this dissertation is structured as follows: Chapter 2 introduces previous related work. The algorithms: *Directed Curiosity* and the manually designed training curriculum is presented in Chapter 3. The details of implementation, pertaining to the learning environments, hyperparameter optimisation and experimental design are contained in Chapter 4. Chapter 5 presents the results of both the optimisation and generalisation experiments, as well a detailed discussion into the strength and limitations of the presented algorithms. Chapter 6 provides concluding remarks and proposes future work. Lastly, additional results (Appendix A) and two published papers (Appendix B, C) are included as appendices at the end of the dissertation.

Chapter 2

Literature Review

This work focused on two problems in sparse reward navigation environments, viz. policy optimisation and generalisation. This chapter discusses previous work in the field. Two established approaches for reinforcement learning in sparse reward environments were investigated: extrinsic reward shaping and intrinsic rewards. Furthermore, we explored the use of curriculum learning as a means of enabling agents to learn difficult tasks, with particularly attention to generalisation in the navigation domain. Previous work on the generalisation problem in RL is then discussed.

2.1 Sparse Rewards

The sparse rewards problem is well-studied in reinforcement learning as it applies to a wide array of RL tasks. Sparse reward functions [50] are better suited to goal-oriented tasks that are defined through binary reward functions that indicate whether an agent has completed a task [2]. This is particularly common in the field of robotics [50]. Sparse rewards are also often observed in video game environments, which are commonly used to evaluate RL algorithms [3, 7, 32]. Additionally, the prevalence of sparse reward environments in the real world [58, 64] has motivated various novel approaches to the problem [3, 50]

2.1.1 Reward Shaping

2.1.1.1 Approaches to Reward Shaping

Reward shaping aims to ensure that agents receive sufficient reward feedback in order to learn how to complete a task optimally. It is a means of introducing prior knowledge to reduce the number of sub-optimal actions [15] and guides the learning process [46]. A common approach is to introduce supplemental rewards for intermediate actions that lead to success [23] which is analogous to setting sub-goals or smaller targets for the agent.

Similarly, the reward function can be designed to include additional rewards that relate to different aspects of the task. For example, Lample et al. introduced additional rewards to enable agents to learn how to play a first-person shooter game [45]. A reward function that rewards agents for a kill and penalises them for dying defines the desired behaviour that agents need to learn but is too sparse and was therefore augmented with additional rewards for picking up objects and penalties for losing health and ammunition.

An alternate approach to reward shaping is to manipulate the reward function to gradually increase rewards as the agent gets "closer" to completing its task, instead of an abrupt change from 0 to 1 when the task is completed. This approach requires the notion of distance or progress to be defined for the task [59]. Our work focused on this particular approach. However, a common issue with distance-based reward functions is that they are prone to local optima [21, 78], as illustrated in a toy environment (see Figure 2-1) defined in [78] and in navigation environments in [35]. This was observed with a distance-based shaped reward function that penalised agents with the euclidean distance between the goal and the agent on every timestep in [21, 78].

Trott et al. proposed a novel approach that learns an auxiliary reward shaping function for avoiding local optima [78]. The experiments were performed in maze environments and also explored combining shaped rewards with additional reward signal for balancing exploration and exploitation. However the reward function was designed to estimate local optima and encourage agents to avoid them, whereas our

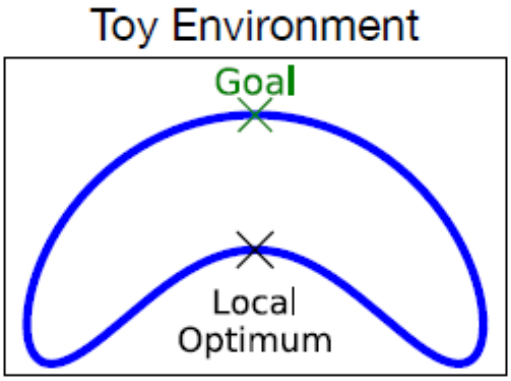


Figure 2-1: Local Optimum problem with rudimentary distance-based shaped reward functions

work combined the shaped rewards with curiosity-driven exploration [58] since it was previously used as a means of enabling agents to escape local optima in [88]. In a further attempt to circumvent the issue of local optima, we defined and evaluated multiple different distance-based reward functions and assessed their strengths and weaknesses.

The process of manually shaping rewards is difficult and is often only possible through trial and error, even for experienced engineers [4, 80]. Moreover, bespoke functions are often required for each new environment or task [15, 34]. The benefit of the distance-based shaped reward function in this work (see Algorithm 1) is that it is based on general principles only (distance metrics) and the shaping process does not require configuration for each environment or any domain expertise.

An alternate approach is implicit reward shaping, that learns from demonstrations of target behaviour. A potential-based reward function was recovered from demonstrations, using state similarity in [68], through inverse reinforcement learning methods in [69] and directly from raw pixel data in [34].

2.1.1.2 Preserving the Optimal Policy

Reward shaping can have a detrimental effect on training and change the optimal policy or the definition of the task if the rewards are not carefully shaped [15, 53].

This is because the agent does not learn the task directly but rather through a proxy function (the shaped reward function). Furthermore, since the sole purpose of RL algorithms is to maximise cumulative long-term rewards, when the reward feedback is too dense, the agent learns to act in a very specific and often undesired manner [46]. Moreover, a poorly shaped reward can introduce new local optima that prevents agents from learning optimal behaviour [78].

Policy invariance is therefore required, whereby changing the reward function does not change the optimal policy [53]. Potential-Based Reward Shaping was proven to preserve the optimal policy of a task [15, 53] by defining the reward function in the form:

$$F(s, s') = \gamma\phi(s') - \phi(s) \tag{2.1}$$

ϕ is a reward function over states that introduces "artificial" shaped reward feedback [5]. F is referred to as a potential function which is the difference between ϕ of the next state s' and the current state s with γ a discount factor on $\phi(s')$. The reward function in Algorithm 1 is of this form since it is based on the relative distance to the goal i.e. the change in distance from one state to the next.

Various studies presented extensions of potential-based reward shaping since the restriction on the form of the reward function limits its expressiveness [46]. Potential-Based Advice is a function defined over both states and actions [82], while a novel Bayesian approach that augments the reward distribution with prior beliefs was presented in [46].

2.1.2 Intrinsic Rewards

An alternative to "shaping" an extrinsic reward is to supplement it with intrinsic rewards [56], which are rewards that the agent generates on its own as opposed to extrinsic rewards that are from the environment. This work focuses on intrinsic rewards specifically pertaining to exploration.

2.1.2.1 Curiosity-based Rewards

One such intrinsic reward is curiosity: Curiosity-Driven Exploration by Self-Supervised Prediction [58] formally defined a framework for training curious agents. The algorithm is further detailed in Section 3.1.1. It empowers agents with the capability of exploration by rewarding them for seeking "novel" states. This enables agents to reach far away states that contain extrinsic rewards and explore environments more efficiently. Pathak et al. evaluated curiosity on navigation tasks in [58] using the *VizDoom* environment [43]: the agents were able to learn to navigate to distant targets when the curiosity reward was combined with a sparse reward function (+1 for finding the target). However, when the distance between the agent and the target was made sufficiently large (thereby increasing the sparsity in rewards), the agents only succeeded on 66% of the training runs. We extended this work by combining the curiosity with a dense distance-based shaped reward in order to encourage agents to explore intelligently, since the dense rewards act as a definition of the task.

Much research has built upon the findings of this paper [58]. Large scale analysis of the approach was performed in [11] where agents learned to play various Atari Games without any extrinsic rewards.

Various studies pointed out limitations in curiosity. If the task is too difficult, and the agent is unable to reach new environment states, the policy deteriorates since the agent receives neither intrinsic or extrinsic rewards [58]. Burda et al [11] highlighted the noisy-tv problem: whereby a tv that samples random images was added to a maze and the agent was equipped with the ability to change channels. The agent learned to maximise its curiosity by continuously changing channels and was never able to learn to navigate to the target. This is also refereed to as the "couch-potato" effect whereby an agent learns to game the system by finding a source of randomness in an environment and by continuously observing it, it is able to satiate its curiosity and loses its motivation to keep exploring [11, 64]. This is analogous to getting stuck in a local optimum. In order to circumvent this issue, Savinov et al. used of a memory buffer to stores experiences and then rewarded agents for "finding" states that are not

already in memory [64].

Previous studies combined extrinsic and intrinsic rewards for navigation [49, 88]. The work in [88] is closely related to our work. The authors explored the use of curiosity-driven exploration for two dimensional navigation in hand-crafted mazes and computed a reward function as a weighted sum of both intrinsic and extrinsic rewards. A custom extrinsic reward function was defined that was also distance-based, with additional penalties for colliding with obstacles as well as not making any progress towards the goal and being incorrectly orientated. We argue that this reward function is heavily fine-tuned for the specific task. Our work instead used a more general reward function that was restricted to be solely based on distance metrics.

The application of the study was also different in that it was focused on robotics. Similar to our work, the agents were not given any information about the dynamics of the environment and relied on ray observations but in order to simulate a robotics platform, the agents were equipped with vastly more rays that were also significantly longer. The action space was also different as the agents were able to change their orientation by rotating themselves.

The study highlighted the benefit of entropy loss for exploration i.e. by increasing the degree of randomness in the agents' action choices, it avoids falling into local optima and is able to explore the environment before converging to an optimal policy. This was also observed in our experiments (refer to Section 4.3.2 and Section 4.4.2). It also emphasised the importance of intrinsic rewards for improving performance in tasks with challenging exploration requirements [88].

2.1.2.2 Alternatives to Curiosity

There are various similar approaches to curiosity-driven exploitation: Variational Information Maximizing Exploration (VIME) [33] maximises information gain, as opposed to minimising prediction error in [58], by encouraging agents to take actions that result in states that they deem surprising i.e. states that cause large updates to the dynamics model distribution. However, curiosity-driven exploration was shown

to perform better in [58], both in terms of convergence rate and accuracy.

Classic work in [10, 42] investigated balancing exploration and exploitation in polynomial time and has inspired research in the area of intelligent exploration. Count-based exploration methods generate an exploration-bonus from state visitation counts [75] by encouraging agents to visit states that were rarely visited before [64]. Exploration bonuses encourage an agent to explore, even when the environment's reward is sparse [7], by optimising a reward function that is the sum of the extrinsic reward and an exploration bonus. It is a simple and effective approach that has achieved promising results, notably on the notoriously difficult "Montezuma's Revenge" Atari game in [7, 8].

Various studies have attempted to extend count-based approaches to large state spaces [25, 55]. Count-based methods naturally lend itself nicely to discrete observation spaces but its extension to continuous observation spaces is non-trivial [64]. Bellemare et al. presented an observation density model for this purpose [7], which was extended in [55]. Alternatively, a hash functions was used to discretise the observation space in [75] While the major benefit of the approach is its simplicity, it does not perform well in visually rich 3D environments [58, 64].

It was shown in multiple studies [25, 58, 64] that curiosity-driven exploration as defined in [58], is superior to similar methods [25, 33, 75] and is regarded as the state of the art. It was therefore chosen as the intrinsic reward function for all the experiments in this study.

There are various other novel approaches for exploration. These include *Random Network Distillation* (RND) [12], which defines an exploration bonus as the loss when predicting a random function of the environment states. RND achieved state of the art performance on various sparse reward Atari games. The study highlighted the difficulties of combining extrinsic and intrinsic rewards and modified the popular policy gradient method PPO (also used in our experiments) by separating a value function into an extrinsic and intrinsic stream.

Furthermore, exploration was learned through maximising empowerment in [28], wherein the long-term goal of the agent is to maximise its control on the environ-

ment, by using demonstration data to learn an exploration policy in [70, 50] and by encouraging agents to behave "diversely" and learn skills without reward functions in [19].

2.2 Curriculum Learning

Curriculum learning is widely used in RL [27]. It imposes an order on training to enable an agent to gradually increase its knowledge, eventually learning to perform difficult tasks [29], that are infeasible to learn directly [73]. Fundamental work by Bengio et al. formally defined the concept of curriculum learning in a machine learning context and performed an empirical investigation under a supervised learning framework [9]. The authors claimed that the major benefits of using a curriculum are faster convergence, since the learner wastes less time on examples that are too challenging, and improved generalisation [41], by guiding training towards better regions in the parameter space. Curriculum learning was also used as a means of bypassing the sparse rewards problem in various studies [60], as is the case in our work.

2.2.1 Approaches to Curriculum Learning

There are multiple approaches to curriculum learning. A common approach is to split a particularly complex task into smaller, easier-to-solve sub-problems [27], ensuring that the current task is of intermediate difficulty i.e. be neither too easy (already solved) nor too difficult (unsolvable), thereby maximising the learning process [60]. Manually controlling this difficulty is not straightforward. A better approach is self-paced curriculum, proposed by Jiang et al. in [36], which enables the learner to reject examples that it regards as too difficult i.e. the curriculum updates at the pace of the learner.

Initial work in the RL domain manually defined a curriculum by presenting learners with tasks from a sequence of predetermined subtasks [41] or through continuously searching for the simplest unsolvable problem [65]. Narvekar et al. defined the problem as transferring knowledge learned from easier source tasks to more difficult target

tasks [52]. The knowledge can be transferred in different ways such as through the policy [71], value function [52] or reward function [73]. The former approach (transferring polices) was used for this work since the task remains consistent throughout training, meaning that the agent can build upon the knowledge learned from easier mazes by updating an existing policy.

2.2.2 Automatic Curriculum Learning

Designing a curriculum is a difficult task since it is analogous to teaching which is not straightforward even for humans [9]. Furthermore, manual curriculum design often requires expert domain knowledge [73]. Various studies attempted to alleviate this problem by automatically generating a curriculum [73, 21, 22, 47].

Automatic Curriculum Learning (ACL) refers to either automatically defining the order with which to learn tasks [73], automatically generating tasks for the learner [21, 47] or continuously increasing the difficulty of the task instead of multiple discrete sub-tasks [6].

The learning process does not necessarily have to be sequential: Svetlik et al. presented a means of generating a curriculum as an acyclic graph so that multiple tasks could be learned in parallel. Another common approach is that of a teacher and student i.e. "the teacher proposes a task, and the student learns to do it.", as seen in [47] where learners were able to revisit previously learned tasks that they may have "forgotten" and in [71] which proposed *Asymmetric Self-Play*, a commonly used method that presents a teacher who proposes a task through a set of steps which the learner must reproduce. The approach was shown to be prone to local optima in [22].

Asymmetric self-play pertains to the automatic generation of a curriculum for exploring an environment. Similarly, curiosity-driven exploration [58] can be considered as automatically devising a learning curriculum to guide the exploration of the state space [60]. This is because the reward function is adapted as a function of the learning trajectory of the agent [60], i.e. the reward function changes as the agents explores the environment and the prediction error of the next state decreases.

ACL has exhibited promising results when applied to navigation [22, 49, 71]. The

work of Florensa et al. is of particular interest where the authors defined a framework for learning in "reverse" by attempting to reach a goal state made increasingly further away or more difficult to reach [22]. The maze environments from that work was incorporated into our experiments. Moreover, the work inspired our curriculum since by introducing more obstacles and increasing the size of the environment, the goal state is made more difficult to reach. However, our work focused on manually designing a curriculum rather than automatically generating one and in an attempt to bypass previous difficulties with manual curriculum design, we restricted the curriculum to only general concepts (environment size and obstacle configuration). Designing the curriculum therefore did not require significant fine-tuning and expert knowledge since the difficulty thresholds (points at which the curriculum advances and the task gets more difficult) were defined as a function of the environment size so that each value did not need to be tuned separately. The work in [49] built upon this study by training agents to find objects in the real world and the authors argued that the approach is more efficient than those in [22, 71].

2.2.3 Hierarchical Reinforcement Learning

Curriculum Learning is closely related to hierarchical reinforcement learning, which decomposes a RL problem into a hierarchy of sub-problems such that higher-level tasks invoke lower-level tasks as primitive actions i.e. actions can be reused in more difficult tasks [31]. The actions are less granular than traditional RL problems for example, a reusable action would be "open door" as opposed to "grasp handle", "rotate hand" etc.Ṫessler et al [77] presented a framework that enabled agents to reuse high-level actions (termed as "skills"), learned from easier sub-tasks, in difficult tasks requiring multiple skills [76]. This enabled agents to learn complex navigations tasks faster since they could reuse the skill of navigating to a new room. Similarly, Frans et al [24] taught agents high-level actions that pertained to walking and moving which allowed them to learn difficult navigation tasks faster. Though our work did not pertain to hierarchies directly, our curriculum was designed to implicitly learn in this manner. Since there were no obstacles in the early stages of training, the agents were

able to focus on learning to control themselves to move around the environment and this knowledge was reused when obstacles were introduced and the environment was made more difficult.

2.3 Generalisation

Generalisation remains a fundamental RL problem since agents tend to memorise trajectories from their training environments [87] instead of learning transferable skills [13] and classic RL benchmarks like the *Arcade Learning Environment* (ALE) [8] focus on creating specialist agents that perform well in a single environment. Various new benchmarks have therefore been proposed to focus research in the direction of generalisation.

2.3.1 Procedural Generation

Generalisation often translates to evaluating the performance of agents across different levels of the same game [13, 54]. The *ProcGen* Benchmark [13] uses procedural generation to generate new environments for 16 unique games. The inherent diversity in the generated environments demands that agents learn robust polices in order to succeed. *ObstacleTower* is a similar benchmark that was presented in [38]. ProcGen presents a large number heterogeneous environments while ObstacleTower is a single complex environment with higher visual fidelity as it is based on the Unity game engine [37]

Justesen et al. expanded on this concept by showing how adaptive difficulty can improve both sample efficiency and generalisation [39]. They also highlighted limitations of procedural generation: it is difficult to automatically scale the difficulty of the task [39] and the distribution of the procedurally generated environments is often different to that of human-generated environments meaning that agents struggle to generalise to the human-generated environments [39]. The work also noted that procedurally generating environments may lead to overfitting to the distribution of the generated environments. A novel approach that uses reinforcement learning to

learn how to generate environments shows promising results in [44]. In this work, all the environments were manually generated. This allowed us to carefully study the merits of different algorithms. Future work would entail performing larger scale analysis by procedurally generating environments and further evaluating the robustness of polices.

2.3.2 Guidelines for Generalisation

Savinov et al. emphasised the need for separate training and testing environments when training RL agents and also investigated generalisation in the navigation framework [63]. These findings were also highlighted in [87]. Zhang et al. proposed "detecting" overfitting during training by continuously providing the agent with feedback on its generalisation performance by computing its score on the testing environments during training. This feedback can be used to adapt the training process [86]. The analysis of the work was performed in simple environments inspired from the classic Gridworld task [72], while our study was performed in a diverse set of maze navigation environments. Ye at al. [84] hypothesised that generalisation suffers due to the lack of input representation in training and showed how simply rotating and translating input observations can improve generalisation results. Furthermore, they highlighted the importance of introducing a large number of different training environments so that the agent cannot simply memorize a sequence of actions that lead to success in a specific environment [84]. The findings of the aforementioned work, where relevant, were incorporated into our curriculum i.e. having a diverse set of both training and testing environments.

2.3.3 Generalisation Capabilities of Existing Approaches

Curiosity-driven exploration was shown to have strong generalisation capabilities in previous studies [58, 88]. Pathak et al. showed that agents trained on one level of a Mario game were able to perform relatively well in another level. However, the policy needed to be fine-tuned and the agents struggled to generalise to environments with

different textures in [11, 58]. This finding is not particularly relevant to our work since agents observations were vectors rather than visual representations.

Zhelo et al. also used curiosity for generalisation (in maze environments) by combining it with shaped extrinsic rewards [88]. The work also pointed out the difficulty of assigning weights to the reward components, as was also pointed out in our experiments. The findings from these studies motivated us to use curiosity as a baseline algorithm in the policy generalisation experiments detailed in Section 4.3.

An alternative approach to curiosity is reachability, which rewards agents for new states defined by how difficult they are to reach. This approach was shown to achieve better generalisation than curiosity on procedurally generated mazes [64]. We therefore wish to investigate this approach further in future work.

Portelas et al. highlighted various studies that utilised a curriculum for improving generalisation [60]. Of particular interest is a method for automatic curriculum learning tailored to the navigation task in [49], where agents learned policies robust to environment changes. However the generalisation was evaluated with respect to changing the visual appearance of the target in both simulated and real word environments. This work rather focuses on generalisation by changing the obstacle configuration of environments.

While reward shaping has enabled agents to learn difficult tasks in various studies, they emphasise learning specialist polices [69] and often do not focus on generalisation [13]. This is because the shaped rewards act as a definition of a specific task. This result was also observed in our experiments. A major benefit of our curriculum is that it does not require reward shaping.

There are various other novel algorithms for generalisation in RL. Relational Deep Reinforcement Learning blends the generalization power of inductive logic programming with reinforcement learning to enable agents to succeed in more complex versions of the training task [85]. Progressive neural networks ensure that agents do not "forget" previously learned knowledge [62] and an embedding space was utilised to learn reusable skills, particularity focused on robotics. in [30]. Alternatively, Farebrother et al. highlighted that commonly used RL algorithms like DQN suffer from poor gen-

eralisation and simple approaches like dropout and regularisation, commonly used in supervised learning, can be used to improve generalisation [20].

2.4 Summary of the Literature Review

The goal of this literature review was to investigate existing approaches for both policy optimisation and generalisation in sparse reward environments as well as to identify existing issues and guidelines. The sparse rewards problem is well-studied since many tasks are naturally defined in this manner [50] and due to its real-world relevance [64]. We explored two reward engineering approaches to the problem: extrinsic reward shaping and intrinsic rewards. The review highlighted various difficulties with manual reward-shaping, particular that distance-based reward shaping is prone to local optima [21, 78]. Intrinsic rewards is a fruitful area of research: this review focused primarily on intrinsic rewards related to exploration. Curiosity was shown to outperform various similar algorithms, improve generalisation [88] and also enable policies to escape local optima [88] and has therefore been incorporated into this work (as a combined reward function with distance-based reward shaping).

Curriculum learning was used in various studies to enable agents to learn difficult tasks and also as a means of improving generalisation [41]. However, designing a curriculum is difficult and often requires expert knowledge [73]. A common solution to this is Automatic Curriculum Learning but in this work, we opted rather for manually designing the curriculum but restricted the curriculum to be based only on general concepts, not optimising it for any particular environment. Furthermore, the review highlighted that most state of the art RL algorithms memorise trajectories [87] and suffer from poor generalisation. Various guidelines were identified and incorporated into our experiments such as crafting separate sets of training and testing environments that are both diverse and comprehensive [84]. We have also identified various avenues for future work such as using procedural generation to generate both training and testing environments.

Chapter 3

Algorithms for Policy Optimisation and Generalisation

This research investigated policy optimisation and generalisation in sparse reward navigation environments. Agents were placed in two-dimensional environments that contained obstacles blocking the path to a fixed target. The goal was to learn to navigate to the target in the shortest possible time.

This chapter presents two algorithms: a novel reward function (Directed Curiosity) which is a combination of distance-based reward shaping and curiosity-driven exploration and a manually-designed curriculum for enabling agents to find targets in unseen environments and improve generalisation.

3.1 Policy Optimisation: Directed Curiosity

Directed Curiosity engineers a hybrid reward function composed of extrinsic and intrinsic rewards. The intrinsic reward, curiosity-driven exploration, is from [58] and equips the agent with an intelligent exploration strategy that enables it to reach distant states in sparse reward environments. Instead of combining the curiosity reward with sparse extrinsic rewards as in [58], we defined a distance-based shaped reward function to ensure that the agent constantly receives feedback about its goal. We theorise that this hybrid approach enables agents to explore in a more directed

manner.

The reward function is a weighted sum of curiosity and distance-based reward shaping. We describe curiosity-driven exploration, as per [58] in Section 3.1.1. Section 3.1.2 details the distance-based reward function and thereafter we formally define Directed Curiosity in Section 3.1.3.

3.1.1 Curiosity-Driven Exploration

Curiosity is an intrinsic reward that empowers an agent with the capability of exploration, enabling it to reach far away states that contain extrinsic rewards. Pathak et al. [58] formally defined a framework for training curious agents that involves training two separate neural-networks: a forward and an inverse model that form an Intrinsic Curiosity Model (ICM), as depicted in Figure 3-1. The inverse model encodes the current and next observations s_t and s_{t+1} respectively, into a feature space ϕ and learns to predict the action $\hat{a}_t$ that was taken between the occurrence of the two encoded observations. The forward model is trained to predict the next encoded observation $\phi(s_{t+1})$ from the current encoded observation $\phi(s_t)$ and action a_t.

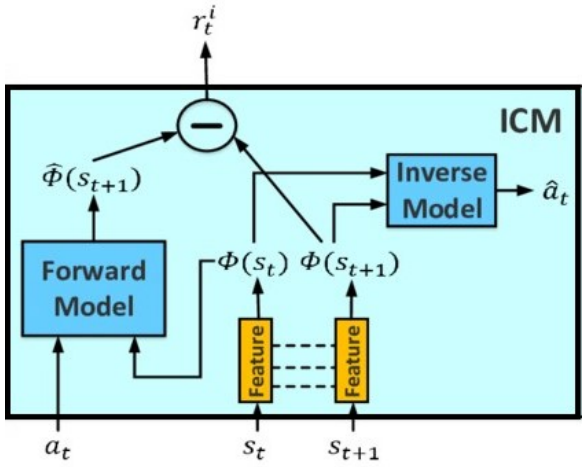


Figure 3-1: The Intrinsic Curiosity Module

In order to generate a curiosity reward signal, the inverse and forward dynamics models' loss functions are jointly optimised i.e. curiosity is defined as the difference between the predicted feature vector of the next state $\hat{\phi}(s_{t+1})$ (from the forward

model) and the real feature vector $\phi(s_{t+1})$. η is a scaling factor in Equation 3.1.

$$r_i^t = \frac{\eta}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2 \tag{3.1}$$

As an agent explores, it learns more about its environment. The prediction error therefore decreases over time and the agent becomes less curious. A major benefit of this approach is that it is robust: through jointly optimising the forward and inverse model, the reward captures surprising states that have come about only directly as a result of the agents actions. By encoding the observations i.e. mapping the pixels into a feature space, the agent is not "distracted" by predicting unnecessary information: if the agent cannot interact with something, it will not observe it.

The intrinsic reward can be used as the sole reward signal, or it can be combined with an extrinsic reward signal that is often sparse [58]. Figure 3-2 from [58] defines how the reward signal is generated at every timestep: the agent starts in state s_t and samples an action a_t from its current policy π, which places it in a new state s_{t+1}. At this point, the environment returns an extrinsic reward r_e^t and the ICM (Figure 3-2) calculates the intrinsic reward r_i^t from (s_t, s_{t+1}, a_t). The policy updates by optimising the sum of the two rewards i.e. $r_e^t + r_i^t$. This process continues until some termination criteria is met.

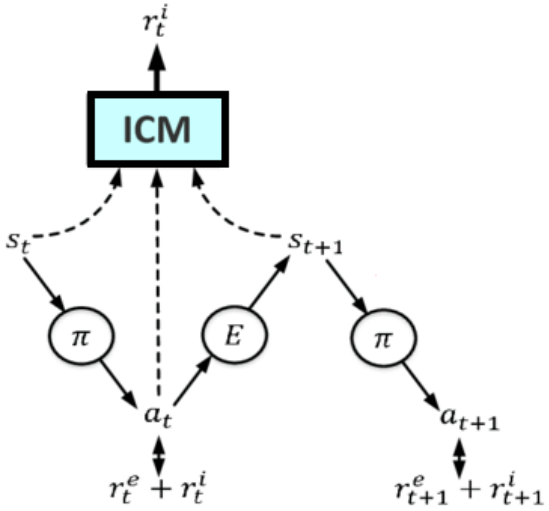


Figure 3-2: Generating an intrinsic reward using the Intrinsic Curiosity Module.

3.1.2 Distance-Based Reward Shaping

The distance-based reward function described here is combined with the intrinsic reward (curiosity) detailed in the previous section.

Reward shaping is the process of reshaping a sparse reward function to ensure that the agent receives sufficient reward feedback to learn a policy. If the agent only receives positive reinforcement for completing a task i.e. navigating to a target, it becomes difficult to judge which intermediate actions are good ones. This is problematic when the environment is sufficiently large.

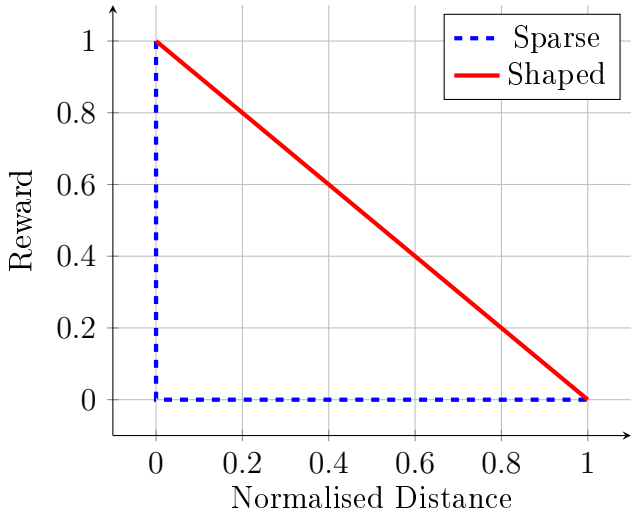


Figure 3-3: Shaping a sparse reward function.

Figure 3-3 illustrates the effect of reward shaping: the sparse reward function returns no feedback to the agent except for when it reaches the target. Alternatively, the shaped reward function returns dense feedback (on every timestep) that is directly proportional to the distance to the target, enabling the agent to understand which intermediate actions take it closer to the target and which take it further away. The distance is normalised such that 1 represents the maximum possible distance away from the target.

However, shaping rewards is a fragile process since small changes in the reward function results in significant changes to the learned policy [81]. The agent can easily fall into a local optimum or learn to game the system [79]. This phenomenon is re-

ferred to as "The Cobra Effect" wherein an attempted solution to the problem actually makes the problem worse i.e. reward shaping introduces unintended consequences and "pollutes" the agents motivations. This is because the agent's goal is solely to maximise long-term cumulative rewards. The issue is often caused by "over-engineering" the reward function: the reward function should not define exactly how to complete a task but rather characteristics or features of "good" and "bad" behaviour to enable the agent to learn how to complete a task on its own, through interacting with the environment.

In order to circumvent the aforementioned issues, we restricted the rewards to general principles only i.e. the rewards were shaped with distance metrics only. The function was not optimised to any particular environment and engineering the reward did not require any domain expertise. The basic principle is to reward an agent for moving closer to the target and penalise it for moving further away.

Furthermore, a comprehensive analysis of various reward functions was performed: we trained agents using different reward functions and then observed their behaviour to highlight limitations and instances where the agents learned to game the system. By tuning the function, to mitigate these limitations, we were able to incrementally optimise the reward function.

The following notation is used for defining reward functions:

- R refers to the terminal reward (for finding the target).
- P refers to the terminal penalty.
- r is the shaped reward in a single timestep.
- p is the shaped penalty in a single timestep.

By definition, $R, r > 0$ and $P, p < 0$. Note that the terminal penalty P in this work was returned when agents fell off the platform, since some environments had no walls along the boundary.

The simplest reward function returned a constant positive reward r on every timestep that resulted in the agent moving closer to the target. To ensure that

the agent's primary goal is to find the target, the terminal reward R needs to be significantly larger then r. The agent was also penalised with a small existential penalty p, on every timestep, to encourage it to find the target as quickly as possible.

The issue with this function is that the positive and negative rewards are not balanced i.e. the agent receives either excess rewards or penalties, resulting in a dominant policy being learned. This can controlled by tuning r and p, though this is not a straightforward process: when r is too high the agent learns to game the system by delaying reaching the target (to gain more positive rewards in an episode). The means that agent is no longer incentivised to find the target in the shortest possible time and its motivations are polluted. On the other hand, if p is too high, the agent is not encouraged to move closer to the target and it learns to game the system by prematurely ending the episode by falling off the platform. Even though it still receives a negative return for the episode, the penalty for falling is less than the total negative rewards it would have received for an entire episode and the agent therefore chooses to take the "lesser" punishment. This points to the importance of not only tuning r and p, but also tuning the maximum number of episode steps. This behaviour is an example of shaped rewards altering the optimal policy of the original task [53].

Another consideration is that at any point in time, the agent should not receive more positive rewards for moving closer to the target, or more negative rewards for moving further away, so as not to introduce loopholes for the agent to exploit. This is possible through returning a constant positive reward r for moving closer to the target and a constant penalty p for moving further away (or staying still) but while this alleviates the dominance issue, the function does not consider the effect of consecutive actions. If an agent moves closer on one timestep, and then moves closer on the next timestep, it should receive a higher reward on the subsequent timestep since it is closer to finding the target and in a more desirable state. The same applies for moving further away from the target.

This can be implemented by introducing two counter variables: one that is incremented for every consecutive positive step and another that incremented for every

consecutive negative step. The count can be reset to 0 when the agent changes direction. The issue with this reward function is that when the environment is sufficiently large, the shaped rewards spiral out of control. This is because the episodic shaped reward becomes significantly larger than the terminal reward. This effect can be dampened by assigning significantly smaller values to p and r but this does not alleviate the problem entirely.

In order to avoid these issues, we propose the following conditions be placed on the reward function. For every episode:

$$r * M < R \tag{3.2a}$$

$$p * M > -R \tag{3.2b}$$

$$R \geq |P| \tag{3.2c}$$

where M refers to the maximum number of timesteps in a single episode, R is the terminal reward for finding the target and P is the penalty for falling off the platform.

These inequalities ensure that the shaped reward function does not alter the motivations of the agent by keeping its primary objective as navigating to the target: Equation 3.2a ensures that the sum of all shaped reward feedback never exceeds that of the terminal reward (Equation 3.2b serves the same purpose for penalties). Additionally, Equation 3.2c expresses that the terminal reward must always dominate the terminal penalty i.e. while it is important to avoid falling off the platform, the primary goal is finding the target. The inequalities are defined for single objective reinforcement learning agents though they can be modified for multi-objective systems.

The inequalities were satisfied by defining a shaped reward function based on the relative distance between target and agent (Algorithm 1).

There are various benefits to Algorithm 1. The agent is penalised if it stays still and the shaped reward signal can be controlled using the reward coefficient C. This ensures that the episodic shaped rewards can be capped such that they never exceed terminal positive reward. Furthermore, the multiplier can be used to normalise the rewards based on the maximum number of steps, thereby ensuring that the maximum

Algorithm 1 A distance-based shaped reward function

Input: Agent position P_{agent}, target position P_{target}, maximum distance D_{max}[1], previous distance D_{prev}, reward coefficient C
Calculate $D_{current} \leftarrow \text{distance}(P_{agent}, P_{target})$
Calculate reward signal: $R \leftarrow D_{current}/D_{max}$
if $D_{current} < D_{prev}$ **then**
  return $C \cdot (1 - R)$
else
  return $C \cdot (-R)$
end if

possible shaped rewards in an episode never exceeds the positive terminal reward, as per Equation 3.2a.

Another benefit of Algorithm 1 is that there is a balance between positive and negative rewards since they are both relative to the change in distance. The agent receives the highest reward when it moves closest to the target and the highest penalty when it moves furthest away. This means that the shaped reward function is policy invariant i.e. it does not alter the goal of the agent to learn the optimal path to the target.

Since the rewards are shaped exclusively based on distance metrics that do not take into account the specific dynamics of the environment, the same function can be used for different environments, and in general, for navigation tasks. A limitation of this approach is that the target location needs to be known beforehand. However, this is a deliberate decision since these experiments aimed to asses whether agents can learn optimal paths to distant targets with limited information i.e. only its current position and fixed destination. If the location is unknown, the definition of the task changes from a navigation-based one to a goal-finding or search task. This is not within the scope of this study but rather a possible area for future work.

A limitation of the shaped reward function is that it is a greedy approach, since it rewards or penalises agents on every timestep. The approach encourages an agent to navigate to the target in the shortest possible time but it does not incorporate foresight. It is necessary for an agent to take "backwards" steps in order to bypass

[1] D_{max} refers to the largest possible distance between the agent and the target, within the environment and was introduced to normalise the reward.

obstacles that block it paths to the target.

We investigated various modifications to the reward function that aimed to introduce foresight in agents and encourage them to act less simplistically. This is possible through decreasing the density of the shaped reward feedback by calculating the change in relative position not on every timestep, but rather after fixed intervals of n steps. The rationale here is that the agent should still move closer to the target but not on every timestep since it is necessary to take "backward" steps to bypass obstacles that block its direct path to the target.

However, the behaviour of agents was observed to be almost the same and this approach therefore was not able to bypass the limitation. The types of environments where agents failed to navigate to the target are detailed in Section 5.1.2. Furthermore, assigning a value to n is a difficult task: if n is too small, the an agents acts in an overly simplistic manner attempting to move closer to the target on every timestep and when it is too large, the reward feedback becomes insufficient and does not result in any meaningful learning.

An alternate approach is to define environment checkpoints that act as subgoals and return smaller rewards. However, it is tedious and inefficient to define checkpoints manually for each environment. A simple automatic checkpoint generator was therefore implemented but this is not directly within the scope of this research and its evaluation is left for future work.

Instead of introducing foresight directly into the reward function, we chose to combine the reward with a curiosity signal. We argue that this encourages agents to implicitly behave in a more flexible manner, since they are encouraged to explore the environment, while still attempting to find the target in the shortest possible time.

3.1.3 Combining Intrinsic and Extrinsic Rewards

We propose a novel algorithm that combines the curiosity-based intrinsic reward function from [58] and the distance-based shaped reward function defined in Algorithm 1, for training agents to navigate to distant targets in sparse reward navigation environments. Training agents with either curiosity or the shaped reward results in limited

behaviour: due to the nature of the shaped reward function, the agent struggles to navigate past obstacles to find the target. This is because the shaped reward function has been designed to encourage agents to continuously move closer to the target. Using curiosity only may cause the agent to spend too much time exploring the environment, even after the target has been found. This is exacerbated if there are no extrinsic rewards that act as a definition of the task. Further details of the limitations of the two approaches are discussed in Section 5.1. For example, it was often observed that agents tended to get "trapped" in suboptimal states or local optima.

Combining the two approaches allows the agent to explore the environment while always keeping in mind its goal of finding an optimal path to the target. Specifically, curiosity enables the agent to learn about the dynamics of the environment through exploration. Over time, the agent learns where the obstacles are placed and uses this knowledge to learn how to navigate past them. It is also able to learn about the dimensions and scale of the environment. By behaving in this manner, curiosity enables the agent to ultimately find the target in the environment.

The shaped rewards from the environment ensure that the agent is informed about whether it is taking steps that bring it closer to the target or further away i.e. it is always knowledgeable about its goal. This feedback enables the agent to learn a path to the goal. It is important to distinguish that the goal of the agent is not to find the target, but rather to learn a optimal path to the target.

In general, this combined approach enables the agent to learn in a more directed and intuitive manner.

Algorithm 2 Directed curiosity-driven exploration

Input: Initial policy π_0, extrinsic reward weighting w_e, intrinsic reward weighting w_i, max steps T, decision frequency D

for $i \leftarrow 0$ to T **do**

Run policy π_i for D timesteps

Calculate distance-based shaped reward r_e^t with Algorithm 1

Calculate intrinsic reward r_i^t with Equation 3.1

Compute total rewards $r_t = w_i \cdot r_i^t + w_e \cdot r_e^t$

Take policy step from π_i to π_{i+1} with reward function r_t

end for

Directed Curiosity (Algorithm 2) therefore simultaneously maximises two reward signals. Since, the reward function components are somewhat conflicting, it is essential to tune the strength of each signal in order to find a balance of curiosity and goal-driven behaviour. The agent needs sufficient time to explore the environment and should not converge to a suboptimal policy too quickly. This is similar to the exploration vs exploitation problem in RL. The rewards can be balanced by tuning the weights w_e and w_i, attached to both the constituent reward signals in Algorithm 2.

The policy is updated using any optimisation/ policy gradient method.

3.2 Policy Generalisation

Instead of learning a policy that is optimised for a single training environment, it is more desirable for an agent to learn a policy that generalises to unseen environments i.e. it enables an agent to navigate to distant targets across multiple spare reward navigation environments that were not encountered during training.

We present a curriculum that was manually designed in order to improve policy generalisation in sparse reward navigation environments.

3.2.1 A Curriculum for Policy Generalisation

The task of learning a policy that generalises to unseen environments is a difficult one since the policy cannot optimise to any specific environments that the agents encounter during training. Furthermore, when the environments are large, with multiple obstacles, the reward feedback is too sparse to enable any meaningful learning.

Since it is not possible to learn the task directly, a curriculum can be used to control the difficulty of the task. We therefore designed a curriculum to act as a means of bypassing the sparse rewards problem and also to improve generalisation by exposing agents to a diverse set of environments during training. The curriculum is applicable to navigation tasks and can be fine-tuned depending on the specifics of the task at hand.

Algorithm 3 varies environment parameters over time to control the difficulty of

Algorithm 3 A manually-designed curriculum for navigation

Input: Single obstacle environments O, obstacle max scale $S_{obstacle}$, maze environments M, environment max scale $S_{environment}$, reward threshold $R_{threshold}$, number consecutive episodes $n_{consecutive}$

```
for i ← 1 to S_environment do
    Reset episode count
    r_average = 0
    repeat (for each episode)
        r_average ← average episodic reward from previous n_consecutive episodes
        Sample an obstacle environment from O
        Sample scale from {0, 1, 2, ..., S_obstacle}
        Sample angle from {0°, 45°, 90°, 135°, ..., 315°}
        Sample agent and target starting positions
    until r_average < R_threshold
    Reset episode count
    r_average = 0
    repeat (for each episode)
        r_average ← average episodic reward from previous n_consecutive episodes
        Sample a maze environment from M
        Sample agent and target starting positions
    until r_average < R_threshold
end for
```

the task, so that the current task is never too difficult for the agent. The first parameter is the environment size ($S_{environment}$): decreasing the size, while keeping the agent size and speed the same, decreases the sparsity of rewards since the goal and target are closer to each other in smaller environments. The second parameter is the obstacle configuration, which is varied through changing the number and size of obstacles by sampling environments that contain either a single obstacle (or none at all) from O or sampling environments with multiple obstacles in a maze-like structure from M. The curriculum jointly varies both these parameters over time to progressively make it more difficult for an agent to reach the target. In the early stages of training, the environments are small and contain a single obstacle or none at all, that varies in terms of its orientation and scale (bounded by $S_{obstacle}$). This is the easiest version of the task. The agent is able to learn how to control itself and navigate around the environment to nearby targets. This is an implicit means of bypassing the sparse rewards problem since the agent receives dense rewards since it is relatively easy to

navigate to the target.

When the reward reaches a predefined threshold $R_{threshold}$, the first adjustment is to increase the size and number of obstacles. Thereafter, the environment is made larger only when the agent is able to consistently find targets in all the environments of the same size. This two-fold difficulty adjustment keeps occurring until the agent progresses to a large environment with multiple maze-like obstacles. The agent needs to exhibit an "understanding" of its task which is possible through setting n, the number of consecutive episodes that the average reward exceeds a reward threshold, to be sufficiently large.

The curriculum aims to ensure that the agent is always "optimally challenged" [37]. A single policy is updated since the objective of the agent remains the same throughout training (to navigate past obstacles to a target or destination) i.e. knowledge learned from easier tasks are transferred to more difficult tasks through the policy, instead of learning new tasks from scratch, with a randomly initialised policy. The tasks progress in a sequential manner i.e. an agent cannot learn a new task until it has sufficiently learned its current one [51].

Inspired by various other work [40, 63, 87], the last aspect of the curriculum attempts to bypass the sparse rewards problem by "densifying" the training environment. The starting locations of both the agent and target are randomised at the start of every episode. This means that the target is often close to the agent, resulting in frequent feedback that enables meaningful learning and also encourages the agent to explore all parts of the environments.

The curriculum aims to improve generalisation though randomly sampling from a wide array of training environments i.e. the set of training environments (O and M in Algorithm 3) must be diverse and incorporate a wide array of obstacle configurations [13] to deter agents from memorising the dynamics of any particular training environment. This is analogous to supervised learning i.e. training on a diverse training set allows for a more generalised model that does not overfit to training data. Here, overfitting specifically refers to a policy that memorises action sequences for the training environments, resulting in poor performance in the testing environments.

Chapter 4

Methods

4.1 The Navigation Task

This work investigated the sparse rewards problem within the domain of navigation. A suite of custom navigation environments was created for experimentation that allowed for extensive analysis of different algorithms and enabled the control of environment parameters

The baseline task is as follows: an agent is placed in a two-dimensional environment with a target that it must learn to navigate to. The environments contain obstacles that block the agents direct path to the target. The goal of the agent is to learn to navigate from its starting point to the distant target in the shortest possible time.

This is similar to the classic Gridworld task [72] that presents a rectangular grid: the cells of the grid correspond to the states of the environment and at each cell, four actions are possible. The agent starts in a predefined start state and must learn to navigate to a predefined target state, upon which a positive reward is returned. There are also additional terminal states that an agent must learn to avoid (analogous to traps) which return a negative rewards to the agent.

Various amendments were made in order to increase the difficulty of the task. The environment was made significantly larger and multiple environments were created by introducing different configurations of obstacles. Furthermore, an agent is able to

move in eight directions (diagonally) in these experiments.

The task is a representation of navigation tasks wherein an agent only receives positive feedback upon reaching its destination. This is a common theme in RL for goal-oriented tasks where an agent receives sparse, binary, success-failure feedback indicating whether an intended goal has been accomplished [2]. The task is also a variation of the classic point-mass navigation task from various studies [17, 21, 26].

Various measures were taken to ensure that the custom environments were sparse reward environments, as controlled through the size of the environment, relative to the agent and target, and the agent speed. These parameters were carefully tuned to ensure that agents that were trained to optimise a sparse reward function (+1 for finding the target) with an arbitrary policy gradient method (PPO) and no exploration strategy, were never able to find the target, even after a large number of training steps [35]. This is depicted in Section 5.1 where the "sparse reward" agents are shown to never converge to an optimal policy, when trained for the same number of timesteps as all of the other algorithms. Furthermore, we also observed the agents after training and noted that they were unable to make meaningful progress towards the goal.

The maximum environment size was defined through the maximum ratio of the agent to the environment size (floor coverage) and was fixed at $1:100$. The agent and the target were the same size and all environments were square environments i.e. the length and width of the environments were the same.

The agent speed was defined by the relative amount of the total environment that an agent can traverse in a single timestep, in the chosen direction. If it is too high, the probability of the agent finding the target by chance (through random movement) becomes too high and if it is too low, the task becomes too difficult and the agent may never find the target (even when equipped with an intelligent exploration algorithm). It was fine-tuned and fixed at 0.2% (the agent moves at a constant speed).

All environments that contain obstacles are "hard exploration" environments, as per [7], since the probability of agents stumbling into the goal through random $\epsilon-$greedy exploration is negligible [58]. This means that an agent requires an in-

telligent exploration algorithm in order to obtain any positive feedback, as depicted in Section 5.1.

4.2 Proximal Policy Optimisation

A standardised optimiser was required for all experiments. Policy gradient methods, which directly optimise the policy with respect to the expected return or long-term cumulative reward [74], were an important breakthrough in Reinforcement Learning [48]. However, many policy gradient methods are either sample inefficient [48] or overly complicated [66]. Proximal Policy Optimisation (PPO) [67] is a simpler, sample efficient alternative that achieved strong results across a wide-range of tasks in previous studies [37, 67]. It uses only first-order optimization, resulting in significant performance gains. Furthermore, a common issue with traditional policy gradient methods is that they are unstable due to policy updates that are too large. PPO overcomes this issue by maximising an objective function defined as:

$$L^{CLIP}(\theta) = \mathbf{E}[min(r_t(\theta)\hat{A}_{old}(s,a), clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{old}(s,a))] \tag{4.1}$$

θ denotes the policy parameter, $r(\theta) = \dfrac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$ is the probability ratio between the old and new policies: the algorithm limits the distance between θ_{old} and θ within $[1-\epsilon, 1+\epsilon]$ to stabilise the policy updates (ϵ represents a tunable hyperparameter, usually 0.1 or 0.2 [67]).

Even though PPO optimises the policy directly, it is referred to as an Actor-Critic method since it learns a value function which is in turn used to optimise the policy i.e. there are two models (neural networks): the critic, learns the value function and the actor learns the policy by updating its parameters in the direction suggested by the critic [72].

At each timestep t, the current state s_t is passed to both models. The environment returns a reward r_t which is passed to the critic to update the value function. This

is sent to the actor to update its policy. The actor then outputs an action a_t from action set A (the final layer of the actor model is a softmax on all actions) which places the agent in a new state s_{t+1} and returns a new reward r_{t+1}. The training process repeats in this manner until some termination criteria is met. The process is depicted in Figure 4-1 where the system represents the environment [74].

In order for more stable training updates, an advantage function ($\hat{A}_{old}$ in Algorithm 4.1) is calculated from the value function. This represents how good a particular action a_t is, relative to other actions. The actor therefore utilises the advantage to update its policy rather than a traditional value function.

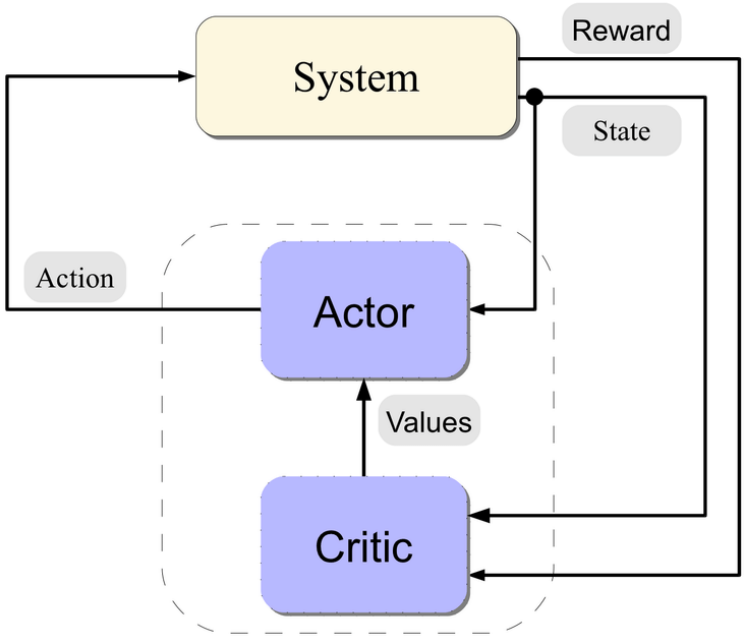


Figure 4-1: The actor-critic architecture

4.3 Policy Optimisation

The aim of the first set of experiments was to critically evaluate the novel reward function, Directed Curiosity (defined in Section 3.1.3), against agents trained with a reward function that comprises only curiosity [58], as well as agents trained with the distance-based shaped reward from Algorithm 1. This is because Directed Curiosity is a composite reward function that is computed by calculating a weighted sum of

curiosity and distance-based reward shaping. Thus, in order to assess the benefits and drawbacks of the approach, it was necessary to compare it directly to the constituent reward functions.

An episode is a sequence of interactions that ends when an agent reaches a terminal state. Thereafter, the environment is reset to an initial state and training continues.

The following conditions were placed on the navigation task: At the beginning of every episode, the agent and target were placed on opposite corners of the environment in order to make the task as difficult as possible. The target remained fixed for the entire duration of the episode.

For the policy optimisation experiments, there were no walls along the boundaries of the environments and it was therefore possible for the agent to fall off the platform (in the standard Gridworld task, any actions that moves the agent off the platform are ignored [72]). This made the task more difficult since an agent did not only have to learn to find the target but it also needed to implicitly learn to avoid falling off the platform. The environment was deliberately designed in this manner since we theorise that the agent can implicitly learn about the dimensions of the environment and about the scale of the task by falling off the platform.

An episode terminated when the agent reached the target, fell off the platform, or after a maximum number of steps and the agents position is reset randomly. The maximum number of steps was carefully tuned: it was initially set at a low value, and gradually increased through observing the training process. The steps need to be large enough to allow the agent to sufficiently explore the environment and learn about its task within a single episode, but also not too large since this unnecessary slows down training as the agent repeatedly explores the same parts of the environment.

An agent interacts with the environment in discrete timesteps. At each timestep t, it observes the environment o_t and then takes an action a_t from a set of actions A.

The observation set O of each agent included the coordinates of its current position and the coordinates of the target. Furthermore, the agent was not given any information about the dynamics of the environment such as the configuration of the obstacles and it was also not equipped with rays (that allow it to detect objects in

front of it). This makes the task significantly more difficult i.e. the environment is not fully observable and the agent is given limited information in the form of coordinates. It therefore needs to learn to find the target through exploration.

The action set A of the agent enables the following movement:

- UpDown: the agent can move up, down or choose to remain in the same position along the y-axis
- LeftRight: similarly, the agent can move left or right or choose to remain in the same position along the x-axis:

Since agents were able to select more than one action at a time, they were able to move diagonally.

The baseline reward function, before any reward shaping was introduced, was $+1$ reward for finding the target (there is no other positive reinforcement in an episode of training) and -1 for falling off the platform.

4.3.1 Policy Optimisation Environments

Five different environments were defined: in every environment the task and the agent's speed and size remained the same. The only modification was the obstacle configuration. The obstacles were carefully designed to make the task progressively more difficult so that we could identify both the strengths and weaknesses of agents. Each experiment occurred independently: an agent was trained within a single environment only to learn the shortest path to the target within that particular environment i.e. polices are not transferred between environments in this set of experiments. The five environments were:

1. SimpleNav

   This is the simplest version of the task. The agent and target were placed at opposite corners of the platform with no obstacles between them. Another simplification was to decrease the environment size by a factor of five to for an agent to floor ratio of 1:20, instead of 1:100. This environment is therefore

neither a sparse reward nor a hard exploration environment since it is possible for an agent trained with the sparse reward function (+1 for finding the target, -1 for falling off the platform) and no exploration strategy, to navigate to the target. This was therefore be referred to as an "easy exploration" environment as per [7]. The environment was used to establish baseline performance of the different algorithms (when the task is not difficult). The agent is depicted (in red) on the top left of Figure 4-2 and the target (in green) on the bottom right.

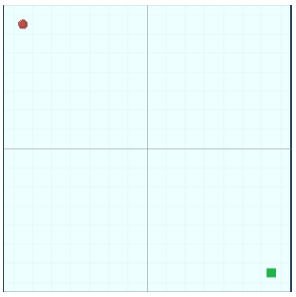


Figure 4-2: SimpleNav environment

2. DifficultNav

   For this and all proceeding environments, the agent to floor ratio was fixed at 1:100 (five times larger than *SimpleNav*). The environments are both sparse reward and hard exploration environments [7, 58]. *DifficultNav* does not contain any obstacles but increasing the size of the environment makes the task significantly more difficult than *SimpleNav*.

3. ObstacleNav

   For all remaining environments, obstacles were introduced to block the direct path to the goal and make navigating to the target more difficult. These environments were designed to test the limitations of the algorithms since an agent that acts without foresight will not be able to find the target. Rather, agents need to learn seemingly counter-intuitive behaviour, by moving further away from the target at the current timestep, in order to pass obstacles and reach the target at a later timestep.

This particular environment has a single obstacle (depicted in black in Figure 4-3) that was deliberately placed perpendicular to the optimal path to the target, forcing the agent to move around the obstacle to reach the target.

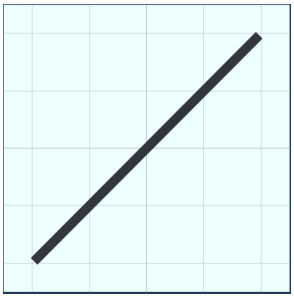


Figure 4-3: ObstacleNav environment

4. Maze1Nav

   The final two environments have multiple obstacles in a maze-like structure. In this environment, there exists only one optimal path to the target, as depicted in Figure 4-4. An agent therefore has to learn to move in a very specific manner in order to navigate to the target.

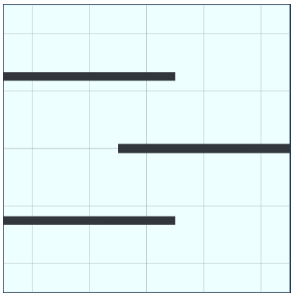


Figure 4-4: Maze1Nav environment

5. Maze2Nav

   This environment is the most difficult version of the task since it has dead-ends and multiple possible paths to the goal (see Figure 4-5). This allowed us to investigate the robustness of Directed Curiosity by analysing (a) whether agents were able to navigate to the target and (b) assessing whether they learned the optimal path. This is not a straightforward task since an agent may find

the target on one episode, but then get stuck behind obstacles as it attempts to replicate it steps.

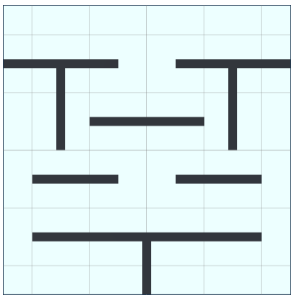


Figure 4-5: Maze2Nav environment

4.3.2 Experimental Setup and Hyperparameter Optimisation

Before training agents, the hyperparameters of each reward function were carefully tuned for each environment. This process was not only to enable agents to learn the task, but also to ensure that the comparison of reward functions was fair. An extensive literature review was performed to understand both the purpose of the different hyperparameters as well as the effect of adjusting them. This allowed for the tuning of hyperparameters in a more directed and systematic manner. We also selected literature that detailed similar experimentation, with a focus on curiosity and reward shaping in sparse reward environments, in order to obtain a set of guidelines and avoid common pitfalls.

The hyperparameters needed to be manually optimised, since the experiments occurred in custom environments. Baseline hyperparameters were identified in the simplest navigation environment, *SimpleNav*. Agents were trained with a set of hyperparameters which was then adjusted based on the resultant behaviour of agents. For example, if the agent was converging to a suboptimal policy, the learning rate was decreased, or the entropy in the system was increased to encourage the agent to take more random actions. Importantly, we took care to change only one parameter at a time, to ensure that it had the desired effect: when too many parameters are changed at once, the observed behaviour changes dramatically and it becomes very difficult to pinpoint the exact reasons for this change. These initial hyperparameters

were then systematically tuned as required, for each of the subsequent four environments, to cater for the increased complexities. An additional benefit of PPO is that it is robust and does not require significant tuning once baseline hyperparameters are identified [11].

A detailed explanation of the specific hyperparameters follows. The first set of hyperparameters are general training parameters that are common to all the algorithms:

- Maximum steps: This refers to the total number of training timesteps (in a single timestep, an observation is collected and an action is taken). This needs to be long enough to allow the agent to learn the desired behaviour but also not too long such that an agent learns to memorise paths to the goal. For each environment, the same number of timesteps was defined, regardless of the reward signal used, to ensure consistency.

- Learning rate (α): The strength of each gradient descent update step. Tuning the learning rate allows for a significantly more stable training process. For these experiments, we found that the learning was inversely proportional to the difficulty of the task i.e. a smaller learning rate was necessary for more complex tasks.

- Time horizon: The number of experiences to collect before adding it to the experience buffer. This number should be large enough to capture all the important behaviour within a sequence of an agent's actions.

- Buffer size: The number of experiences to collect before updating the policy.

- Batch size: The number of experiences in each iteration of gradient descent. The buffer size is generally a multiple of the batch size and is also significantly smaller.

The architecture of the actor and critic networks needed to be carefully tuned:

- Hidden layers: It is is necessary to include hidden layers in the neural network to allow the agents to learn complex behaviour.

- Neurons: This refers to the number of units in each of the hidden layers of the neural network. Generally, more neurons allow agents to learn more complex behaviour but increase the training time.

The environments were implemented with the Unity game engine and the agents were trained with the Unity ML-Agents Framework [37]. The framework acts as a means of abstracting the training process, however, it was possible to change the hyperparameters, as required. The swish activation function [61] was used after hidden layers since it is robust and was shown to perform better than other activation functions over a large number of RL tasks in [18, 61]. The function is defined as $f(x) = x \cdot sigmoid(\beta x)$ where β defines an arbitrary trainable hyperparameter. Both the actor and critic models have the same number architecture (hidden units and neurons).

PPO-specific hyperparameters include:

- Entropy Strength (β): Entropy refers to the "predictability" of the actions chosen by an agent: a higher entropy results in policies that are more random, thereby encouraging agents to explore the environment during training. Entropy should be high in the early stages of training and drop as the policy converges and the agent gains confidence in its actions. It was observed that varying this parameter had a significant impact on training.

- Epsilon (ϵ): This parameter defines the acceptable threshold of divergence between the old and new policies during the gradient descent update.

- Lambda (λ): Lambda influences how much an agent relies on its current knowledge when updating its policy.

- The number of epochs: This parameter is closely related to the batch size and defines the number of passes to make through the experience buffer when performing gradient descent optimization.

The final set of hyperparameters refer to the strength of the extrinsic and intrinsic reward signals (if applicable). For each reward signal, it is necessary to specify the:

- Strength: This refers to the weights attached to the reward components in Algorithm 2.
- Discount factor (γ): Gamma is a standard RL parameter and defines a discount factor for future rewards. This relates to the far-sightedness of the agent i.e. how far into the future the agent should consider possible rewards.

The baseline hyperparameters that were defined for SimpleNav are:

Table 4.1: The baseline hyperparameters for policy optimisation in *SimpleNav* environment.

Hyperparameter	Value
Maximum Steps	50000
Learning Rate α	1.0×10^{-5}
Time Horizon	64
Buffer Size	256
Batch Size	32
Hidden Layers	2
Number Neurons	128
Beta β	0.005
Epsilon ϵ	0.1
Lambda λ	0.95
Number Epochs	3
Extrinsic Reward Strength	1.0
Intrinsic Reward Strength	0.1
Extrinsic Reward Discount Factor γ	0.99
Intrinsic Reward Discount Factor γ	0.99

The baseline parameters were adjusted for each environment: the maximum training steps were increased to 250000 in *DifficultNav*, 750000 in *ObstacleNav* and 1000000 in *Maze1Nav* and *Maze2Nav*. This is to cater for the increased complexity.

Furthermore, the reward coefficient C, (from the distance-based shaped reward function in Algorithm 1), which represents the "influence" of the shaped reward, was also carefully tuned. If C is too large, the agent acts too simplistically since it attempts to find the target by continuously moving closer on every timestep, often resulting in the policy converging to a local optimum [78]. Generally, a smaller value is necessary in environments with obstacles and the value was therefore dampened as the difficulty of the environment increased. In *SimpleNav*, C was 0.01. This figure was decreased by a factor of ten in *DifficultNav* and by a factor of 1000 in *ObstacleNav* and *Maze1Nav*. Due to the complexities of *Maze2Nav*, where an agent needs to learn to move in a very specific manner to circumvent obstacles and find the target, C was further dampened by a factor of ten and assigned to a value of 0.000001. Note that C was assigned the same value for agents trained using the shaped reward only and those trained using the directed curiosity reward. The reason for such small values was to enable agents to explore the environment sufficiently by ensuring that the shaped rewards did not dominate the curiosity reward signal.

After the hyperparameters were optimised, the three reward signals (directed curiosity, curiosity and a distance-based shaped reward) were compared by learning a policy using PPO (Algorithm 4.1), in each of the five environments, using euclidean distance to calculate the distance between the agent and the target in Algorithm 1. Five independent runs were performed and the results that are reported is the mean learning curve with standard deviation. Another important training setting is utilising multiple instances of the same environment during training i.e. multiple agents were trained concurrently and they were all controlled by a single policy. This was introduced in order to decrease the training time since the policy could be updated more frequently due to the increased feedback. All experiments were run on a Centre for High Performance Computing [1] node with 24 cores.

4.4 Policy Generalisation

The second aim of this work was on policy generalisation. Agents needed to learn a single policy that enabled them to find targets across different training environments, instead of a policy that was optimised for a single environment. Agents were therefore trained across a large number of training environments and the generalisability of the policies was evaluated in previously unseen testing environments, without making any changes to the learned policy. The set of testing environments included environments that were either variations of the training environments or had with new obstacle configurations.

Since it is difficult for an agent to learn this task directly, a manually-designed curriculum was created for this purpose. The curriculum was evaluated against curiosity-driven exploration [58] and a hybrid approach that combined the curriculum with curiosity.

For these experiments, walls were added along the boundaries of the square environment. This is because it was previously shown in Section 4.3 that the agents were able to learn to avoid falling off the platform. We therefore did not wish to detract from the focus of these experiments: the agent's goal was to navigate to targets and adding an additional expectation of learning to avoid falling off the platform made training unnecessarily more difficult.

The observation set O of the agents comprised the coordinates of the agents current position, the coordinates of the target and rays that extend in eight directions, at 45° intervals.

The observations were stacked to equip the agent with a memory of the immediate past. The previous ten observations were stored at any given time and the agent was not given any explicit information about the dynamics of the environments.

The rays, which were not part of the observation set in the policy optimisation experiments, provide essential feedback to the agent by enabling it to detect walls and targets that are in its vicinity. In the optimisation experiments, the agent was trained in a single environment and the rays were therefore not necessary but in

the generalisation experiments, where an agent observes multiple different obstacle configurations (across different environments), it is necessary to equip it with a means of observing the obstacles so that the policy can be adapted accordingly. The policy needs to be robust such that it is not optimised to one single environment but rather enables agents to find targets in arbitrary environments.

Furthermore, the rays take on additional importance when agents are placed in previously unseen environments. If an agent repeats memorised actions, it will move directly into walls and never reach its destination. When an agent detects an obstacle in its vicinity, it needs to use the ray feedback to move away from the obstacle in the direction of an open path. If the agent was not equipped with a means of observing the environment, it would not be possible for it to navigate in environments with obstacle configurations that are vastly different to those observed in training.

The ray length was tuned to balance the difficulty of the task: if the rays are too long, the agent is able to unrealistically detect objects that are very far away i.e. it is too far-sighted. If it is too short, the agent is unable to detect anything besides that which is immediately in front of it, causes it to move directly into obstacle, making the task significantly more difficult. This is analogous to the field of view. It was also observed that when the rays were too long, it became difficult for the agent to make sense of its observations since one (or more) of the rays were being "activated" on almost all timesteps. In order to balance the difficulty of the task, the ray length was fixed at 10% of total width (and length) of the environment. The size of the agent and rays are depicted in Figure 4-6 to illustrate the scale of the task.

The action set A is the same as in Section 4.3 i.e. the agent can move up and down along the y-axis, left and right along the x-axis, diagonally or remain still.

By default, before any training modifications were made, all the environments were sparse reward environments since an agent only received a +1 reward for finding the target. The starting positions of the agent and the target were at opposite ends of the environment. The agents did not receive any intermediate rewards and incurred a small penalty on every timestep to encourage them to find the target in the shortest possible time. The penalty was normalised to ensure that it never dominated the

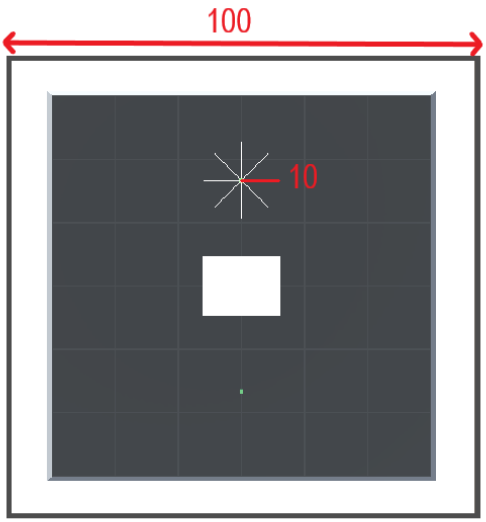


Figure 4-6: Agent and ray length relative to the environment.

terminal reward. On every episode, an agent received a penalty of $1/m$, where m is the maximum number of episode steps, which in this instance is 20000.

4.4.1 Policy Generalisation Environments

There were multiple environments and each varied in terms of the configuration of walls and obstacles. This was to deter agents from learning an optimal policy in a single environment. Rather, agents needed to learn the "skill" of finding a target that can be transferred to an arbitrary navigation environment. The predefined environments were carefully designed to represent high-level features or environment characteristics such as dead-ends and multiple paths to the target. Each environment characteristic aimed to teach an agent a particular type of behaviour or skill. For example, in an environment with a dead-end, the agent needed to learn to backtrack and try a different path. In some environments, it needed to learn to move away from the target for a large number of consecutive steps in order to bypass an obstacle. The rationale is therefore to equip the agent with enough of these "skills", so that it may succeed in unknown environments with similar or new characteristics i.e. it is necessary to introduce agents to numerous environment features in training so that they

may learn a flexible policy that enables them to find targets when similar features are found in new environments.

The sizes of the environment, agent and target, as well as the agent speed remained the same as in the experiments from Section 4.3.

We defined a separate set of training and testing environments. This was necessary in order to evaluate the generalisability of the policies.

4.4.1.1 Training Environments for Policy Generalisation Experiments

The training environments were partitioned into 2 categories: *Obstacle* environments and *Maze* environments.

Obstacle Training Environments These environments contained only a single obstacle that varied in terms of size and orientation. The size ranged from a scale of between 0 and 3 and the orientation was defined by an angle from 0°, in 45° increments. Obstacles are depicted in white in Figure 4-7.

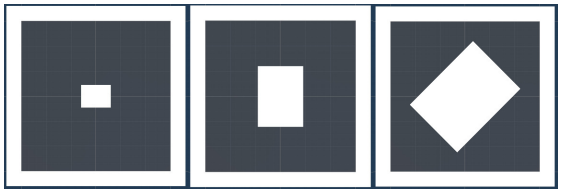


Figure 4-7: Obstacle training environments

Maze Training Environments Maze environments are more complex since they contain multiple obstacles. They were subdivided based on difficulty. *Standard* mazes are shown in Figure 4-8 and *Difficult* mazes in Figure 4-9. *Difficult* mazes had multiple obstacles that spanned more than half the width of the entire environment. They also included more complex versions of some of the *Standard* mazes, by manipulating the size of each obstacle in an environment. Another metric for difficulty is the number of "counter-intuitive" or backward moves that agents need to make to navigate to the target i.e. movement that places an agent further away from the target in order to navigate past an obstacle. A difficult environment requires agents to make more

"counter-intuitive" moves.

A popular maze structure, known as the "u-maze" [17, 21, 78], due to the shape of the obstacles, was incorporated into the standard mazes. Standard RL algorithms failed to solve this task in [17], highlighting the difficulty of this task.

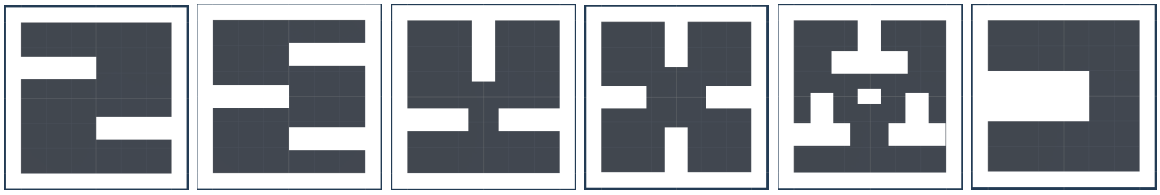


Figure 4-8: Standard training mazes

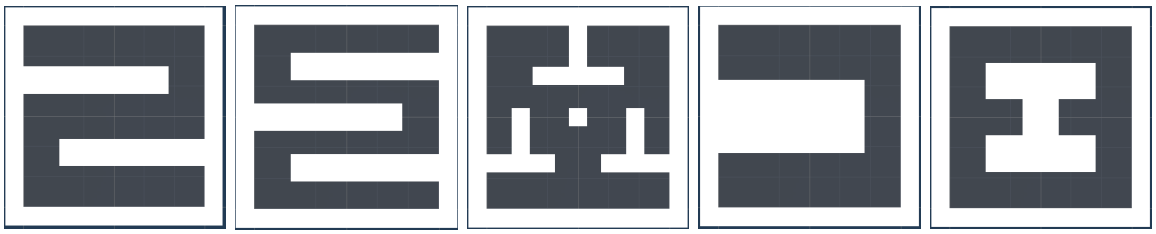


Figure 4-9: Difficult training mazes

4.4.1.2 Testing Environments

The testing environments were used to assess the extent to which the learned policies have generalised. They were divided into two categories:

1. Orientation Testing Environments

   An orientation testing environment is created by rotating a maze training environment by a fixed angle, (either 90° or 180° depending on whether an environment is symmetrical), as depicted in Figure 4-10. Even though the shape of the obstacles remains the same, rotating the obstacles changes the path to the goal significantly. This means that an agent will only be able to navigate to the destination if its policy is robust and adaptable. The *Standard Orientation* and *Difficult Orientation* mazes were created by rotating the standard (Figure 4-8) and difficult training mazes (Figure 4-9) respectively. All of the orientation mazes are depicted in appendix A in Figure A-4 and Figure A-8.

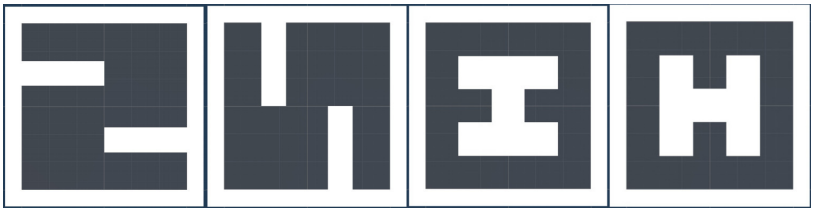


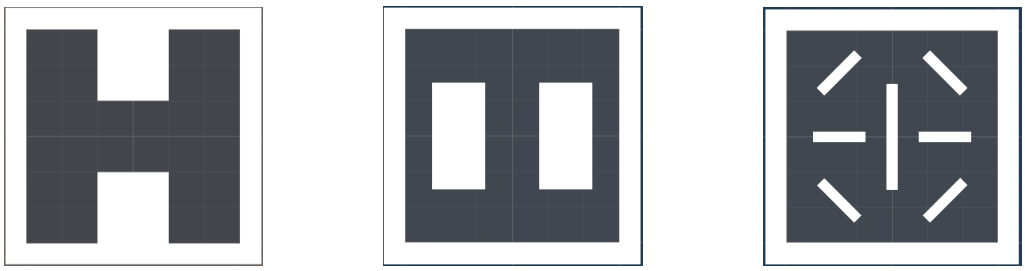
Figure 4-10: Rotating training mazes to create orientation testing environments

2. New Testing Environments

   New environments contain different obstacle configurations to the training environments. Previously unseen features or environment characteristics (bottlenecks, repeated symmetric obstacles and a large number of small scattered obstacles) were incorporated into this group. This allowed us to analyse whether the agents were able to learn advanced skills and further assess the extent of the generalisation.

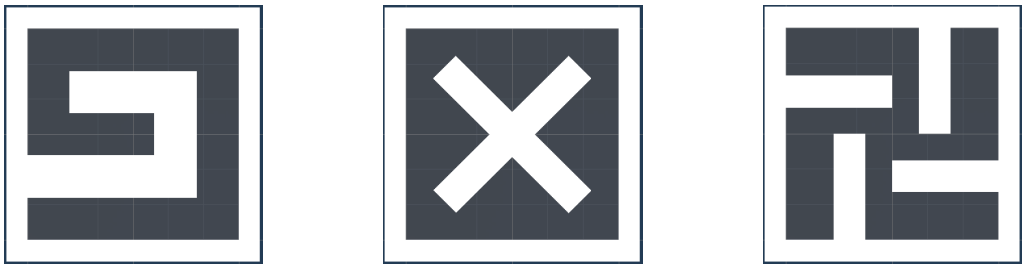
   Both these categories were further subdivided into *Standard* and *Difficult* subcategories, as per the definition used for the training environments.

   Both the *Standard New* and *Difficult New* groups, depicted in Figure 4-11 and Figure 4-12 respectively, contained 3 mazes each. The *Multi-Path* Maze (Figure 4-11b) and *Spiral* Maze (Figure 4-12a), as seen in previous work [21], was incorporated into the standard and difficult categories respectively. The difficult mazes were deliberately designed to test the boundaries of the algorithms and to identify their limitations.



(a) H maze (b) Multi-Path maze (c) Slim maze

Figure 4-11: Standard New testing mazes



(a) Spiral maze (b) X maze (c) Overlap maze

Figure 4-12: Difficult New testing mazes

Even though the testing environment categories were carefully designed to include a diverse set of mazes, the sample sizes of the groups are fairly small. In future work, we wish to increase the size of the groups and investigate whether the same principles and results hold. However, while it was possible to generate or design more mazes, the focus of this work was rather on the comparison between different training algorithms. Consistency in the training and testing environments was therefore more important. If there were too many environments, it would have been difficult to identify limitations of the algorithms. We therefore restricted the environments so that each environment represented a distinct feature.

4.4.2 Experimental Setup and Hyperparameter Optimisation

Agents were trained in the training environments defined in Section 4.4.1.1 under three different training settings: using the curriculum detailed in Algorithm 3, curiosity-driven exploration [58] and a hybrid "curiosity-curriculum" approach.

A policy is represented by a neural network. Under all three training settings, PPO [67] was again used to learn policies (for the reasons detailed in Section 4.3.2). The hyperparameters were therefore tuned in a similar manner as well. We defined baseline hyperparameters by training agents in the easiest version of the task and then carefully tuned and optimised these values by observing the training process and optimising the relevant parameter as required. The optimised hyperparameters from Section 4.3.2 were also used as a baseline.

The specific hyperparameters are given in Table 4.2 :

Table 4.2: The baseline hyperparameters for the policy generalisation experiments.

Hyperparameter	Value
Maximum Steps	2.0×10^7
Learning Rate α	3.0×10^{-4}
Time Horizon	64
Buffer Size	5120
Batch Size	256
Hidden Layers	2
Number Neurons	256
Beta β	0.01
Epsilon ϵ	0.1
Lambda λ	0.95
Number Epochs	3
Extrinsic Reward Strength	1.0
Intrinsic Reward Strength	0.01
Extrinsic Reward Discount Factor γ	0.99
Intrinsic Reward Discount Factor γ	0.99

Both the actor and critic models have two hidden layers (each with 256 units). The swish activation function [61] was once again used due to its robustness and also since it was shown to perform better then other activation functions across various RL tasks [18].

All agents were trained for twenty million training steps. The number of steps was carefully tuned to ensure that the agents had sufficient time to learn the task. However, we observed an interesting phenomenon whereby agents overfitted to the training environments, resulting in weak policy generalisation in the testing environments, when the number of training steps was too high. This observation was also made in [87].

Due to the increased complexity of the task, the maximum number of training steps and the number of neurons in the hidden layers needed to be drastically increased, when compared to the optimisation experiments. The entropy coefficient was also increased: this was to equip the agents with versatility such that there is more randomness introduced into the policy so that agents may explore the different training environments with vastly different obstacle configurations. This also resulted in stable training: when the entropy was too low, the policy did not converge since the agents tended to get stuck in local optima. The learning rate was decreased to deter agents from overfitting and also to speed up training. To ensure that neither the extrinsic or intrinsic reward signal dominated, the strengths of each of the reward signals were carefully tuned. This is a difficult task [12]: decreasing the curiosity strength resulted in shorter training times though there is a trade-off since the curiosity signal must be large enough to equip the agents with sufficient exploration capabilities.

Since the environment changes at the beginning of every episode, the agent observations are vastly different. It was therefore necessary to increase the batch and buffer size to allow the agent to gather more experience and capture a larger window of information, before updating its policy.

Both the algorithm used to train agents as well as the process of hyperparameter optimisation are essential. This is because subtle changes to a single hyperparameter can alter the results of the experiments drastically and ultimately be the difference between agents succeeding and failing.

4.4.2.1 Curriculum Parameters

The curriculum from Section 3.2.1 was used in the following manner in these experiments: the obstacle environments in Figure 4-7 were assigned to O in Algorithm 3, with a maximum obstacle scale ($S_{obstacle}$) of three. The set of maze environment M was represented by the union of the standard and difficult mazes from Figure 4-8 and Figure 4-9. The maximum environment scale ($S_{environment}$) was fixed at five to ensure that the largest environment is a sparse reward environment.

The curriculum threshold was fixed at 5000 i.e. the task changed when the average reward from the previous 5000 episodes ($n_{consecutive}$) reached a predefined threshold, $R_{threshold}$. This was to ensure that the agent had sufficiently learned to complete a task i.e. it consistently found the target across all the training environments. The reward threshold $R_{threshold}$ was gradually decreased as the curriculum advanced, since agents obtained lower episodic rewards in larger environments with multiple obstacles because more steps were necessary to navigate to the target. $R_{threshold}$ was fixed at 0.995 for the easiest task (this was tuned such that it was marginally lower than the average reward that an agent obtains when navigating to the target in the shortest possible time).

When the task was made more difficult by sampling maze environments instead of single obstacle environments (without increasing the environment size), the reward threshold was decreased by 0.0025. Thereafter, when the environment size was increased, the threshold was decreased by 0.005. This means that $R_{threshold}$ was 0.965 in the final version of the task.

4.4.2.2 Baseline Algorithms

The performance of the curriculum was compared to curiosity-driven exploration (see Equation 3.1) defined by Pathak et al. in [58] since it showed promising generalisation capabilities in previous studies [11, 58]. This equips the agent with an intrinsic reward that allows it to explore the training environments by seeking "novel" states, thereby gaining an understand of the dynamics of the various training environments.

The final approach combined the curiosity reward with the hand-crafted curriculum which we term "Hybrid": agents were trained using the curriculum from Section 3.2.1 and a reward function augmented with a curiosity signal. Both algorithms were shown to improve generalisation individually in previous work [12, 22, 11], it was therefore necessary to investigate if there were any merits to combining them.

Directed Curiosity from Section 3.1.3 was not included as a baseline algorithm due to the difficulties of reward shaping when there are multiple environments i.e. when we trained agents using the technique, we found that it resulted in poor generalisation.

This is referred to as the *Reward Engineering Principle* [16], whereby the more general the RL system is required to be, the more difficult it is to design reward functions to enable agents to behave specifically as required. The limitations of the shaped reward are discussed in more detail in Section 5.2.2.5.

After tuning the parameters independently for each algorithm, the agents were trained on a node, from the Centre for High Performance Computing [1], with 24 cores. The Unity ML-Agents platform [37] was used to train agents, as well as to design and implement the task and environments.

The codebase for these experiments can be accessed at: `https://github.com/AsadJeewa/Learning-to-Generalise-in-Sparse-Reward-Navigation-Environments`.

As was the case in the previous set of experiments, training was performed on five independent runs (for each algorithm) and the mean result with standard deviation is depicted. The best performing policy, for each algorithm, was selected and evaluated in each of the different testing environment groups, to assess the extent to which the policy transferred to the unseen environments. The evaluation process is detailed further in Section 5.2.

Chapter 5

Results and Discussion

This work investigated both policy optimisation and generalisation in a set of custom sparse reward navigation environments. We present a novel reward function, called Directed Curiosity, that combines curiosity-driven exploration [58] with distance-based shaped rewards from Algorithm 1. The aim of the experiments was to enable an agent to learn an optimal path within the single environment it was trained in.

Furthermore, we present a curriculum (Algorithm 3) that attempts to bypass the sparse rewards problem as well as improve policy generalisation. Instead of learning polices that optimise to a specific environment, we attempted to train agents to learn the "skill" of finding a target in arbitrary environments and then analysed the extent to which policies generalised by evaluating agents in unseen testing environments.

5.1 Policy Optimisation

Directed Curiosity was evaluated against agents trained using a reward function made up exclusively of (a) the distance-based shaped reward function and (b) the curiosity reward signal [58], across the five different sparse reward navigation environments. A third baseline is also depicted: a sparse reward function that returns a terminal reward of +1 for finding the target and -1 for falling off the platform.

Figure 5-1 to Figure 5-5 compares the algorithms in each of the five environments: the extrinsic rewards from a single episode is depicted as a function of steps i.e. a

single iteration of the RL loop (see Figure 1-1). The rewards were normalised between 0 and 1 to allow for the algorithms to be compared directly to each other.

Figure 5-1: Learning curves for SimpleNav environment.

Figure 5-2: Learning curves for DifficultNav environment.

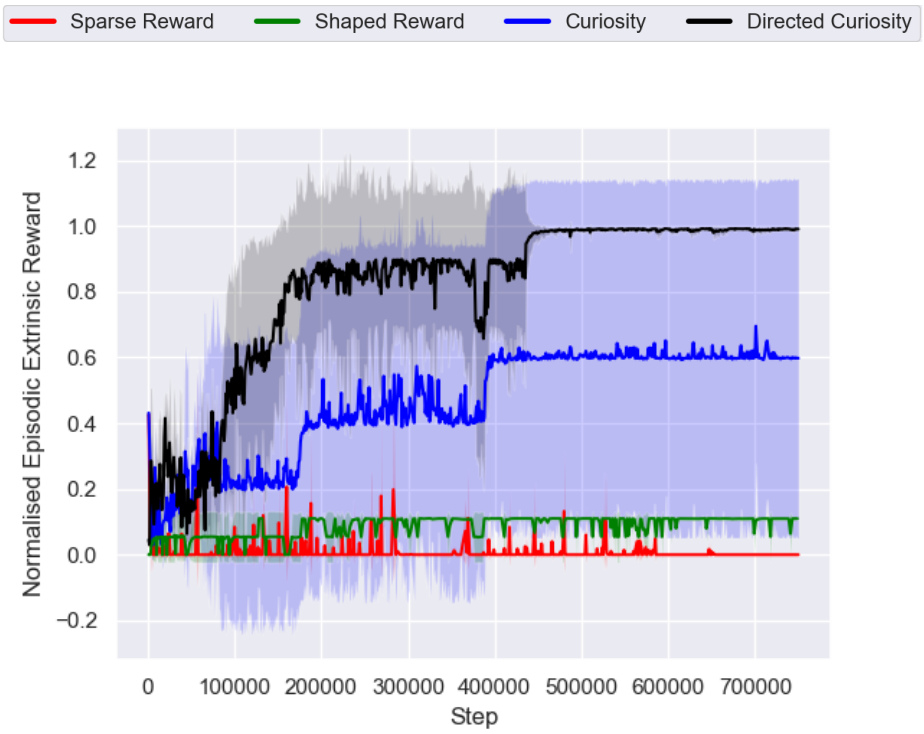


Figure 5-3: Learning curves for ObsatcleNav environment.

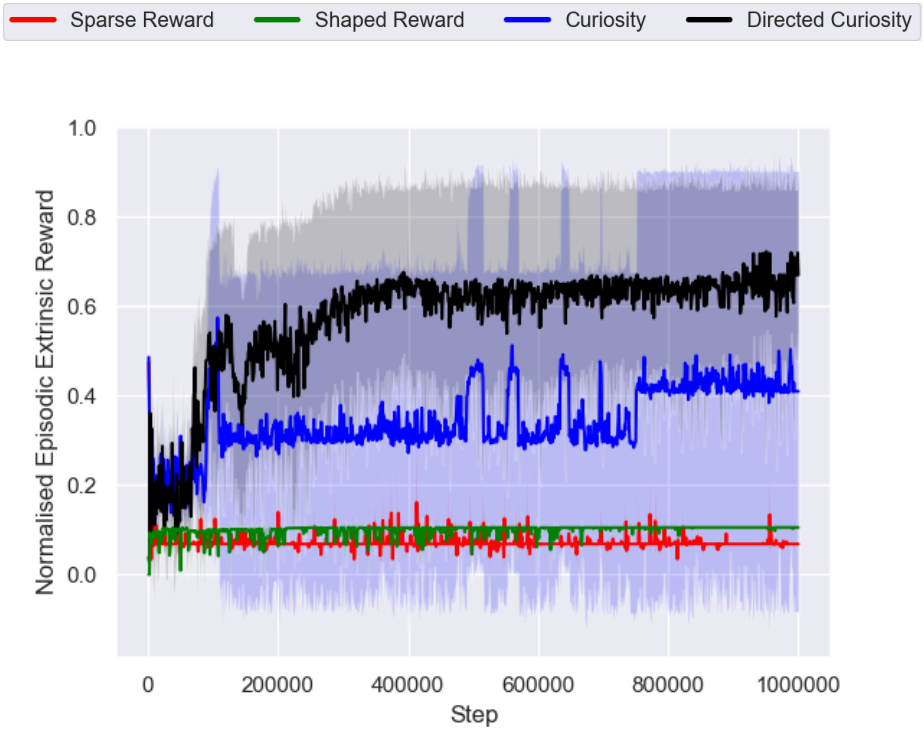


Figure 5-4: Learning curves for Maze1Nav environment.

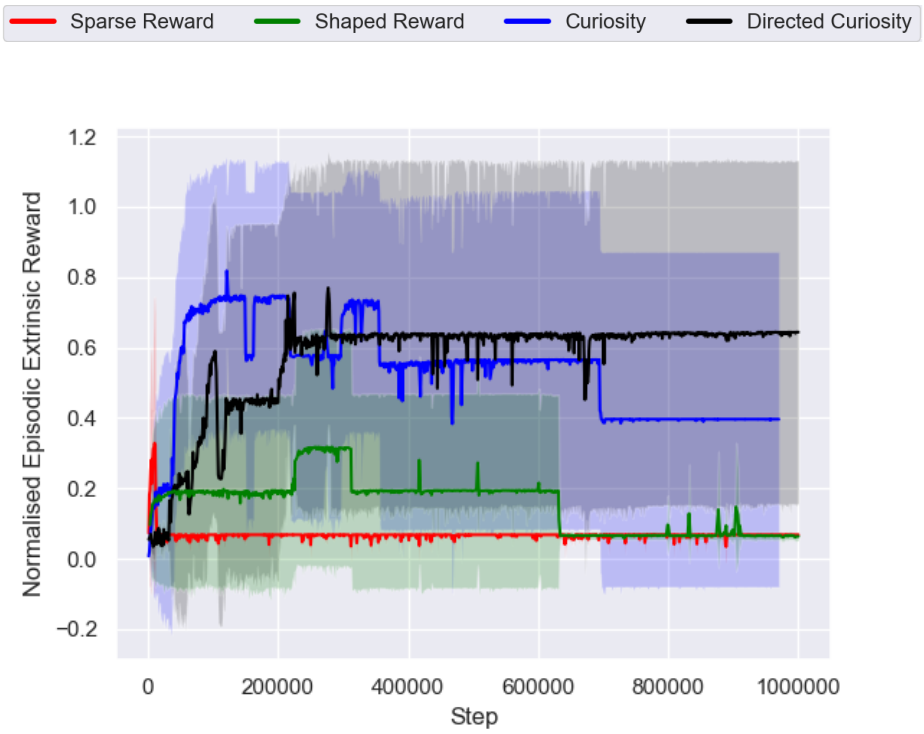


Figure 5-5: Learning curves for Maze2Nav environment.

Each algorithm was run five times in each of the five environments and the shaded area represents the standard deviation of the curves (as is customary in RL) [3, 63]. This is an important metric for various reasons. Firstly, the standard deviation of all algorithms increases proportionally to the difficulty of the task: the agents do not always converge to an optimal policy in difficult environments i.e. if an agent fails to find the target, it does not receive the terminal reward and hence its episodic rewards are significantly lower. Also, since PPO learns a stochastic policy, the algorithms converge at different times, even on successful runs. Furthermore, the agent explores differently on every run and therefore visits states in a different order.

5.1.1 Sparse Reward Agents

SimpleNav was the only environment wherein the sparse reward agents were able to navigate to the target. They did not perform consistently in this environment, only finding the target and learning an optimal policy on some runs. This is the reason for the high variance in Figure 5-1.

In all of the subsequent environments, the agents were unable to navigate to the target on all five runs and therefore received no positive rewards during training. The small changes in reward are as a result of the negative feedback that the agent received when it fell off the platform. This feedback was frequent since falling off the platform was a common occurrence and the agents were able to use this to learn to remain on the platform for the entire duration of an episode. This is a local optimum and not the behaviour that the agents were required to learn. These results therefore verify that the environments are sparse reward environments. They also highlight the need for an exploration strategy when rewards are sparse.

5.1.2 Reward Shaping Agents

The reward shaping agent performed well in *SimpleNav*. This is because the shaped rewards act as a definition of the task when there are no obstacles blocking the direct path to the target. Selecting an action that moves the agent closer to the target on every timestep leads it directly to the goal in the shortest possible time. In *DifficultNav*, the agents were able to learn an optimal policy significantly faster than all other reward functions, for the same reasons. The rewards converged almost immediately: this highlights that reward shaping can be very effective when the desired behaviour of an agent can be easily defined. However, this environment is not practical and the agent is required to move in a more complex manner in more realistic environments.

The deficiencies of the shaped reward function began to surface when obstacles were introduced and the difficulty of the task was increased (see Figure 5-3 and Figure 5-4). The agents failed to find the target on all runs in *ObstacleNav* and *Maze1Nav*, and got stuck behind obstacles in the early stages of training. This is because the shaped reward function is a greedy approach and the agents were not equipped with the foresight to learn to move around the obstacles. An agent is unable to learn to move further away from the target at the current time, in order to reach the target at a later stage. This is generally a common problem with distance-based shaped reward functions [78].

The same problem was still observed even when the agents were initialised with higher entropy i.e. encouraging the agents to take actions more randomly. As discussed in Section 3.1.2, we did attempt to address this issue by making the reward function more flexible, by calculating the change in position every n steps but this did not have a significant effect on training and the agents still failed to navigate to the target.

It is interesting to note that the agents were able to find the target on two of the five runs in *Maze2Nav* (see Figure 5-5). Even though there are multiple obstacles in the environment and the task is generally a difficult one, the optimal path to the target is more straightforward than *Maze1Nav* and *ObstacleNav*. The path is not a direct line though the there are fewer "backward" steps necessary. The agents therefore "ignored" the obstacles and avoided dead-ends by acting simplistically, eventually finding their way to the target. However the approach is not robust since the agent was unable to converge to an optimal policy on any of the runs.

Overall, the results indicate that distance-based reward shaping provided the agents with some important feedback that enabled them to succeed in simple environments. However, due to the greedy nature of the reward function and the lack of an intuitive exploration strategy, the agents were unable to learn to move past obstacles that block its path to the target.

5.1.3 Curiosity Agents

The curiosity agents were able to consistently learn an optimal policy in the environments without obstacles. However, Figure 5-1 shows that the curiosity agents took longer to converge to an optimal policy in *SimpleNav*. This highlights that curiosity is not necessary in environments that are not "hard exploration" or when the reward feedback is not sparse. A similar pattern was observed in *DifficultNav* (see Figure 5-2), where the curiosity agents learned an optimal policy significantly slower than agents trained with the shaped reward function.

The necessity of the curiosity signal was highlighted when obstacles were introduced. Curiosity enables agents to not only find distant targets, but also to implicitly

learns about the dynamics of the environment. As agents explore an environment, they observe new states and begins to build an understanding of the location and dimensions of the walls and obstacles, even though none of this information is given to it explicitly. The agents use this knowledge to learn how to control themselves to move around obstacles and find an optimal path to the target.

In *ObstacleNav* (see Figure 5-3), the agents were able to learn an optimal policy on three of the five runs, unlike the Directed Curiosity agents which were successful on all five runs. The performance was less successful in *Maze1Nav* (see Figure 5-4) and *Maze2Nav* (see Figure 5-5), where the agents were only successful on two of the five runs. This highlights the difficulties of these environments and of the task: even if an agent reaches the target in an episode, it does not guarantee that the policy will converge i.e consistency is key and the task of learning the optimal path to a target is significantly more difficult than simply finding the target.

It was observed that the curious agents seemed to keep exploring after initially finding the target and got stuck behind obstacles and in dead-ends, eventually converging to an unsuccessful policy, without being able to reach the target again. This highlights the insufficiencies with the curiosity signal, since it does not equip agents with the knowledge it needs to "guide" it back to the target and learn a path to the destination. This is the reason for the increase of the average reward in the early stages of training and the subsequent drop thereafter in Figure 5-5.

A similar effect was observed in [58]: when the task becomes too difficult for the agent, it is unable to reach new environment states, and therefore hits a curiosity blockade. Naturally, without any intrinsic and extrinsic feedback, the policy deteriorates. The authors likened this to boredom whereby the agent no longer has any motivation to explore the environment.

Generally, the results indicate that curiosity equips an agent with the ability to find a target in hard exploration environments that contain obstacles. However, the agent requires additional feedback to learn a path from the start point to the destination that enables it to navigate to the target consistently.

5.1.4 Directed Curiosity Agents

The Directed Curiosity agent is shown to be the most robust technique. Figure 5-1 and Figure 5-2 show that the agents learned optimal policies on all runs in *SimpleNav* and *DifficultNav*. The inclusion of the shaped reward function enabled agents to converge to a solution faster, after a smaller number of training steps than the curiosity agents, in *DifficultNav*.

The hard exploration environments further highlight the benefits of the technique. Directed Curiosity is the only technique that converged to an optimal solution on all runs in *ObstacleNav* (see Figure 5-3). The results indicate that the curiosity signal enables the agents to find the target and move past the obstacles, while the shaped reward signal provides additional feedback that allows the agent to learn an optimal path to the target, once it has been found. In this way, it combines the strengths of the two separate techniques, resulting in training that is more reliable and ultimately more successful.

Maze1Nav (see Figure 5-4) and *Maze2Nav* (see Figure 5-5) exhibit promising results since the Directed Curiosity agents learned an optimal policy on more runs than any other technique i.e. on three of the five runs. The training process was also more stable than the curiosity agent. While the agents did find the target on all five runs, they were unable to consistently learn an optimal policy, though this effect was observed on fewer runs than was observed for the curiosity agents. A major reason is due to the limitations that we have previously highlighted with the shaped reward function.

In future work, we wish to investigate a more intuitive reward function that incorporates foresight in a more intelligent manner than the approach discussed in Section 5.1.2. One interesting avenue is to investigate automatically learning a reward function through inverse reinforcement learning techniques.

The maze environments were deliberately designed to be challenging to solve so that we could identify limitations in Directed Curiosity. It was therefore expected that the agents would not be able to learn an optimal policy on all runs.

The results indicate that the reward feedback is not sufficient to guide the agent out of dead-ends back to the target. However, these results indicate that the two components of Directed Curiosity, when balanced correctly, allow the agent to learn in a more directed and intuitive manner.

5.1.5 Summary of Optimisation Results

The results of the sparse reward agents verify that the environments are sparse reward environments, since they failed in all of the environments. It also highlights the need for an exploration strategy in sparse reward hard exploration environments. The distance-based shaped reward function provided agents with important feedback that enabled them to succeed in simple environments, but the limitations of the function was exposed in environments with obstacles: due to the greedy nature of the reward function and the lack of an intuitive exploration strategy, the reward shaping agents were unable to learn to moves past obstacles that block its path to the target. The curiosity agents were more successful: curiosity equips agents with the ability to reach distant targets in the maze environments. However, the agents often failed to converge to optimal polices since they are not given any reward feedback to "guide" them back to the target and the agents often got stuck in local optima. The Directed Curiosity agents were the more successful: not only did they converge to optimal polices after a smaller number of training steps, the results also indicate that the two components of the reward function, when balanced correctly, allow the agent to learn in a more directed and intuitive manner, since agents are able to learn optimal polices the most often, in the difficult maze environments.

5.2 Policy Generalisation

Experiments in policy optimisation highlighted the merits of using Directed Curiosity to learn an optimal path within a single environment. However, the experiments also highlighted limitations in the shaped reward function and when agents were trained to find targets across multiple environments, policy generalisation was poor.

In general, Directed Curiosity is a useful technique for learning specialist policies, that are optimised for a specific environment, but it is not suited for policy generalisation.

The task of learning to navigate in an arbitrary environment is significantly more difficult. Instead of attempting to learn the task directly, we designed a training curriculum for this purpose, defined in Section 3.2.1

The curriculum was evaluated against an agent trained using curiosity [58] and a hybrid of the two approaches. Analysis was performed in three stages: the first stage compares the training performance of each algorithm, both in terms of average episodic reward and training time, in Section 5.2.1.

The agents were trained under a dense reward setting where the spawn positions of both the target and agents randomised at the start of each episode. Since the focus of the work is on sparse reward navigation environments and it is therefore important to evaluate the algorithms under a sparse reward setting (the default setting of the environments). This was achieved by positioning the agent and target at distant locations in every training environment. The locations were fixed at points that made the task as difficult as possible. We term this as *soft-generalisation* (Section 5.2.2.1).

A critical evaluation of the generalisability of each algorithm in the unseen testing environments from Section 4.4.1.2 was performed. This is referred to as *hard-generalisation* in Section 5.2.2.2.

Finally, in Section 5.2.3, we performed trajectory analysis by assessing the movement paths of the trained agents. This allowed us to understand the strengths and limitations of each algorithm by understanding the intricacies of how agents move within different environments.

5.2.1 Training Performance

The training curves i.e. the average episodic reward of the agents over time (for each algorithm) are depicted in Figure 5-6. As is the norm in RL [58, 63, 87] and for consistency with the experiments in Section 5.1, we performed five independent runs of each algorithm and report on the average learning curve. The standard deviation is also highlighted. Twenty independent instances of the environments were used for

more efficient data collection during training.

The dashed line in Figure 5-6 depicts the point at which both the curriculum and hybrid agents progressed to the final lesson, which corresponds to the training environments that the curiosity agent was trained in.

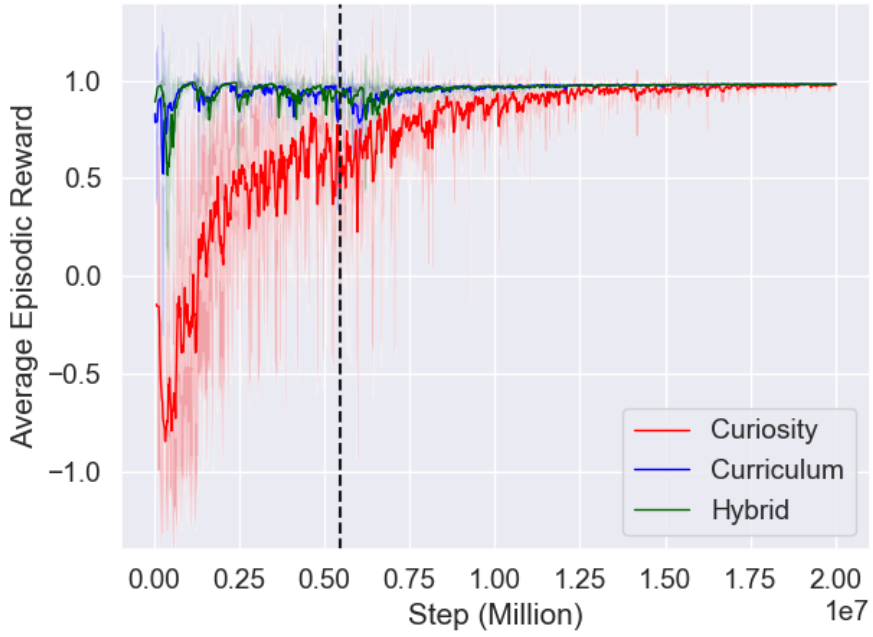


Figure 5-6: Learning curves during training.

A smoothing factor of 0.2 is applied for all algorithms in Figure 5-6. The curves highlight the benefits of using the curriculum. The blue curriculum curve never drops significantly since an agents' task is never too difficult. The curriculum advances quickly in the early stages of training when the task is easier i.e. the sudden drops in reward are indicative of points at which the task is made more difficult but the fact that the curve peaks very quickly thereafter, indicates that knowledge is being transferred between tasks.

The curiosity curve is a more traditional learning curve with rewards slowly increasing over time, since the task is difficult for the agents in the early stages of training and it receives minimal rewards.

The hybrid training curve is very similar to the curriculum agent. Even when the curiosity strength was varied, the curves still followed a similar pattern. This

indicates that the curiosity rewards had little effect on the training process when coupled with the curriculum.

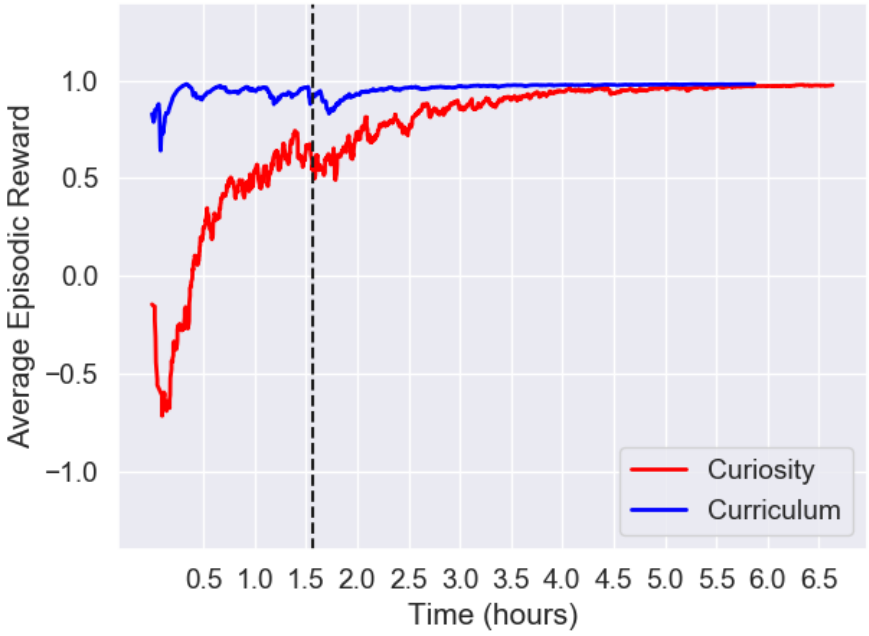


Figure 5-7: Average rewards against training time.

Figure 5-7 depicts the average episodic rewards as a function of training time. Since the hybrid and curriculum agents have training curves that are very similar, only the curriculum curve is depicted. A smoothing factor of 0.05 is applied. In all runs, it was noted that the curriculum agent converged significantly faster than the curiosity agent. Even though both algorithms ran for twenty million training steps, the curriculum agents trained for roughly 15% less time (almost 1 hour). This is attributed to the fact that the episodes were very long in the early stages of training, when the task was too difficult for the curiosity agent i.e. the episodes terminated only upon reaching the maximum number of steps.

There is also a clear gap between the rewards obtained, with the curriculum agents achieving significantly more rewards, especially in the early stages on training. Focusing on the most difficult version of the task, beyond the dashed line in Figure 5-7, the curriculum agents' rewards stabilised 22% faster (roughly 1.5 hours) than the curiosity agent. To further test this theory, we trained the curriculum agent for half

the number of steps and noted that it still converged on all runs while the curiosity agent was unable to do so on any of the runs.

5.2.2 Generalisation Performance

The algorithms were compared as follows: The best performing training run from Section 5.2.1 was selected for each algorithm since all the algorithms converged during training and the variance between different runs was low after convergence. Furthermore, all the results are reported using a confidence interval of 95%, as depicted through error bars.

The average episodic reward was then calculated and analysed for each of the separate environment groups defined in Section 4.4.1. Each algorithm was run for 1000 episodes, with a random testing environment being sampled at the start of the episode, from the corresponding group. This is necessary due to the stochastic nature of the policies: the agents sometimes succeed and fail within the same testing environments. This results in vastly different episodic rewards, meaning that a large number of episodes is necessary to stabilise the average rewards.

The average episodic rewards are in the range $[-1, 1)$. A successful run is one in which agents are able to navigate to the target. The faster an agent finds the target, the higher the reward it receives. An average reward approaching one therefore indicates that the agent successfully found the target on all runs. A score less than one indicates that on most runs, the agents were unable to find the target, across all environments, with zero representing an inflection point. Magnified results (that highlight precise differences between each algorithm) can be found in Appendix A.

5.2.2.1 Soft-Generalisation

Figure 5-8 illustrates that, for all algorithms, the agents were able to efficiently find the target in all training environments, under the sparse reward setting. All algorithms have an average reward that approaches a maximum possible reward of $+1$, with negligible difference between them (refer to Figure A-3).

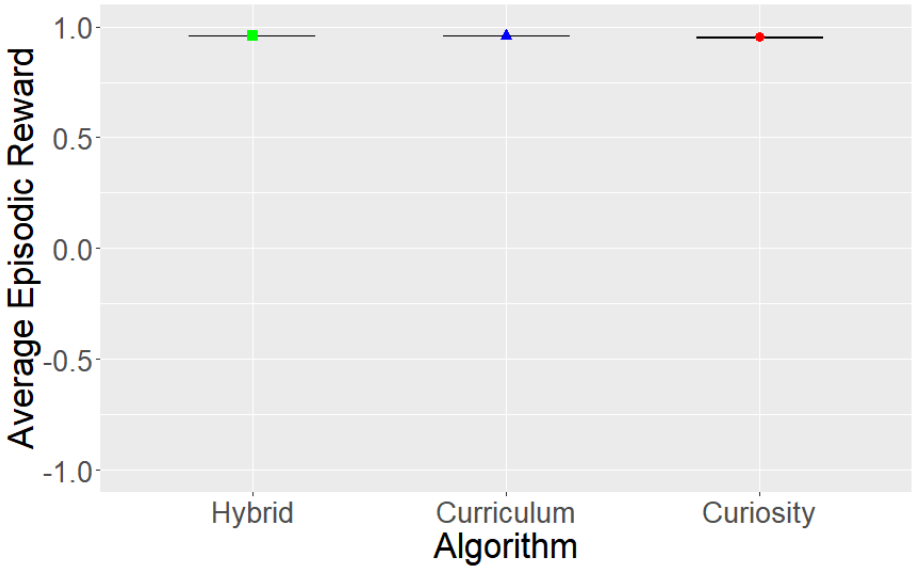


Figure 5-8: Average rewards in sparse versions of the training environments.

These results act as a validation of each algorithm since it indicates that all agents obtained sufficient knowledge of the task and were able to find targets across a diverse set of mazes. This allowed us to perform a fair comparison of the hard-generalisation capabilities of each algorithm, in the testing environments.

5.2.2.2 Hard-Generalisation

When analysing the hard-generalisation results, there are certain important considerations that need to be made: the task is not trivial since it as analogous to placing a human or vehicle in a new environment and only equipping them with information about its current location, destination and the ability to "see" what's around it. It does not have any knowledge of the dynamics of the environment that it is placed in. This means that some "exploration" is necessary and it is expected that agents will move into obstacles as they try to advance towards the goal. In other words, it is not possible to solve the generalisation problem completely: it was not expected that the agents would obtain expert performance in the testing environments. The goal was rather to transfer some knowledge that could be reused in the environments.

The policies were used "as-is" and there was no fine-tuning for any of the testing environments, as is the case in other studies [58]. It is possible to improve the results

in each testing environment by fine-tuning the policy though that was not the aim of this study. This work instead investigated the flexibility of the learned policy by analysing the extent to which it was able to generalise to unseen environments.

The performance of each algorithm is often different i.e. agents do not succeed and fail within the same testing environments. There were instances when one algorithm enabled agents to navigate to the target in a short time, but another resulted in agents only finding the target after a large number of episode steps or never at all, in the same environment. We wish to perform further analysis of this phenomenon, in order to discover any inherent environment characteristics that cause this divergence.

The gap between training and testing performance translates to the extent to which the policies have overfitted to the training environments [14]. As expected, there is a gap in performance though the results indicate that some generalisation has taken place.

5.2.2.3 Standard Mazes

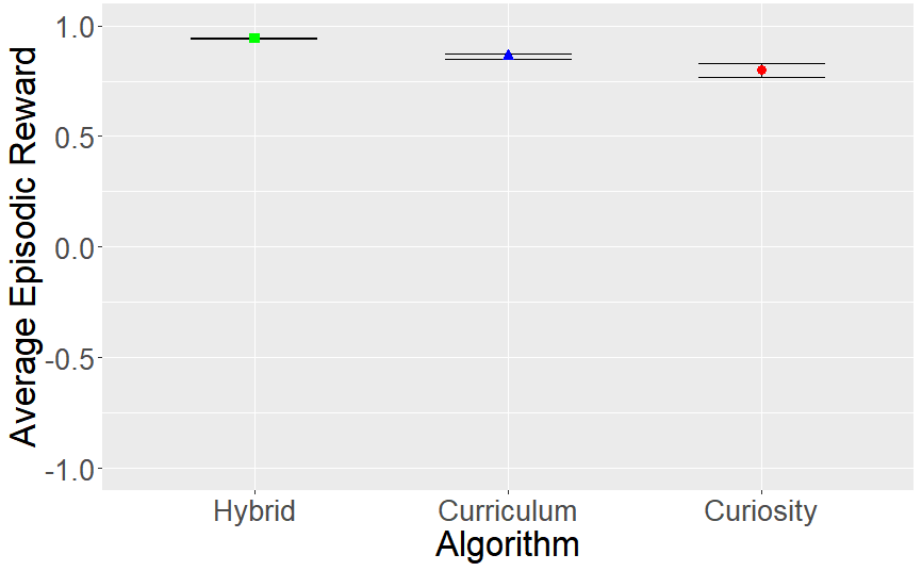


Figure 5-9: Average rewards in the Standard New mazes.

All the algorithms performed well in the standard mazes. Figure 5-9 depicts similar performance in the *Standard New* environments. Notably, all the agents were able to consistently navigate to the target in all three environments, indicating that all

algorithms generalise well to these environments. This is a promising result, since the obstacle configurations are different to those in the training environments, meaning that the policies are robust. The agents were able to utilise the ray feedback to adapt their policies to move around obstacles, while the stacked observations enabled them it to keep advancing towards the target. The difference in results comes from the episode length: on average, the hybrid agents found the targets marginally faster.

The *Standard Orientation* results in Figure 5-10 depict that all algorithms were able to succeed on most runs. The curriculum and hybrid agents performed marginally better than those trained with curiosity, across the six environments. Even though the environments are variations of training environments, the paths to the target are vastly different. This is one reason why the agents do not succeed in some environments.

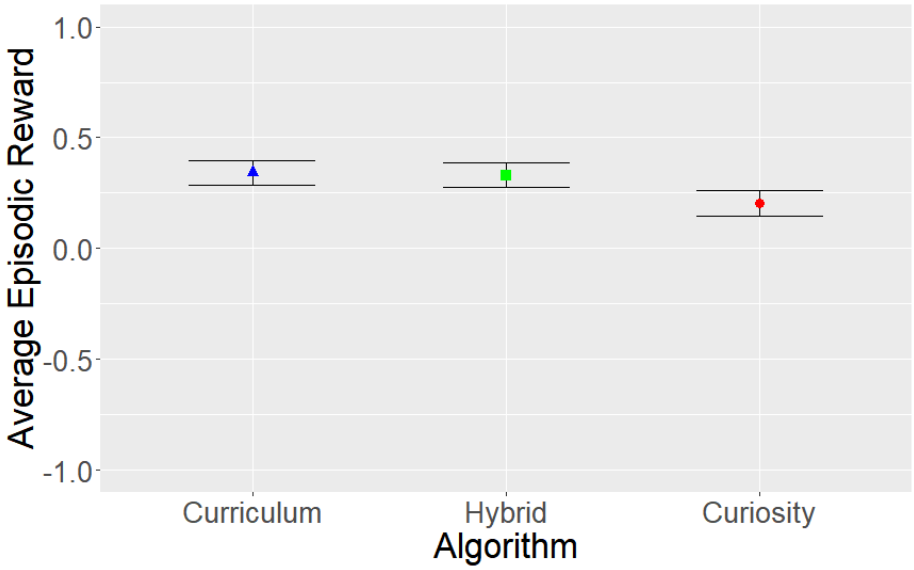


Figure 5-10: Average rewards in the Standard Orientation mazes.

In order to increase generalisability, there are other avenues that we wish to explore specifically with regards to optimising the curriculum. One such way is through fine-tuning the definition of the training environments, to include more environment characteristics. Instead of manually adding environments, a more efficient avenue would be to procedurally generate them, as explored in Section 2.3. The limitations of the curriculum are further analysed through trajectory analysis in Section 5.2.3.

5.2.2.4 Difficult Mazes

Generalisability is shown to decrease for all algorithms, as the difficulty of the environments was increased. However, unlike the standard mazes where all algorithms performed similarly, the benefits of the curriculum are highlighted in the difficult mazes. The agents that were trained using the curriculum were able to find the targets in more environments than the other two algorithms.

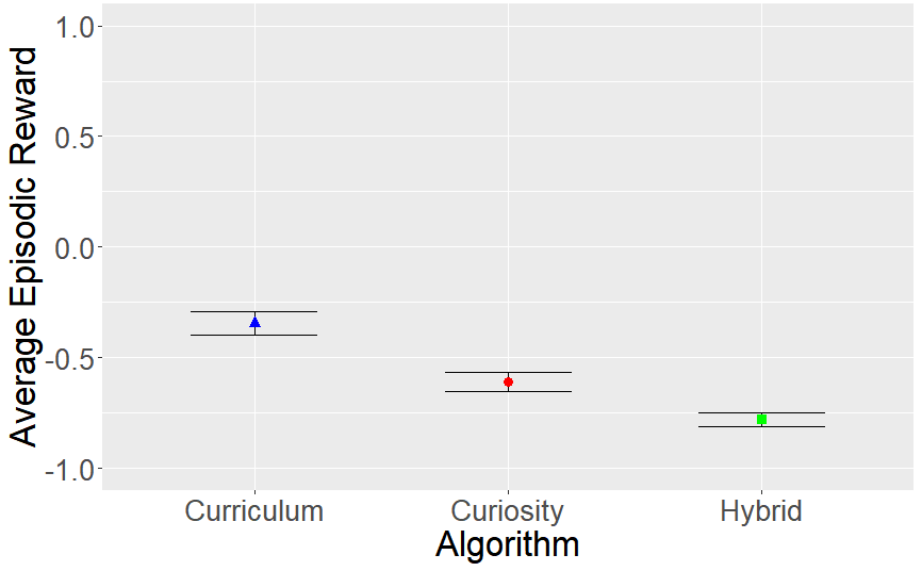


Figure 5-11: Average rewards in the Difficult Orientation mazes.

The *Difficult Orientation* results in Figure 5-11 indicate that, for all algorithms, the agents were not able to find the target on most runs, however, some transfer has taken place. The curriculum obtained the highest average reward and the error bars indicate that the difference is statistically significant under a 95% confidence interval. The performance of the curiosity agent showed limited transfer to the testing environments, with agents only succeeding in a single environment. Even though both the curriculum and hybrid agents succeeded in two of the five environments, the hybrid agents took significantly longer to find the targets. This resulted in the hybrid algorithm performing the most poorly.

The *Difficult New* environments show the least transfer (see Figure 5-12) though as per previous environment groups, the curriculum agents were the most successful. Due to the difficulty of the environments in this group, agents were not able to find

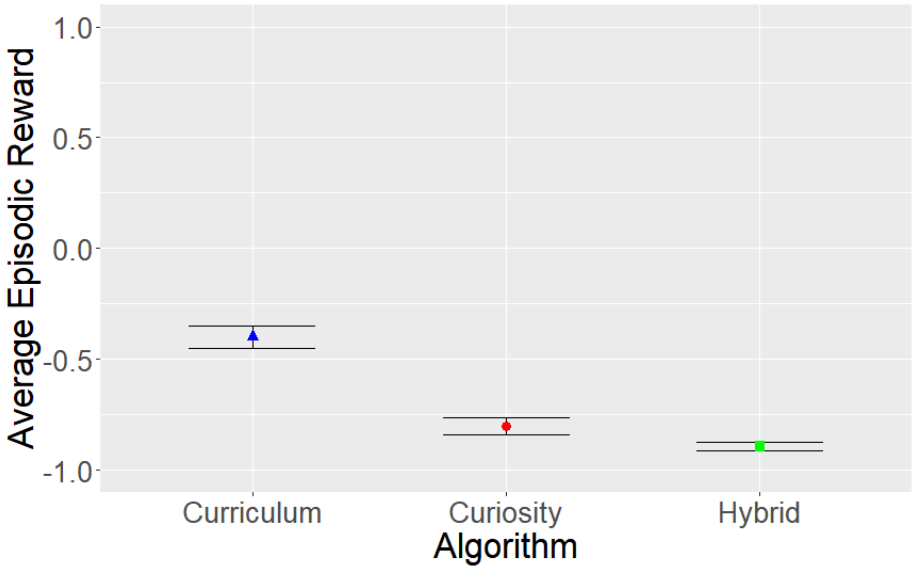


Figure 5-12: Average rewards in the Difficult New mazes.

the targets consistently i.e. they succeed on some runs and fail on others, even in the same environment.

The most promising result is that the curriculum agents were the only agents that succeeded in the *Spiral* maze depicted in Figure 4-12a. The curriculum agents were also able to navigate to the target fairly efficiently.

Even though all of the algorithms succeeded on some runs in the *Overlap* Maze, the performance was poor since the number of steps to find the target was almost at the maximum number of steps of 20000.

None of the algorithms succeeded in the X Maze (Figure 4-12b). We regard this environment as the most difficult. Even though the agent and the target are both positioned very close to each other, they are separated by obstacles and the agent has to immediately take a large number of consecutive steps away form the target, to navigate past the obstacle i.e. this environment requires the most "counter-intuitive" moves, as defined in Section 4.4.1.1. Furthermore, the shortness of the rays increase the difficulty of the task since the agents often detect the walls only after they have positioned themselves in one of the "crossings" of the "X". Increasing the memory capacity of the agents will likely improve performance.

5.2.2.5 Reward Shaping

A further benefit of the curriculum is that it does not require any reward shaping. This is due to the manner in which the curriculum was designed that ensures that the agents always receive sufficient reward feedback during training. We performed an empirical investigation into various different shaped rewards and found no performance improvements. Rather, the motivations of the agents became polluted [15, 53]. For example, when an agent was rewarded for moving closer to the target, it lacked the foresight to move past obstacles as previously observed in Section 5.1.2.

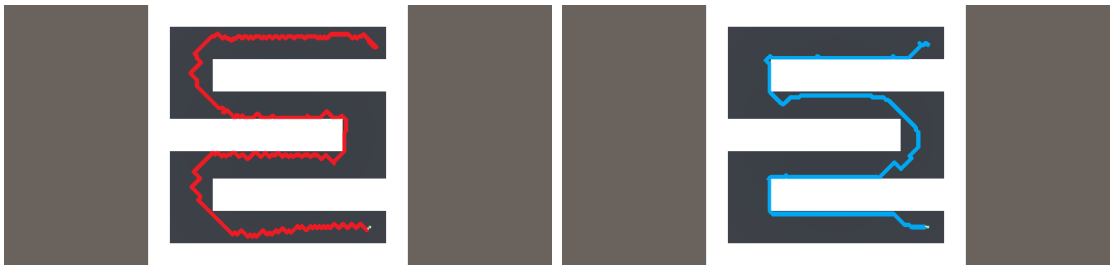
Shaping rewards also resulted in poor generalisation performance i.e. it resulted in specialist policies that worked well in some environments, but poorly in others. We theorise that this is possibly because the agent was receiving too much feedback which made it difficult to generalise a policy across all the environments. While the rewards were deliberately designed to promote general behaviour (move closer to the target where possible), the paths to the goal across the different environments are vastly different. The agents tend to learn better when they are left to explore on their own.

Reward shaping may also require additional information which may not be available in the real-world: it may not be possible for an agent to evaluate whether a single action has moved the agent closer or further away from the target.

5.2.3 Trajectory Analysis

We performed trajectory analysis by analysing the movement patterns of the trained agents across the different environments, as per [63].

It was often observed that the curriculum agents tended to move in a more directed manner than the curiosity agents i.e. the paths were smoother and with fewer "vibrations". The curriculum agents also tended to "stick" to the walls for longer periods of time. Since the walls are fixed, they acted as reference points for the curriculum agents and were used as a guide for finding a path to the target. An example of this is depicted in Figure 5-13. The trail of a curiosity agent is shown in red and



(a) Curiosity (b) Curriculum

Figure 5-13: Path comparison between a curiosity and a curriculum agent.

that of a curriculum agent in blue. There is further proof of this in Figure 5-14.

Figure 5-14: Path analysis of a curriculum agent.

Figure 5-14 highlights a behaviour pattern that is often observed for curriculum agents: they initially attempt to move directly towards the goal, along the shortest possible path, but when the agents detect an obstacle, they adapt to move around it, and then continue to attempt to move along the fastest route. This highlights the robustness of the policy.

(a) Repetitive movement (b) Local optimum (c) Too difficult

Figure 5-15: Limitations of curriculum agents.

A limitation of all algorithms is that it was sometimes observed that the agents repeatedly move along a similar path and only make slight advancements towards the target over a long period of time i.e. the agents tend to make a lot of redundant moves. However, the agents often find their way to the target eventually, as shown

in Figure 5-15a. This, therefore, does not point to deficiencies in the curriculum, but rather in the observations and capacity of the agents. In future work, we wish to explore different methods for increasing the "memory" of the agents, to alleviate this problem. One such way is to increase the number of stacked observations so that agents can "remember" more of their previous failures and avoid repeating the same actions. An alternative approach is to use recurrent architectures.

Understanding the limitations of the curriculum is important: Figure 5-15b and Figure 5-15c highlight examples of environments in which the curriculum agents failed to find the target. In Figure 5-15b, as the agent was progressing towards the target, it got stuck in a local optimum and then continuously repeated a similar sequence of actions, until the maximum episode steps was reached. Furthermore, even though it is possible that the agent would eventually have still found its way to the target, this points to a level of memorisation in the policy.

In some cases, as depicted in Figure 5-15c, the environment is simply too difficult and the agent does not have enough knowledge to be able to navigate to the target. When this environment was incorporated into the training set, the agent learned how to navigate to the target. We therefore theorise that introducing a larger number of training environments would mitigate this issue, as well as increasing the memory of the agents.

(a) Curiosity (b) Curriculum (c) Hybrid

Figure 5-16: Path comparison of all algorithms for the spiral maze.

The most promising result is shown in Figure 5-16. The spiral maze is difficult because the agent needs to learn a very specific trajectory in order to find the target. The curriculum agent was the only agent that succeeded in this environment. This further highlights the robustness of the algorithm: it was able to continuously adapt its actions as it observed the environment. The curiosity and hybrid agents both got

stuck in local optima, far away from the target. It was observed that once the agents were parallel to the target, they repeatedly attempted to move directly towards it, moving directly into the obstacle. The agents then attempted to adapt their policies but due to the difficulty of the task, eventually fell into the same movement pattern.

5.2.4 Summary of Generalisation Results

The generalisation experiments highlighted the benefits of using the training curriculum. By ensuring that the task is never too difficult for the agent, as well as enabling agents to transfer skills learned in simple environments to difficult ones, the agents are able to converge to an optimal policy (in the training environments) faster than agents trained with a curiosity [58]. Furthermore, the curriculum acts as a means of bypassing the sparse rewards problems, since agents trained with the curriculum, in a dense setting of the training environments, are able to find targets in sparse versions of the same environments.

Training with the curriculum resulted in policies that generalised better to unseen testing environments: it was able to navigate to the target across the most environments. The experiments also highlighted that there were no significant benefits to combining the curriculum with curiosity and it was actually shown to decrease the generalisation capabilities of agents. Detailed results for each environment can be found in Appendix A. A further benefit is that the curriculum removes the need for any manual reward shaping: a task which is not straightforward when trying to learn general policies that are not optimised to a single environment.

We performed trajectory analysis to identify limitations of the curriculum. This highlighted a memory issue whereby agents were found to continuously move in a repeating pattern and also sometimes got stuck in local optima. We propose increasing the memory of agents through stacking more observations together or by using recurrent architectures. However, this is not a limitation of the curriculum but rather a general training limitation. While the results indicate that there is a transfer of knowledge to unseen environment, there is still a considerable gap between training and testing performance. We proposed various areas for improving and fine-tuning

the curriculum in order to increase the generalisation capabilities of agents, such as procedurally generating environments to introduce more diversity in training.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

This work investigated both policy optimisation and generalisation in sparse reward environments within the domain of navigation. A custom suite of sparse reward navigation environments was designed where an agent needs to learn to navigate from its starting point, past obstacles, to a distant target in the shortest possible time.

A novel approach, *Directed Curiosity* was presented that engineers a reward function through computing a weighted sum of curiosity-driven exploration [58] and distance-based reward-shaping. In the policy optimisation experiments, where agents attempted to learn the optimal path to a target in five different environments, *Directed Curiosity* was evaluated against agents trained with a reward function of only curiosity and only distance-based shaped rewards. Due to the greedy nature of the shaped rewards, the agents that were trained with the function were unable to navigate to targets in the environments with obstacles. The curiosity agents exhibited improved results but in the three environments with obstacles, they failed to converge to optimal polices on all runs, due to the lack of reward feedback to direct agents to the target. *Directed Curiosity* enabled agents to navigate to targets more often and also resulted in shorter training times.

Directed Curiosity is therefore useful for learning specialist policies that are opti-

mised for a specific environment, however, it is not suited for policy generalisation. This is due to the limitations with the shaped reward function. Furthermore, engineering a shaped reward function is very challenging when there are multiple environments since optimising to a particular environment leads to poor performance in other environments.

A manually-designed training curriculum was designed to improve policy generalisation in sparse reward navigation environments i.e. to learn the more difficult task of finding targets in previously unseen environments with different obstacle configurations. The curriculum was evaluated against agents trained using a curiosity reward function and a hybrid approach that combined curiosity with the curriculum. The results of the experiments highlighted showed generalisation: the curriculum enables agents to find targets in more testing environments, including some with completely new environment characteristics.

Even though the curiosity agents performed better than the curriculum agents in a few environments, their performance was more erratic i.e. they sometimes performed optimally and sometimes poorly within the same environments. The curriculum allowed for more robust policies while also decreasing training times and eliminating the need for manual reward shaping.

Combining curiosity with the curriculum provided no meaningful benefits: the training performance was very similar to the curriculum agent but it resulted in inferior policy generalisation.

6.2 Future Work

In future work, we wish to investigate more flexible reward shaping methods to bypass the limitations of the current greedy approach, as well as a more intuitive means of combining intrinsic and extrinsic rewards such as automatically adapting the reward strengths as required.

Since *Directed Curiosity* is based on intelligent exploration, it would be interesting to apply the algorithm in domains other than navigation, by investigating alternative

shaped reward signals and exploration strategies. Another interesting direction is to adapt the algorithm to difficult settings such as environments with multiple targets agents.

Trajectory analysis (assessing the movement paths of the trained agents) was used to identify limitations in the algorithms and highlighted a memory issue whereby agents were found to continuously move in a repeating pattern and also sometimes got stuck in local optimums. In order to address these limitations, we wish to investigate a means of increasing the memory of the agents such as stacking more observations together or by introducing a recurrent architecture. Another interesting direction is to perform further large scale analysis by increasing the number of testing environments, either manually or by procedurally generating them [13].

Bibliography

[1] Centre for high performance computing. `https://www.chpc.ac.za/`.

[2] Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. Learning to Generalize from Sparse and Underspecified Rewards. In *International Conference on Machine Learning*, pages 130–140. PMLR, May 2019. ISSN: 2640-3498.

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[4] Saurabh Arora and Prashant Doshi. A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress. *arXiv:1806.06877 [cs, stat]*, June 2018. arXiv: 1806.06877.

[5] Babak Badnava and Nasser Mozayani. A new Potential-Based Reward Shaping for Reinforcement Learning Agent. *arXiv:1902.06239 [cs]*, May 2019. arXiv: 1902.06239.

[6] Andrea Bassich and Daniel Kudenko. Continuous Curriculum Learning for Reinforcement Learning. In *Proceedings of the 2nd Scaling-Up Reinforcement Learning (SURL) Workshop*, page 7, 2019.

[7] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[8] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, Montreal, Quebec, Canada, June 2009. Association for Computing Machinery.

[10] Ronen I. Brafman and Moshe Tennenholtz. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[11] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-Scale Study of Curiosity-Driven Learning. In *International Conference on Learning Representations*, 2019.

[12] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations (ICLR 2019)*, pages 1–17, May 2019.

[13] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *International Conference on Machine Learning*, pages 2048–2056. PMLR, November 2020. ISSN: 2640-3498.

[14] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying Generalization in Reinforcement Learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, May 2019. ISSN: 2640-3498.

[15] Sam Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 433–440, Richland, SC, June 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[16] Daniel Dewey. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*, 2014.

[17] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. In *International Conference on Machine Learning*, pages 1329–1338, June 2016. ISSN: 1938-7228 Section: Machine Learning.

[18] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, November 2018.

[19] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Machine Learning*, 2019.

[20] Jesse Farebrother, Marlos C. Machado, and Michael Bowling. Generalization and Regularization in DQN. *arXiv:1810.00123 [cs, stat]*, January 2020. arXiv: 1810.00123.

[21] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic Goal Generation for Reinforcement Learning Agents. In *International Conference on Machine Learning*, pages 1515–1528, July 2018. ISSN: 1938-7228 Section: Machine Learning.

[22] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse Curriculum Generation for Reinforcement Learning. In *Conference on Robot Learning*, Proceedings of Machine Learning Research, pages 482–495. PMLR, October 2017. ISSN: 2640-3498.

[23] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, December 2018. Publisher: Now Publishers, Inc.

[24] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. META LEARNING SHARED HIERARCHIES. In *International Conference on Learning Representations*, February 2018.

[25] Justin Fu, John D. Co-Reyes, and Sergey Levine. EX2: Exploration with Exemplar Models for Deep Reinforcement Learning. *arXiv:1703.01260 [cs]*, May 2017. arXiv: 1703.01260.

[26] Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall Traces: Backtracking Models for Efficient Reinforcement Learning. In *International Conference on Machine Learning*, September 2018.

[27] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1311–1320, Sydney, NSW, Australia, August 2017. JMLR.org.

[28] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[29] Guy Hacohen and Daphna Weinshall. On The Power of Curriculum Learning in Training Deep Networks. In *International Conference on Machine Learning*, pages 2535–2544. PMLR, May 2019. ISSN: 2640-3498.

[30] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an Embedding Space for Transferable Robot Skills. February 2018.

[31] Bernhard Hengst. Hierarchical Reinforcement Learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 495–502. Springer US, Boston, MA, 2010.

[32] Ionel-Alexandru Hosu and Traian Rebedea. Playing Atari Games with Deep Reinforcement Learning and Human Checkpoint Replay. *arXiv:1607.05077 [cs]*, July 2016. arXiv: 1607.05077.

[33] Rein Houthooft, Xi Chen, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational Information Maximizing Exploration. *Advances in Neural Information Processing Systems*, 29:1109–1117, 2016.

[34] Ahmed Hussein, Eyad Elyan, Mohamed Medhat Gaber, and Chrisina Jayne. Deep reward shaping from demonstrations. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 510–517. IEEE, 2017.

[35] Asad Jeewa, Anban Pillay, and Edgar Jembere. Directed curiosity-driven exploration in hard exploration, sparse reward environments. In Marelie H. Davel and Etienne Barnard, editors, *Proceedings of the South African Forum for Artificial Intelligence Research, Cape Town, South Africa, 4-6 December, 2019*, volume 2540 of *CEUR Workshop Proceedings*, pages 12–24. CEUR-WS.org, 2019.

[36] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. Self-Paced Curriculum Learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, February 2015.

[37] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A General Platform for Intelligent Agents. *arXiv:1809.02627 [cs, stat]*, September 2018. arXiv: 1809.02627.

[38] Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 2684–2691, Macao, China, August 2019. International Joint Conferences on Artificial Intelligence Organization.

[39] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. *arXiv:1806.10729 [cs, stat]*, November 2018. arXiv: 1806.10729.

[40] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy Optimization with Demonstrations. In *International Conference on Machine Learning*, pages 2469–2478, July 2018.

[41] Andrej Karpathy and Michiel van de Panne. Curriculum Learning for Motor Skills. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Leila Kosseim, and Diana Inkpen, editors, *Advances in Artificial Intelligence*, volume 7310, pages 325–330. Springer

Berlin Heidelberg, Berlin, Heidelberg, 2012. Series Title: Lecture Notes in Computer Science.

[42] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

[43] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, September 2016. ISSN: 2325-4289.

[44] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. PCGRL: Procedural Content Generation via Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1):95–101, October 2020. Number: 1.

[45] Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2140–2146, San Francisco, California, USA, February 2017. AAAI Press.

[46] Ofir Marom and Benjamin Rosman. Belief reward shaping in reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[47] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-Student Curriculum Learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–9, 2019. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[48] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[49] Steven D. Morad, Roberto Mecca, Rudra P. K. Poudel, Stephan Liwicki, and Roberto Cipolla. Embodied Visual Navigation with Automatic Curriculum Learning in Real Environments. *arXiv:2009.05429 [cs]*, September 2020. arXiv: 2009.05429.

[50] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and P. Abbeel. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299, May 2018. ISSN: 2577-087X.

[51] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum Learning for Reinforcement Learning Domains:

A Framework and Survey. *arXiv:2003.04960 [cs, stat]*, March 2020. arXiv: 2003.04960.

[52] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. Source Task Creation for Curriculum Learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 566–574, Richland, SC, May 2016. International Foundation for Autonomous Agents and Multiagent Systems.

[53] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[54] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of Memory, Active Perception, and Action in Minecraft. In *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 2790–2799, New York, New York, USA, June 2016. PMLR.

[55] Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2721–2730, Sydney, NSW, Australia, August 2017. JMLR.org.

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

[57] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing Generalization in Deep Reinforcement Learning. *arXiv:1810.12282 [cs, stat]*, March 2019. arXiv: 1810.12282.

[58] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 488–489, Honolulu, HI, USA, July 2017. IEEE.

[59] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient Deep Reinforcement Learning for Dexterous Manipulation. *arXiv:1704.03073 [cs]*, April 2017. arXiv: 1704.03073.

[60] Rémy Portelas, Cédric Colas, Pierre-Yves Oudeyer, Katja Hofmann, and Lilian Weng. Automatic Curriculum Learning For Deep RL: A Short Survey. volume 5, pages 4819–4825, July 2020. ISSN: 1045-0823.

[61] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for Activation Functions. *arXiv:1710.05941 [cs]*, October 2017. arXiv: 1710.05941.

[62] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, September 2016. arXiv: 1606.04671.

[63] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Machine Learning*, February 2018.

[64] Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphael Marinier, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic Curiosity through Reachability. In *International Conference on Machine Learning*, September 2018.

[65] Jürgen Schmidhuber. PowerPlay: Training an Increasingly General Problem Solver by Continually Searching for the Simplest Still Unsolvable Problem. *Frontiers in Psychology*, 4, June 2013.

[66] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, June 2015. ISSN: 1938-7228.

[67] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, July 2017. arXiv: 1707.06347.

[68] Halit Bener Suay, Tim Brys, Matthew E. Taylor, and Sonia Chernova. Reward Shaping by Demonstration. In *Proceedings of the Multi-Disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2015.

[69] Halit Bener Suay, Tim Brys, Matthew E. Taylor, and Sonia Chernova. Learning from Demonstration for Shaping through Inverse Reinforcement Learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 429–437, Richland, SC, May 2016. International Foundation for Autonomous Agents and Multiagent Systems.

[70] Kaushik Subramanian, Charles L. Isbell Jr, and Andrea L. Thomaz. Exploration from demonstration for interactive reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 447–456. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[71] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. February 2018.

[72] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, second edition: An Introduction.* MIT Press, November 2018. Google-Books-ID: uWV0DwAAQBAJ.

[73] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. Automatic Curriculum Graph Generation for Reinforcement Learning Agents. In *Thirty-First AAAI Conference on Artificial Intelligence*, February 2017.

[74] Csaba Szepesvári. *Algorithms for Reinforcement Learning*, volume 4. January 2010. Journal Abbreviation: Synthesis Lectures on Artificial Intelligence and Machine Learning Publication Title: Synthesis Lectures on Artificial Intelligence and Machine Learning.

[75] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2753–2762. Curran Associates, Inc., 2017.

[76] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. *arXiv:1604.07255 [cs]*, November 2016. arXiv: 1604.07255.

[77] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 1553–1561, San Francisco, California, USA, February 2017. AAAI Press.

[78] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards. pages 10376–10386, 2019.

[79] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping Your Distance: Solving Sparse Reward Tasks Using Self-Balancing Shaped Rewards. In *Advances in Neural Information Processing Systems*, pages 10376–10386, 2019.

[80] Aaron Tucker, Adam Gleave, and Stuart Russell. Inverse reinforcement learning for video games. *arXiv:1810.10593 [cs, stat]*, October 2018. arXiv: 1810.10593.

[81] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. *arXiv:1707.08817 [cs]*, July 2017. arXiv: 1707.08817.

[82] Eric Wiewiora, Garrison W. Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 792–799, 2003.

[83] Sam Witty. Measuring and Characterizing Generalization in Deep Reinforcement Learning. *CoRR*, abs/1812.02868, 2018.

[84] Chang Ye, Ahmed Khalifa, Phillip Bontrager, and Julian Togelius. Rotation, Translation, and Cropping for Zero-Shot Generalization. In *2020 IEEE Conference on Games (CoG)*, pages 57–64, August 2020. ISSN: 2325-4289.

[85] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Relational Deep Reinforcement Learning. *arXiv:1806.01830 [cs, stat]*, June 2018. arXiv: 1806.01830.

[86] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning. *arXiv:1806.07937 [cs, stat]*, June 2018. arXiv: 1806.07937.

[87] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A Study on Overfitting in Deep Reinforcement Learning. *arXiv:1804.06893 [cs, stat]*, April 2018. arXiv: 1804.06893.

[88] Oleksii Zhelo, Jingwei Zhang, Lei Tai, Ming Liu, and Wolfram Burgard. Curiosity-driven Exploration for Mapless Navigation with Deep Reinforcement Learning. *arXiv:1804.00456 [cs]*, May 2018. arXiv: 1804.00456.

Appendix A

Additional Results

A.1 Policy Generalisation

This chapter depicts detailed results of the policy generalisation experiments defined in Section 4.4. The mean reward and the standard deviation is depicted for each algorithm (curiosity, curriculum and a hybrid of both) in each individual environment. "Maze 1" refers to the left-most image within a specific category of environments.

A.1.1 Standard Training Environments

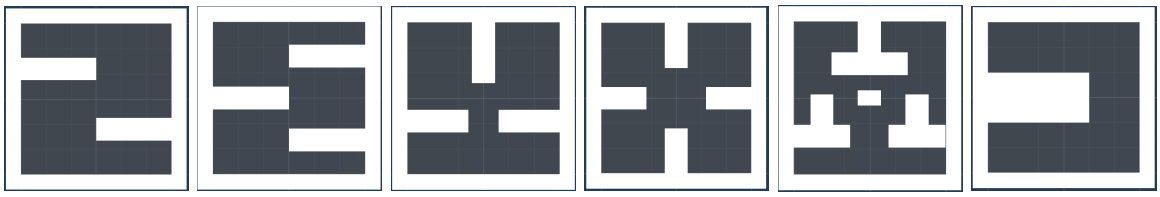


Figure A-1: Standard training mazes

Table A.1: Results for all algorithms in sparse versions of each of the standard training mazes.

Environment \ Algorithm	Curiosity	Hybrid	Curriculum
Maze 1 (Figure A-1)	0.962 ± 0.002	0.964 ± 0.001	0.958 ± 0.001
Maze 2	0.954 ± 0.003	0.957 ± 0.003	0.955 ± 0.000
Maze 3	0.976 ± 0.000	0.975 ± 0.000	0.975 ± 0.000
Maze 4	0.977 ± 0.002	0.979 ± 0.000	0.978 ± 0.001
Maze 5	0.974 ± 0.000	0.974 ± 0.000	0.974 ± 0.000
Maze 6	0.963 ± 0.000	0.963 ± 0.000	0.961 ± 0.001

A.1.2 Difficult Training Environments

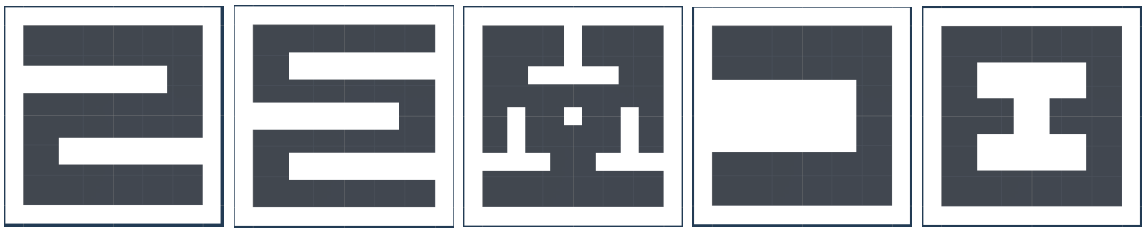


Figure A-2: Difficult training mazes

Table A.2: Results for all algorithms in sparse versions of each of the difficult training mazes.

Environment \ Algorithm	Curiosity	Hybrid	Curriculum
Maze 1 (Figure A-2)	0.934 ± 0.062	0.969 ± 0.000	0.952 ± 0.017
Maze 2	0.937 ± 0.001	0.935 ± 0.005	0.935 ± 0.001
Maze 3	0.911 ± 0.004	0.910 ± 0.017	0.918 ± 0.007
Maze 4	0.975 ± 0.000	0.974 ± 0.000	0.975 ± 0.001

Continued on next page

Table A.2 – *Continued from previous page*

Environment \ Algorithm	Curiosity	Hybrid	Curriculum
Maze 5	0.954 ± 0.001	0.955 ± 0.00	0.954 ± 0.000

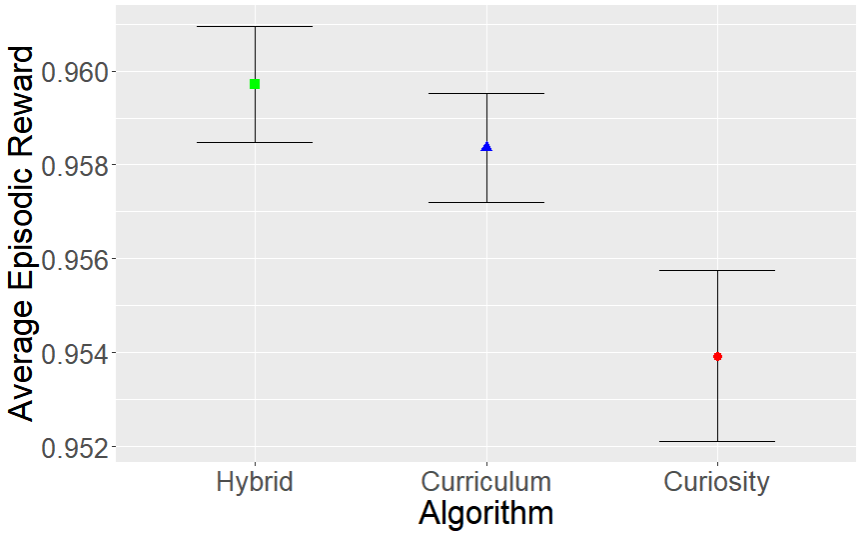


Figure A-3: Average rewards in sparse versions of the training environments (magnified).

A.1.3 Standard Orientation Environments

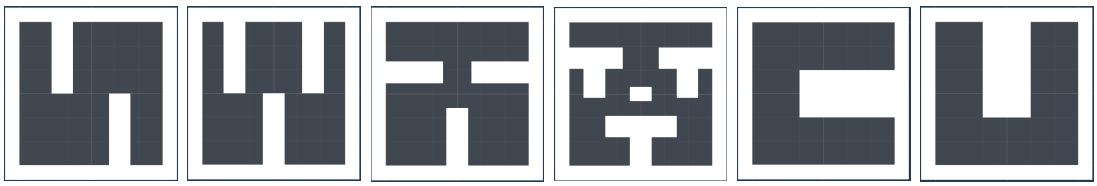


Figure A-4: Standard Orientation testing mazes

Table A.3: Results for all algorithms in each of the Standard Orientation testing mazes.

Environment \ Algorithm	Curiosity	Hybrid	Curriculum
Maze 1(Figure A-4)	0.920 ± 0.040	0.930 ± 0.035	-0.962 ± 0.008
Maze 2	-0.961 ± 0.008	-0.965 ± 0.007	-0.925 ± 0.221
Maze 3	0.973 ± 0.001	0.954 ± 0.022	0.976 ± 0.000
Maze 4	0.161 ± 0.953	0.955 ± 0.011	0.973 ± 0.002
Maze 5	-0.962 ± 0.009	-0.964 ± 0.007	0.894 ± 0.072
Maze 6	0.962 ± 0.001	0.952 ± 0.020	0.954 ± 0.019

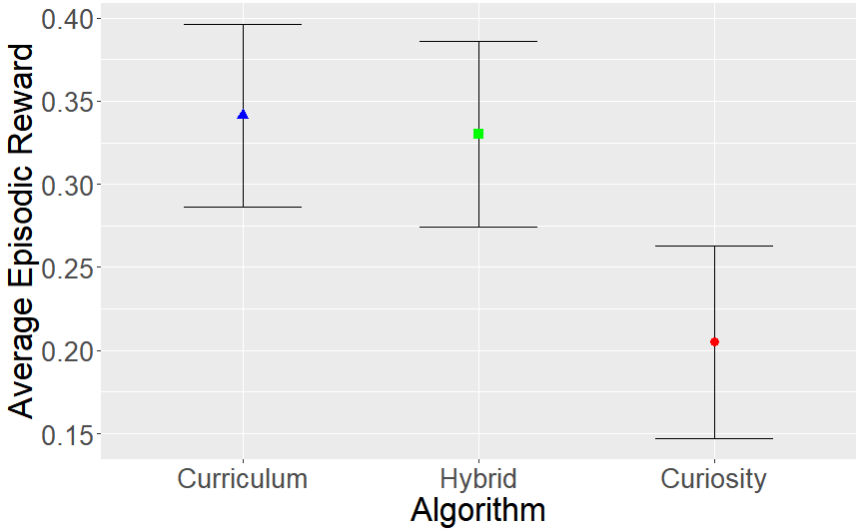
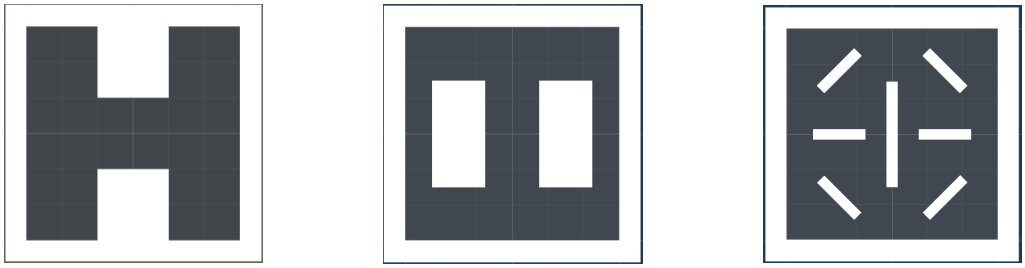


Figure A-5: Average rewards in the Standard Orientation mazes (magnified).

A.1.4 Standard New Environments



(a) H Maze (b) Multi-Path Maze (c) Slim Maze

Figure A-6: Standard New testing mazes

Table A.4: Results for all algorithms in each of the Standard New testing mazes.

Environment \ Algorithm	Curiosity	Hybrid	Curriculum
H Maze (Figure A-6a)	0.970 ± 0.009	0.929 ± 0.044	0.854 ± 0.108
Multi-Path Maze	0.928 ± 0.022	0.938 ± 0.015	0.958 ± 0.001
Slim Maze	0.496 ± 0.752	0.964 ± 0.006	0.779 ± 0.239

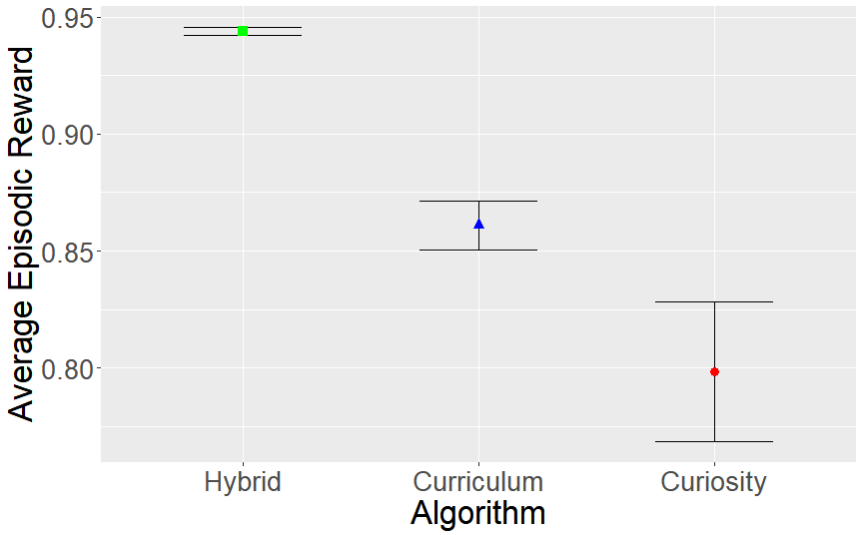


Figure A-7: Average rewards in the Standard New mazes (magnified).

A.1.5 Difficult Orientation Environments

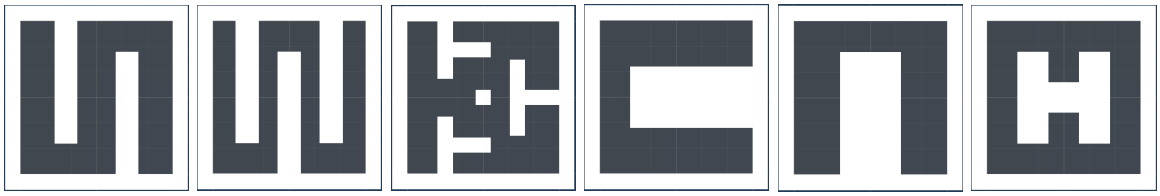


Figure A-8: Difficult Orientation testing mazes

Table A.5: Results for all algorithms in each of the Difficult Orientation testing mazes.

Environment \ Algorithm	Curiosity	Hybrid	Curriculum
Maze 1 (Figure A-8)	-0.818 ± 0.452	-0.961 ± 0.010	-0.967 ± 0.009
Maze 2	-0.962 ± 0.008	-0.961 ± 0.008	-0.967 ± 0.008
Maze 3	-0.963 ± 0.008	-0.558 ± 0.787	0.853 ± 0.130
Maze 4	-0.962 ± 0.009	-0.961 ± 0.009	0.953 ± 0.003
Maze 5	0.900 ± 0.046	-0.298 ± 0.774	-0.968 ± 0.006
Maze 6	-0.962 ± 0.008	-0.960 ± 0.008	-0.967 ± 0.008

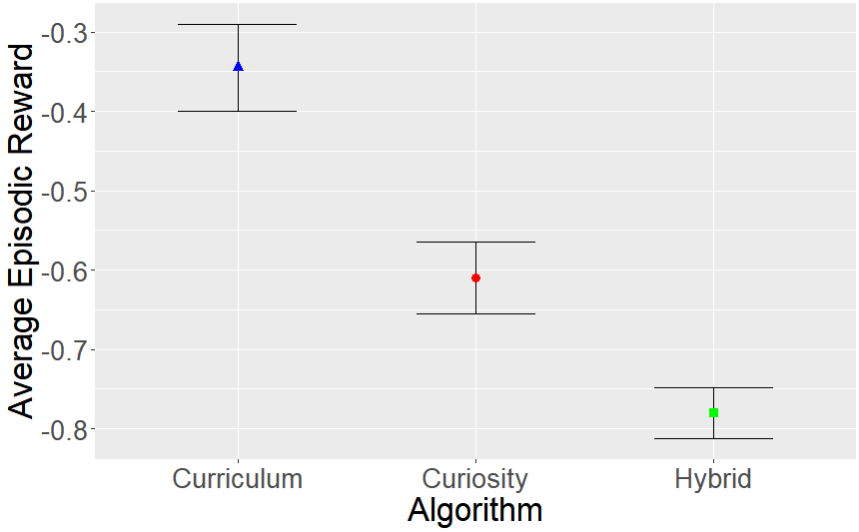
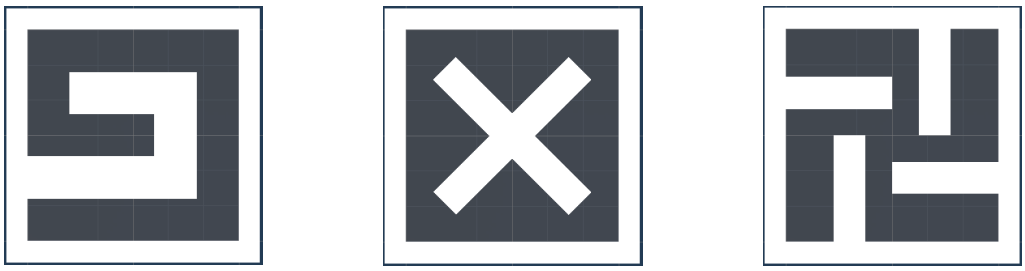


Figure A-9: Average rewards in the Difficult Orientation mazes (magnified).

A.1.6 Difficult New Environments



(a) Spiral Maze (b) X Maze (c) Overlap Maze

Figure A-10: Difficult New testing mazes

Table A.6: Results for all algorithms in each of the Difficult New testing mazes.

Environment \ Algorithm	Curiosity	Hybrid	Curriculum
Spiral Maze (Figure A-10a)	-0.965 ± 0.009	-0.966 ± 0.008	**0.794 ± 0.138**
X Maze	-0.966 ± 0.007	-0.965 ± 0.008	-0.963 ± 0.009
Overlap Maze	-0.500 ± 0.759	-0.745 ± 0.540	-0.923 ± 0.254

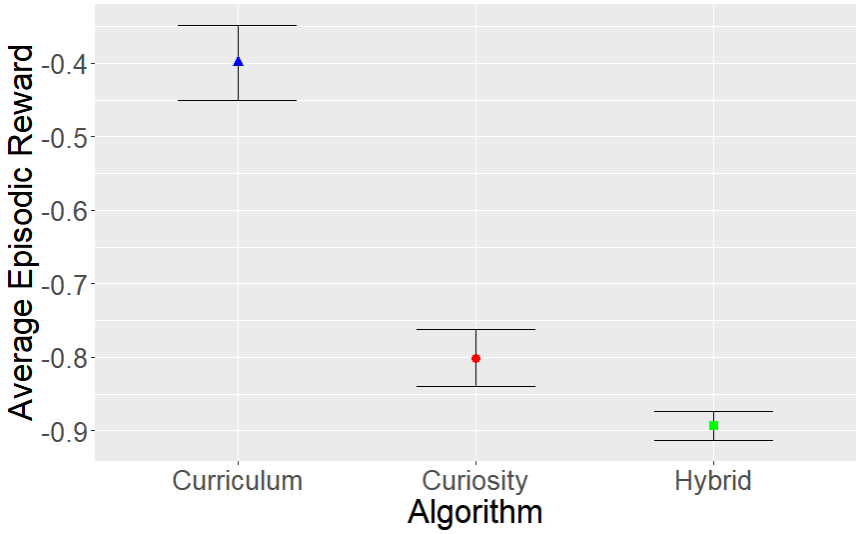


Figure A-11: Average rewards in the Difficult New mazes (magnified).

Appendix B

Learning to Generalise in Sparse Reward Navigation Environments

Asad Jeewa[1,2(✉)], Anban Pillay[1,2], and Edgar Jembere[1,2]

[1] School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Westville 4000, South Africa
asad.jeewa@gmail.com, {pillayw4,jemberee}@ukzn.ac.za
[2] Centre for Aritificial Intelligence Research, Pretoria, South Africa

Abstract. It is customary for RL agents to use the same environments for both training and testing. This causes the agents to learn specialist policies that fail to generalise even when small changes are made to the training environment. The generalisation problem is further compounded in sparse reward environments. This work evaluates the efficacy of curriculum learning for improving generalisation in sparse reward navigation environments: we present a manually designed training curriculum and use it to train agents to navigate past obstacles to distant targets, across several hand-crafted maze environments. The curriculum is evaluated against curiosity-driven exploration and a hybrid of the two algorithms, in terms of both training and testing performance. Using the curriculum resulted in better generalisation: agents were able to find targets in more testing environments, including some with completely new environment characteristics. It also resulted in decreased training times and eliminated the need for any reward shaping. Combining the two approaches did not provide any meaningful benefits and resulted in inferior policy generalisation.

AQ1
AQ2

Keywords: Generalisation · Curriculum learning · Sparse rewards · Navigation

1 Introduction

A fundamental challenge in reinforcement learning (RL) is that of generalisation [7]. It is customary for RL agents to use the same environments for both training and testing [8], as is the case for the Arcade Learning Environment [3], the classic RL benchmark. Agents therefore exhibit breakthrough results on very specific tasks but fail to generalise beyond the training environment [28]. Making small changes to the environment or task often leads to a dramatic decrease in performance [41,42]. This is because agents tend to memorise action sequences and therefore overfit to the training environments [7].

The generalisation problem is compounded in sparse reward environments. RL agents learn behaviour based solely on rewards received through interactions

A. Gerber (Ed.): SACAIR 2020, CCIS 1342, pp. 1–16, 2020.
https://doi.org/10.1007/978-3-030-66151-9_6

with an environment [31]. However, many environments have extrinsic rewards that are sparsely distributed, meaning that the environments does not return any positive or negative feedback to the agent on most timesteps. These environments are prevalent in the real-world [29] and training RL agents in them remains a major challenge [2]. There are various novel approaches to learning in these environments [2] but they tend to emphasise learning specialist polices that fail to generalise to unseen testing environments [7].

This research focuses on policy generalisation in sparse reward navigation environments. Policy generalisation refers to the extent to which a policy transfers to unseen environments within the same domain [8] without any additional training or fine-tuning. This is a difficult task since it is only possible for agents to learn on a small subset of possible states but it is desirable that they should be able to generalise and produce a good approximation over a larger state space [38]. In this work, agents are required to learn to navigate to distant targets across multiple environments, with different characteristics or obstacle configurations.

This work focuses on two approaches. The first technique is curriculum learning. When it is difficult for an agent to learn a task directly, a training curriculum can be designed to gradually increase an agent's knowledge over time. The curriculum imposes an order on training [14]: the agent is trained on a series of simpler tasks that progressively gets more difficult [25]. This enables it to learn "skills" that can be transferred to solve difficult tasks [25]. In this manner, the curriculum can be used to bypass the sparse rewards problem [12]. Curriculum learning has been shown to decrease training times as well as improve generalisation [4, 12].

The second approach introduces intrinsic rewards to augment sparse extrinsic rewards. Intrinsic rewards are generated by the agent itself, instead of relying on feedback from the environment. Curiosity is a type of intrinsic reward that encourages an agent to find "novel" states [29] and has been used to learn policies that generalise to unseen environments.

In this research, we investigate the problem of generalisation in sparse reward navigation environments by evaluating the efficacy of curriculum learning for improving generalisation in this domain. A manually-designed curriculum for sparse reward navigation environments is presented and used to train agents in a suite of hand-crafted environments. Both training and testing performance of the curriculum is empirically compared and contrasted to two baseline algorithms: curiosity-driven exploration [29] and a hybrid approach that combines the curriculum with curiosity. The policies are evaluated in multiple testing environments that are either variations of the training environments or include entirely new characteristics.

The task, algorithms and environments are formally defined in Sect. 3. The benefits and limitations of the curriculum are discussed in Sect. 4: using the curriculum resulted in polices that generalised better than curiosity as well as decreased training times. Section 5 summarises the findings and discusses directions for future work.

2 Related Work

Generalisation remains a fundamental RL problems since agents tend to memorise trajectories from their training environments instead of learning transferable skills [7]. Classic RL benchmarks like the Arcade Learning Environment (ALE) [3] focus on creating specialist agents that perform well in a single environment. New benchmarks have been proposed to focus research on generalisation. The ProcGen Benchmark [7] uses procedural generation to generate new environments. The inherent diversity in the generated environments demands that agents learn robust polices in order to succeed. A similar framework is presented in [19] with larger scale three-dimensional environments.

Justesen et al. [20] however, highlighted limitations of procedural generation: it is difficult to automatically scale the difficulty of the task [20] and the distribution of the procedurally generated environments is often different to that of human-generated environments. Procedurally generating environments may lead to overfitting to the distribution of the generated environments [20]. A novel approach that uses reinforcement learning to learn a policy for generating environments shows promising results in [23].

Our work is inspired by Savinov et al. [32]. The authors emphasised the need for separate training and testing environments and investigated generalisation in custom maze environments with random goal placements. The aims of the study were different but the principles were incorporated into the curriculum defined in Subsect. 3.3. Similar findings were highlighted in other studies [8,42].

Curriculum learning was shown to decrease training times and improve generalisation across multiple common datasets in [4]. The main idea is to split a complex task into smaller, easier-to-solve sub-problems and controlling the curriculum to ensure that the task is never too difficult for the agent [17]. Previous work manually generated training curricula for various tasks [22,34]. A limitation of this approach is the requirement of expert domain knowledge [39]. Various studies attempted to alleviate this problem by presenting novel techniques for automatically generating a curriculum [12,24,39]. Florensa et al. [12] presented a method for automatically generating a curriculum that exhibited promising results in sparse reward navigation environments. The maze environments from the study have been incorporated into this study. The curriculum in this work is manually designed though only general concepts, such as environment size and obstacle configuration, were varied so as to ensure it did not require significant fine-tuning or expert knowledge.

Curriculum learning is an implicit form of generalisation [4]. Closely related to curriculum learning is hierarchical reinforcement learning. Tessler et al. [40] presented a framework that enabled agents to transfer "skills" learnt from easy sub-tasks to difficult tasks requiring multiple skills. Agents learnt "high-level" actions that pertain to walking and movement and used these skills to learn difficult navigation tasks faster in [13]. Our curriculum has been designed to implicitly learn in this manner since there are no obstacles in the early stages of training, thereby allowing agents to focus on locomotion.

The sparse reward problem is well-studied in reinforcement learning. Many novel approaches emphasise learning specialist polices and do not focus on generalisation [7]. Reward shaping augments the reward signal with additional rewards to enable learning in sparse reward environments. It can have a detrimental effect on training if it is used incorrectly and can change the optimal policy or the definition of the task [9,26,28]. Manually engineering reward functions for each new environment is difficult [9,15]. Alternatively, reward functions were recovered from demonstrations in [15,37]. Shaped rewards can result in specialist policies that generalise poorly [15]. The problem was investigated in [16] where agents learnt polices that were optimised for single training environments. A major benefit of our curriculum is that it does not require any reward shaping.

An alternative to "shaping" an extrinsic reward is to supplement it with intrinsic rewards [27] such as curiosity. Curiosity-Driven Exploration by Self-Supervised Prediction [29] formally defined a framework for training curious agents. Curiosity empowers the agent by giving it the capability of exploration, enabling it to reach far away states that contain extrinsic rewards. A well-known limitation of the approach is that agents often find a source of randomness in an environment that allows it to inadvertently satiate its curiosity [5]. There are various other novel approaches [6,33].

Curiosity has been chosen as a baseline as it has shown promising generalisation capabilities in previous studies [5,29]. Agents struggled to generalise to environments with different textures in [5,29]. This is not relevant to this study since agents observations are vector rather than visual representations (see Subsect. 3.1).

To our knowledge, curriculum learning has not been evaluated extensively with regards to generalisation in sparse reward navigation environments.

3 Methodology

3.1 The Task

The goal of the agent is to navigate from its starting point to a fixed distant target, with obstacles or walls placed along its route. The agent is required to learn foresight: it needs to learn to move further away from the target in the present, in order to find the target in the future. The task is a variation of the classic point-mass navigation task in various studies [10,11]. We consider an agent interacting with an environment in discrete time steps. At each time step t, the agent gets an observation o_t of the environment and then takes an action a_t from a set of actions A.

The observation set O comprises the coordinates of the agent's current position, the coordinates of the target, the distance to the goal and rays that extend in 8 directions, at 45° intervals. These short rays provide essential feedback to the agent by enabling it to detect walls and targets that are in its vicinity and therefore adapt its policy accordingly.

The rays take on additional importance when agents are placed in previously unseen environments since they enable the agents to learn robust policies: when

an agent detects an obstacle in its vicinity, it needs to learn to move away from the obstacle, in the direction of an open path. If an agent executes memorised actions, it will move directly into walls and never reach its destination.

The ray length was tuned to balance the difficulty of the task: if the rays are too long, the agent unrealistically detects objects that are far away but if it is too short, the agent is unable to detect anything except that which is immediately in front of it. This is analogous to the field of view. The observations were stacked to equip the agents with a small memory of the immediate past. The previous ten observations were stored at any given time.

The action set a_t allows the agent to move in eight directions: forwards, backwards, sideways as well as diagonally, unlike the standard Gridworld task [42].

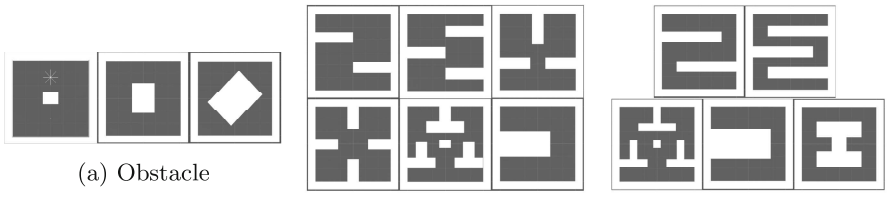
By default, before any training modifications are made, the environments are all sparse reward environments since the agent only receives a +1 reward for finding the target. The starting positions of the agent and the target are far away from each other, on different ends of the environment. The agents do not receive any intermediate rewards and incur a small penalty on every timestep, to encourage them to find the target in the shortest possible time.

3.2 Environments

There are multiple environments and each varies in terms of the configuration of walls and obstacles (see Fig. 1). This is to deter agents from learning an optimal policy in one single environment, rather learning the "skill" of finding a target in an arbitrary navigation environment. The predefined environments were carefully designed to represent high-level features or environment characteristics that include dead-ends and multiple paths to the target. We theorise that introducing agents to numerous environment features in training enables them to learn a flexible policy that enables them to find targets when similar features are found in new environments. The environments were divided into a set of training and testing environments. The generalisability of the agents was evaluated in the testing environments.

The training environments were further divided into three categories: *Obstacle* environments (see Fig. 1a) contain only a single obstacle that varies in terms of size and orientation. The sizes range from a scale of 0 to 3 and the orientation is defined as any angle from 0°, in 45° increments. The size of the agent and ray length are also depicted in Fig. 1a to illustrate the scale of the task.

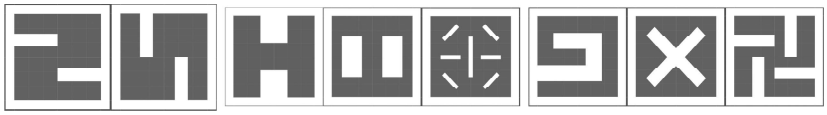
Maze environments have multiple obstacles and were subdivided based on difficulty. There are *Standard* mazes in Fig. 1b and *Difficult* mazes in Fig. 1c. *Difficult* mazes have multiple obstacles that span more than half the width of the entire environment. They also include more complex versions of some of the *Standard* mazes, by manipulating the size of each obstacle in an environment. The "u-maze" from [11] was also incorporated into this group. The difficult mazes were deliberately designed to test the boundaries of the algorithms and to identify limitations.



(a) Obstacle (b) Standard Mazes (c) Difficult Mazes

Fig. 1. Training environments

The testing environments were divided into two categories: *Orientation* and *New*. *Orientation* testing environments were created by rotating the training mazes by 90° and without changing the overall structure of the obstacles. *New* testing environments have different obstacle configurations to the training environments. New features or environment characteristics, such as bottlenecks or repeated obstacles, were incorporated into this group. This allowed us to analyse whether the agents were able to learn advanced skills and further assess the extent of the generalisation. Both these categories were further subdivided into *Standard* and *Difficult* subcategories, as per the definition used for the training environments. An illustration of the *Orientation* environments are shown in Fig. 2a. Both the *Standard New* and *Difficult New* groups, depicted in Fig. 2b and c respectively, contain three mazes each. The "spiral-maze", a commonly used maze seen in [11], was incorporated into the difficult category.



(a) Orientation (b) Standard New (c) Difficult New

Fig. 2. Testing environments

3.3 Algorithms

Curriculum Learning. A curriculum was manually designed to enable agents to learn the task of finding distant targets across multiple sparse reward navigation environments (see Algorithm 1). This is difficult since agents cannot optimise a policy for any specific environments and when the environments are large, with multiple obstacles (the most difficult version of the task), the reward feedback is sparse.

The curriculum has been designed to act as a means of bypassing the sparse rewards problem. It also improves generalisation by exposing agents to a diverse set of environments during training.

Algorithm 1. Manually-Designed Curriculum

Input: Obstacle Environments O, Obstacle Max Scale $S_{obstacle}$, Maze Environments M, Environment Max Scale $S_{environment}$, Reward Threshold $R_{threshold}$, Number Consecutive Episodes $n_{consecutive}$

```
for i ← 1 to S_environment do
    Reset episode count
    r_average = 0
    repeat (for each episode)
        r_average ← average episodic reward from previous n_consecutive episodes
        Sample an obstacle environment from O
        Sample scale from {0, 1, 2, ..., S_obstacle}
        Sample angle from {0°, 45°, 90°, 135°, ..., 315°}
        Sample agent and target starting positions
    until r_average < R_threshold
    Reset episode count
    r_average = 0
    repeat (for each episode)
        r_average ← average episodic reward from previous n_consecutive episodes
        Sample a maze environment from M
        Sample agent and target starting positions
    until r_average < R_threshold
end for
```

Environment parameters are varied over time to control the difficulty of the task to ensure that the current task is never too difficult for the agent. The first parameter is the environment size: decreasing the size, while keeping the agent size and speed the same, decreases the sparsity of rewards since the goal and target are closer to each other in smaller environments. The second parameter is the obstacle configuration, which is varied through changing the number and size of obstacles: either single obstacles or multiple obstacles in a maze-like structure.

In the early stages of training, the environments are small and contain a single obstacle or none at all. This was achieved by assigning O, in Algorithm 1, to the obstacle environments in Fig. 1a. Agents are able to learn how to control themselves by navigating around the environment to nearby targets. When the average reward (over the past 5000 consecutive episodes) reaches a predefined threshold, the difficulty is increased. The first adjustment is to increase the size and number of obstacles, through randomly sampling maze environments from Fig. 1b and in Fig. 1c. When the agent reaches the same predefined reward threshold, the environment size is increased. This two-fold difficulty adjustment keeps occurring until the agent progresses to large maze environments with multiple obstacles. This ensures that the curriculum only progresses when the agent has succeeded in its current task.

Randomly sampling environments is an important aspect of the curriculum. It is also essential that the set of training environments is diverse and incorporates a wide array of obstacle configurations [7]. This deters agents from memorising the dynamics of any particular training environment, instead learning how

to navigate past arbitrary obstacles to find distant targets. This is analogous to supervised learning i.e. training on a diverse training set al.lows for a more generalised model that does not overfit to training data. Specifically, overfitting means memorising a policy that is optimised for the training environments, resulting in poor performance in the testing environments. Similarly, policy memorisation refers to a policy that optimises the dynamics of a particular environment by memorising actions that lead to success, resulting in poor performance even when subtle changes are made to the environment [41].

The maximum environment size ($S_{environment}$ in Algorithm 1) was carefully tuned to ensure that the task is a sparse rewards problem. This was verified by running an agent trained with policy gradient on the sparse reward function (+1 for finding the target), with no exploration strategy, and observing that it was not possible for it to find the target, after a large number of training steps [16].

Inspired by various other work [21,32,42], the last aspect of the curriculum attempts to bypass the sparse rewards problem by "densifying" the training environment. The starting locations of both the agent and target are randomised at the start of every episode. This means that the target is often close to the agent, resulting in frequent feedback that enables meaningful learning. This also encourages the agent to explore different parts of the environments.

Baseline Algorithms. We compare the performance of the curriculum to curiosity-driven exploration defined by Pathak et al. in [29]. This equips the agent with an intrinsic reward that allows it to explore the training environments by seeking "novel" states, thereby gaining an understand of the dynamics of the various environments. A reward is generated through a prediction error: agents are trained to predict the next state as well as actions taken in between states. In this way, the reward only captures surprising states that have come about directly as a result of the agents actions. Curiosity has shown promising generalisation capabilities in previous studies [5,29].

In the curiosity setting, the curriculum defined in Subsect. 3.3 is omitted i.e. the size of the environment is fixed at the largest configuration and a training maze is randomly sampled from Fig. 1b and c on each episode. Training also occurs under the dense reward setting with random target and agent starting locations.

The final approach combines the curiosity reward with the hand-crafted curriculum which we term "Hybrid": Agents are trained using the curriculum from Subsect. 3.3 and the reward function is augmented with a curiosity signal. Both algorithms have been shown to improve generalisation individually [5,6,12] and it was therefore necessary to investigate if there were any merits to combining them.

3.4 Experimental Design

We evaluated the performance of the curriculum by comparing it to curiosity-driven exploration [29] and a hybrid "curiosity-curriculum" approach. All policies

are represented by neural networks with Proximal Policy Optimisation (PPO) [35] being used as an optimiser. PPO is robust [35] and requires lesser hyperparameter tuning when compared to similar methods [5]. However, an arbitrary policy gradient method could have been: the focus of this work is rather on the comparison of different training methods so consistency in the optimisation method is more important.

We defined baseline hyperparameters for each algorithm by training agents in the easiest version of the tasks. The hyperparameters were then carefully tuned and optimised by observing the training process and then tweaking the relevant parameters as required. The networks have two hidden layers, each with 256 units. The swish activation function [30] was used. The learning rate and entropy coefficient were fixed at 3.0×10^{-4} and 0.01 respectively, along with a batch size of 256 and a buffer size of 5120. All agents were trained for 20 million training steps. This was carefully tuned to ensure that the agents had sufficient time to learn. However, it was observed that agents tended to overfit to the training environments when the steps were too high [42]. After tuning the hyperparameters independently for each algorithm, the agents were trained on a cluster of machines on the Centre for High Performance Computing [1], using the Unity ML-Agents platform [18]. The polices for each algorithm were then evaluated against each other. The codebase and further details are available at https://github.com/AsadJeewa/Learning-to-Generalise-in-Sparse-Reward-Navigation-Environments.

4 Results and Discussion

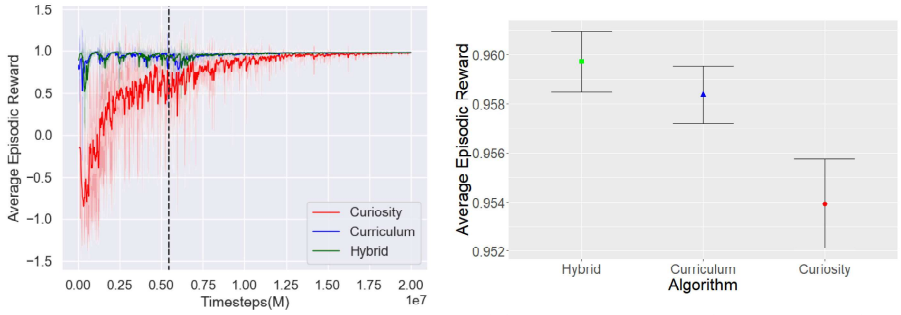
Analysis is performed in three stages: the first stage compares the training performance of each algorithm. Since the agents were trained under a dense rewards setting, with randomised agent and target starting positions for each episode, it is necessary to evaluate the algorithms under a sparse reward setting. This was achieved by positioning the agent and target at distant locations in every training environment, fixed at points that make the task as difficult as possible.

We perform a critical evaluation of the generalisability of each algorithm in the unseen testing environments. The last stage performs trajectory analysis to understand the strengths and limitations of each algorithm. It provides insight into the intricacies of how agents move within different environments.

4.1 Training Performance

The training curves are depicted in Fig. 3a i.e. the average episodic reward of the agents over time, with a smoothing factor of 0.2. For each algorithm, we performed five independent runs and computed the mean learning curve and standard deviation. Twenty independent instances of the environment were used for more efficient data collection during training.

The dashed line depicts the point at which both the curriculum and hybrid agents progressed to the final lesson, which corresponds to the training environments of the curiosity-driven agent.



(a) Learning curves during training

(b) Average rewards (sparse setting)

Fig. 3. Training performance for all algorithms

The curves highlight the benefits of using the curriculum. The blue curve never drops significantly since the agents' task is never too difficult. The curriculum advances quickly in the early stages of training when the task is easier. The sudden drops in reward are indicative of points at which the task is made more difficult but the fact that the curve peaks very quickly thereafter, indicates that knowledge is being transferred between tasks. In all runs, it was noted that the curriculum agent converged significantly faster than the curiosity-driven agent.

A major benefit of the curriculum is that there is no reward shaping necessary. This is due to the manner in which the curriculum was designed that ensures that the agents always receive sufficient reward feedback during training. We performed an empirical investigation into various different shaped rewards and found no performance improvements. Rather, the motivations of the agents became polluted [9,26]. For example, when an agent was rewarded for moving closer to the target, it lacked the foresight to move past obstacles. Shaping rewards also resulted in more specialist policies that work well in some environments, but poorly in others. Reward shaping also requires additional information which may not be available in the real-world.

The curiosity curve shows rewards slowly increasing as training progresses. The hybrid training curve is very similar to the curriculum agent. When the curiosity strength was varied, the curves still followed a similar pattern. This indicates that the curiosity rewards had little effect on the training process when coupled with the curriculum.

Figure 3b illustrates that, for all algorithms, the agents were able to efficiently find the target in all training environments, under the sparse reward setting. All algorithms have an average reward that approaches a maximum possible reward of +1. These results act as a validation of each algorithm since it indicates that all agents have obtained sufficient knowledge of the task and are able to find targets across a diverse set of mazes. This allowed us to perform a fair comparison of the generalisation capabilities of each algorithm in the testing environments. Error bars are depicted with a confidence interval of 95%.

4.2 Generalisability

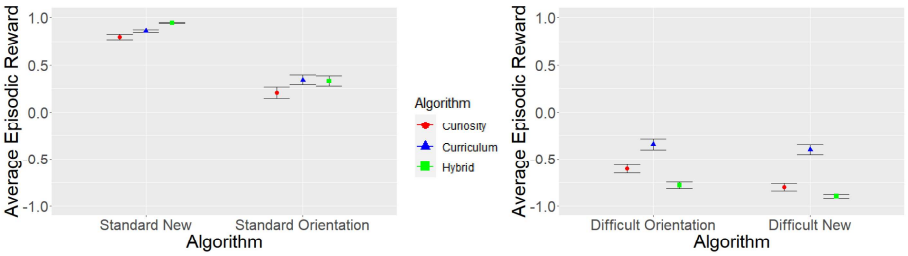
The best performing training run from Subsect. 4.1 was selected for each algorithm. The average reward was then analysed for each of the different groups of testing environments. Each algorithm was run for 1000 episodes, with a random testing environment being sampled at the start of the episode, from the corresponding testing group. This is necessary due to the stochastic nature of the polices: the agents sometimes succeed and fail in the same testing environment. This results in vastly different episodic rewards and a large number of episodes is therefore necessary to stabilise the average rewards.

When analysing the results, there are certain important considerations that need to be made. The performance of each algorithm is often different i.e. agents succeed and fail in different testing environments. There are instances when one algorithm enabled agents to navigate to the target in a short time, but another resulted in agents only finding the target after a large number of episode steps or never at all. We wish to investigate this phenomenon further in future work.

The task is not trivial since it as analogous to placing a human or vehicle in a new environment and only equipping them with information about its current location, destination and the ability to "see" what's around it. It does not have any knowledge of the dynamics of the environment that it is placed in. This means that some "exploration" is necessary and it is expected that agents will move into obstacles as they try to advance towards the goal. It is not possible to solve the generalisation problem completely: it was not expected that the agents would obtain expert performance in the testing environments. The goal is rather to transfer some knowledge that can be reused in the environments.

The policies are used "as-is" and there is no fine-tuning for any of the testing environments, as is the case in other studies [29]. It is definitely possible to improve the results in each testing environment by fine-tuning the policy though that is not the aim of this study. This work rather investigated the extent to which the learned policy generalised.

Lastly, the sample size in the testing environment groups is fairly small. There are only three environments in some groups. In future work, we wish to investigate whether the results hold when increasing the size of the groups.



(a) Standard Mazes (b) Difficult Mazes

Fig. 4. Generalisability in the maze testing environments

The average episodic rewards are in the range $[-1, 1)$. A successful run is one in which agents are able to navigate to the target. The faster an agent finds the target, the higher the reward it receives. An average reward approaching one therefore indicates that the agents successfully found the targets on all runs. A score below one indicates that on most runs, the agents were unable to find the target, across all environments, with zero representing an inflection point.

The results highlight an expected gap between training and testing performance. However, they also indicate that some generalisation has taken place.

Standard Mazes. The experiments that we conducted indicate good performance for all algorithms in the standard mazes. Figure 4a illustrates that the algorithms preformed similarly in the *Standard New* environments. Notably, all the agents were able to consistently navigate to the target in all three environments. This is promising since the obstacle configurations are completely different to the training environments. This indicates that the policy is robust and generalises well (in these environments). On average, the hybrid agents found the targets marginally faster.

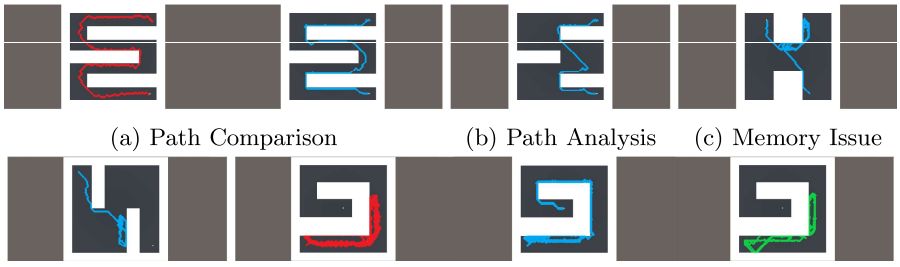
The *Standard Variation* results depict that all algorithms are able to succeed on most runs. The curriculum agent was marginally the most successful across the six environments but the performance is once again similar for all algorithms. It is encouraging that the agents succeed in some environments however, we theorise that the results can be improved by fine-tuning the set of training and testing environments, or procedurally generating training mazes to improve the robustness of the policies.

Difficult Mazes. While the results of the algorithms in the standard mazes showed similar performance, the agents trained using the curriculum performed best in the difficult mazes.

The *Difficult Orientation* results in Fig. 4b indicate that the agents weindicate that the agents were re unable to find the target on most runs. However, some transfer has taken place. The curriculum obtained the highest average reward: the result is statistically significant under a 95% confidence interval. Interestingly, both the curriculum and hybrid agents succeeded in two of the five environments but the hybrid agent took significantly longer to find the targets. The hybrid agent is the worst performing algorithm; this indicates that generalisability decreases significantly as the difficulty of the environments are increased. The performance of the curiosity-driven agent showed limited transfer to the testing environments with agents only succeeding in one environment.

Difficult New experiments show the least transfer, as expected. The curriculum agent is once again the most successful. The nature of the environments mean that agents are able to find the targets on some runs, though not consistently. The most promising result was that the curriculum agent was the only algorithm that succeeded in the "spiral-maze" [11] depicted in Fig. 2c.

4.3 Trajectory Analysis



(a) Path Comparison (b) Path Analysis (c) Memory Issue

(d) Local Optimum (e) Spiral Mazes

Fig. 5. Walkthrough trajectories

We performed trajectory analysis by analysing the movement patterns of the trained agents across the different environments, as per [32]. It was often observed that the curriculum agents tended to move in a more directed manner than the curiosity-driven agents. The curriculum agents also tended to "stick" to the walls for a longer time, using them to guide it to the target. An example of this is depicted in Fig. 5a. The trail of a curiosity-driven agent is shown in red and that of the curriculum agent is in blue. There is further proof of this in Fig. 5b. This figure also highlights common behaviour of the curriculum agents: they initially attempted to move directly towards the goal, along the shortest possible path, but when the agents detected an obstacle, they adapted to move around it. The highlights the robustness of the policy.

A limitation of all the algorithms is that it was sometimes observed that the agents repeatedly move along a similar path and only make slight advancements towards the target, over a long period of time. However, the agents often still find their way to the target, as shown in Fig. 5c. In an attempt to alleviate this problem, we would like to look into different methods for increasing the "memory" of the agents. The number of stacked observations could be increased, so that agents can "remember" more of their previous failures, or recurrent architectures could be used.

Figure 5d shows an example of an environment in which the curriculum agent failed to find the target. The agent was progressing towards the target but then got stuck in a local optimum and kept repeating the same actions, until the maximum episode steps was reached. It is possible that the agents would eventually have found its way to the target. This result points to some memorisation in the policy. We theorise that improving the memory of the agent would also alleviate this problem.

The most promising result is shown in Fig. 5e. The spiral maze is difficult because the agent needs to learn a very specific trajectory in order to find the target. The curriculum agent was the only agent that succeeded in this environment. This further highlights the robustness of the curriculum: it was able to

continuously adapt its actions as it observed the environment. The curiosity and hybrid agents both got stuck in local optima and failed to reach the target.

5 Conclusions and Future Work

We have designed a training curriculum that improves generalisation in sparse reward navigation environments. It was evaluated against a curiosity-based agent [29] and a hybrid of the two algorithms, in a suite of manually-designed navigation environments.

The curriculum agents showed the most promising generalisation results. Agents were able to find targets in more testing environments, including some with completely new environment characteristics. There are certain environments when curiosity performed better than the curriculum agent but the performance of the agents were more erratic i.e. they sometimes performed excellently and sometimes very poorly within the same environments. Curriculum learning proved to be more a robust approach. It also resulted in decreased training times and eliminated the need for any reward shaping.

Combining curiosity with the curriculum provided no meaningful benefits. The training performance was very similar to the curriculum agent and it exhibited inferior policy generalisation in the difficult maze testing environments.

There are limitations to the curriculum, as indicated by the generalisation gap between the training and testing environments. Agents sometimes get stuck in local optimums and also repeated the same movement patterns in an episode. There is some memorisation occurring since the agents perform excellently in the training environments and struggle in some testing environments. However, the results are promising, since it shows clear evidence of knowledge transfer to unseen environments.

In future work, we propose further increasing the diversity in the training environments and fine-tuning the curriculum to further improve the results. We also wish to investigate the effects of increasing the memory of agents to deter them from repeating trajectories in testing environment. Another interesting direction is to perform further large scale analysis of the algorithms by increasing the number of testing environments, either manually or by procedurally generating them [7].

AQ3

References

1. Centre for high performance computing. https://www.chpc.ac.za/
2. Andrychowicz, M., et al.: Hindsight experience replay. In: Advances in Neural Information Processing Systems, pp. 5048–5058 (2017)
3. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: an evaluation platform for general agents. J. Artif. Intell. Res. **47**, 253–279 (2013)
4. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 41–48. Association for Computing Machinery, Montreal (2009). https://doi.org/10.1145/1553374.1553380

5. Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A.A.: Large-scale study of curiosity-driven learning. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=rJNwDjAqYX
6. Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. arXiv preprint arXiv:1810.12894 (2018)
7. Cobbe, K., Hesse, C., Hilton, J., Schulman, J.: Leveraging procedural generation to benchmark reinforcement learning. arXiv preprint arXiv:1912.01588, p. 27 (2019)
8. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. arXiv preprint arXiv:1812.02341, p. 8 (2018)
9. Devlin, S.M., Kudenko, D.: Dynamic potential-based reward shaping (2012). http://eprints.whiterose.ac.uk/75121/
10. Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: International Conference on Machine Learning, pp. 1329–1338 (2016). http://proceedings.mlr.press/v48/duan16.html, ISSN: 1938–7228 Section: Machine Learning
11. Florensa, C., Held, D., Geng, X., Abbeel, P.: Automatic goal generation for reinforcement learning agents. In: International Conference on Machine Learning, pp. 1515–1528 (2018). http://proceedings.mlr.press/v80/florensa18a.html. ISSN: 1938–7228 Section: Machine Learning
12. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., Abbeel, P.: Reverse curriculum generation for reinforcement learning. arXiv:1707.05300 [cs] (2018)
13. Frans, K., Ho, J., Chen, X., Abbeel, P., Schulman, J.: Meta learning shared hierarchies. arXiv:1710.09767 [cs] (2017)
14. Hacohen, G., Weinshall, D.: On the power of curriculum learning in training deep networks. arXiv:1904.03626 [cs, stat] (2019)
15. Hussein, A., Elyan, E., Gaber, M.M., Jayne, C.: Deep reward shaping from demonstrations. In: Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), pp. 510–517. IEEE (2017)
16. Jeewa, A., Pillay, A., Jembere, E.: Directed curiosity-driven exploration in hard exploration, sparse reward environments. In: Davel, M.H., Barnard, E. (eds.) Proceedings of the South African Forum for Artificial Intelligence Research, Cape Town, South Africa, 4–6 December 2019, CEUR Workshop Proceedings, vol. 2540, pp. 12–24. CEUR-WS.org (2019). http://ceur-ws.org/Vol-2540/FAIR2019_paper_42.pdf
17. Jiang, L., Meng, D., Zhao, Q., Shan, S., Hauptmann, A.G.: Self-paced curriculum learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (2015). https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9750
18. Juliani, A., et al.: Unity: a general platform for intelligent agents. arXiv:1809.02627 [cs, stat] (2018)
19. Juliani, A., et al.: Obstacle tower: a generalization challenge in vision, control, and planning. arXiv:1902.01378 [cs] (2019)
20. Justesen, N., Torrado, R.R., Bontrager, P., Khalifa, A., Togelius, J., Risi, S.: Illuminating generalization in deep reinforcement learning through procedural level generation. arXiv:1806.10729 [cs, stat] (2018)
21. Kang, B., Jie, Z., Feng, J.: Policy optimization with demonstrations. In: International Conference on Machine Learning, pp. 2469–2478 (2018). http://proceedings.mlr.press/v80/kang18a.html
22. Karpathy, A., van de Panne, M.: Curriculum learning for motor skills. In: Kosseim, L., Inkpen, D. (eds.) AI 2012. LNCS (LNAI), vol. 7310, pp. 325–330. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30353-1_31

23. Khalifa, A., Bontrager, P., Earle, S., Togelius, J.: PCGRL: Procedural content generation via reinforcement learning. arXiv:2001.09212 [cs, stat] (2020)
24. Matiisen, T., Oliver, A., Cohen, T., Schulman, J.: Teacher-student curriculum learning. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–9 (2019). https://doi.org/10.1109/TNNLS.2019.2934906
25. Narvekar, S., Stone, P.: Learning curriculum policies for reinforcement learning. arXiv:1812.00285 [cs, stat] (2018)
26. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: theory and application to reward shaping. ICML **99**, 278–287 (1999)
27. Oudeyer, P.Y., Kaplan, F.: What is intrinsic motivation? A typology of computational approaches. Front. Neurorobotics **1**, 6 (2009)
28. Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., Song, D.: Assessing generalization in deep reinforcement learning. arXiv:1810.12282 [cs, stat] (2019)
29. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2017), pp. 488–489. IEEE, Honolulu (2017). https://doi.org/10.1109/CVPRW.2017.70, http://ieeexplore.ieee.org/document/8014804/
30. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv:1710.05941 [cs] (2017)
31. Ravishankar, N.R., Vijayakumar, M.V.: Reinforcement learning algorithms: survey and classification. Indian J. Sci. Technol. **10**(1), 1–8 (2017). https://doi.org/10.17485/ijst/2017/v10i1/109385, http://www.indjst.org/index.php/indjst/article/view/109385
32. Savinov, N., Dosovitskiy, A., Koltun, V.: Semi-parametric topological memory for navigation. arXiv:1803.00653 [cs] (2018)
33. Savinov, N., et al.: Episodic curiosity through reachability. arXiv:1810.02274 [cs, stat] (2019)
34. Schmidhuber, J.: POWERPLAY: training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. arXiv:1112.5309 [cs] (2012)
35. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv:1707.06347 [cs] (2017)
36. Suay, H.B., Brys, T.: Learning from demonstration for shaping through inverse reinforcement learning, p. 9 (2016)
37. Suay, H.B., Brys, T., Taylor, M.E., Chernova, S.: Reward shaping by demonstration. In: Proceedings of the Multi-Disciplinary Conference on Reinforcement Learning and Decision Making (RLDM) (2015)
38. Sutton, R.S., Barto, A.G.: Reinforcement Learning, Second Edition: An Introduction. MIT Press, Cambridge (2018). google-Books-ID: uWV0DwAAQBAJ
39. Svetlik, M., Leonetti, M., Sinapov, J., Shah, R., Walker, N., Stone, P.: Automatic curriculum graph generation for reinforcement learning agents. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (2017). https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14961
40. Tessler, C., Givony, S., Zahavy, T., Mankowitz, D.J., Mannor, S.: A deep hierarchical approach to lifelong learning in minecraft. arXiv:1604.07255 [cs] (2016)
41. Ye, C., Khalifa, A., Bontrager, P., Togelius, J.: Rotation, translation, and cropping for zero-shot generalization. arXiv:2001.09908 [cs, stat] (2020)
42. Zhang, C., Vinyals, O., Munos, R., Bengio, S.: A study on overfitting in deep reinforcement learning. arXiv:1804.06893 [cs, stat] (2018)

Appendix C

Directed curiosity-driven exploration in hard exploration, sparse reward environments

Asad Jeewa[0000−0003−4329−8137], Anban Pillay[0000−0001−7160−6972], and Edgar Jembere[0000−0003−1776−1925]

University of KwaZulu-Natal, Westville 4000, South Africa
asad.jeewa@gmail.com
{pillayw4,jemberee}@ukzn.ac.za

Abstract. Training agents in hard exploration, sparse reward environments is a difficult task since the reward feedback is insufficient for meaningful learning. In this work, we propose a new technique, called Directed Curiosity, that is a hybrid of Curiosity-Driven Exploration and distance-based reward shaping. The technique is evaluated in a custom navigation task where an agent tries to learn the shortest path to a distant target, in environments of varying difficulty. The technique is compared to agents trained with only a shaped reward signal, a curiosity signal as well as a sparse reward signal. It is shown that directed curiosity is the most successful in hard exploration environments, with the benefits of the approach being highlighted in environments with numerous obstacles and decision points. The limitations of the shaped reward function are also discussed.

Keywords: Sparse Rewards · Hard Exploration · Curiosity · Reward Shaping · Navigation.

1 Introduction

A reinforcement learning agent learns how to behave based on rewards and punishments it receives through interactions with an environment [18]. The reward signal is the only learning signal that the agent receives [2]. Many environments have extrinsic rewards that are sparsely distributed, meaning that most timesteps do not return any positive or negative feedback. These environments, known as sparse reward environments [12,21], do not provide sufficient feedback for meaningful learning to take place [17]. The most difficult sparse reward environments are those where an agent only receives a reward for completing a task or reaching a goal, meaning that all intermediate steps do not receive rewards. These are referred to as terminal reward environments [7].

Closely related to the sparse rewards problem is the issue of exploration. Exploration algorithms aim to reduce the uncertainty of an agents understanding of its environment [4]. It is not possible for an agent to act optimally until it has sufficiently explored the environment and identified all of the opportunities for reward [24]. An agent may never obtain positive rewards without an intuitive

exploration strategy when rewards are sparse. Hard exploration environments are environments where local exploration strategies such as ϵ-greedy are insufficient [4]. In these environments, the probability of reaching a goal state through local exploration is negligible.

These types of environments are prevalent in the real-world [17] and training reinforcement learning (RL) agents in them forms one of the biggest challenges in the field [1]. This research focuses on learning in hard exploration, terminal reward environments.

A popular approach to learning in these environments is reward shaping, which guides the learning process by augmenting the reward signal with supplemental rewards for intermediate actions that lead to success [15]. This ensures that the agent receives sufficient feedback for learning.

Intrinsic rewards that replace or augment extrinsic rewards is another area of research that has exhibited promising results [4,7,17]. Instead of relying on feedback from the environment, an agent engineers its own rewards. Curiosity is a type of intrinsic reward that encourages an agent to find "novel" states [17].

In this research, we present *Directed Curiosity*: a new technique that hybridises reward shaping and Curiosity-Driven Exploration [17] to allow agents to explore intelligently. The algorithm is defined in Section 3 and the custom navigation environments used for evaluation are described in Section 4. The performance of the algorithm is evaluated by comparing it to its constituent algorithms i.e. agents trained with only the shaped reward and only the curiosity reward. Directed Curiosity is shown to be the most robust technique in Section 5. The environment characteristics that are suited to this technique are highlighted and the limitations of the shaped reward function are also discussed.

2 Related Work

Learning in hard exploration, sparse reward environments is a well-studied area in reinforcement learning. Reward shaping is a popular approach that augments the reward signal with additional rewards to enable learning in sparse reward environments. It is a means of introducing prior knowledge to reduce the number of suboptimal actions [9] and guide the learning process [14]. A concern is that when reward shaping is used incorrectly, it can have a detrimental effect and change the optimal policy or the definition of the task [9,15].

Potential-Based Reward Shaping has been proven to preserve the optimal policy of a task [9,15]. It defines ϕ, a reward function over states that introduces "artificial" shaped reward feedback [3]. The potential function F is defined as a difference between ϕ of the next state s' and the current state s with γ as a discount factor on $\phi(s')$.

The restriction on the form of the reward shaping signal limits its expressiveness [14]. Potential-Based Advice is a similar framework that introduces actions in the potential function [26]. A novel Bayesian approach that augments the reward distribution with prior beliefs is presented in [14].

It is difficult to manually engineer reward functions for each new environment [9,11]. Implicit reward shaping is an alternate approach that learns from demonstrations of target behaviour. A potential-based reward function is recovered from demonstrations using state similarity in [22] and through inverse reinforcement learning methods in [21]. The shaped reward function is learnt directly from raw pixel data in [11].

An alternative to "shaping" an extrinsic reward is to supplement it with intrinsic rewards [16] such as curiosity. Curiosity-Driven Exploration by Self-Supervised Prediction [17] is a fundamental paper that defined a framework for training curious agents. Curiosity empowers the agent by giving it the capability of exploration, enabling it to reach far away states that contain extrinsic rewards. Much research has built upon the findings of this paper. Large scale analysis of the approach is performed in [7] where agents learned to play various Atari Games using intrinsic rewards alone. A limitation of the approach is that it struggles to learn in stochastic environments [7].

Classic work in [6,13] investigated balancing exploration and exploitation in polynomial time and has inspired much research in the area of intelligent exploration. Count-based exploration methods generate an exploration-bonus from state visitation counts [24]. It has been shown to achieve good results on the notoriously difficult "Montezuma's Revenge" Atari game in [4,5]. Exploration bonuses encourage an agent to explore, even when the environment's reward is sparse [4], by optimising a reward function that is the sum of the extrinsic reward and exploration bonus.

Approximating these counts in large state spaces is a difficult task [24]. Hash functions were used in [24] to extend the method to high-dimensional, continuous state spaces. Random Network Distillation (RND) [8] is a novel technique that consists of a fixed randomly initialised target network and a prediction network. The target network outputs a random function of the environment states which the prediction network learns to predict. An intrinsic reward is defined as the loss of the prediction network. It achieved state of the art performance on "Montezuma's Revenge" [5] in 2018.

Other methods of exploration include maximising empowerment [10], wherein the long-term goal of the agent aims to maximise its control on the environment, using the prediction error in the feature space of an auto-encoder as a measure of interesting states to explore, and using demonstration data to learn an exploration policy [23].

3 Directed Curiosity

We propose a new reward function that is made up of two constituents: a distance-based shaped extrinsic reward and a curiosity-based intrinsic reward.

3.1 Distance-Based Reward Shaping

Shaping rewards is a fragile process since small changes in the reward function result in significant changes to the learned policy [25].

Various functions were engineered and compared. It is essential that the positive and negative rewards are balanced. In an episode, the agent should not receive more positive rewards for moving closer to the target, or more negative rewards for moving further away, so as not to introduce loopholes for the agent to exploit. If the weighting of positive rewards is too high, the agent learns to game the system by delaying reaching the target to gain more positive rewards in an episode. If the weighting of the negative rewards is too high, the agent does not receive sufficient positive reinforcement to find the target. This means that the shaped rewards alter the optimal policy of the original task [15].

The shaped reward should encourage the agent to keep advancing towards the target by favouring consecutive positive moves and punishing consecutive negative ones. It must not dominate the terminal reward such that the agent is no longer incentivised to find the target and its motivations become polluted. To overcome these issues, a shaped reward function based on relative distance between target and agent is used.

Algorithm 1 Distance-based shaped reward function

Input: Agent position P_{agent}, target position P_{target}, maximum distance D_{max}, previous distance D_{prev}, reward coefficient C
Calculate distance $D_{current} \leftarrow \text{distance}(P_{agent}, P_{target})$
Calculate reward signal: $R \leftarrow D_{current}/D_{max}$
if $D_{current} < D_{prev}$ **then**
  return $C \cdot (1 - R)$
else
  return $C \cdot (-R)$
end if

There are various benefits to Algorithm 1. The agent is penalised if it stays still and the shaped reward signal can be controlled using the reward coefficient C. This ensures that the episodic shaped rewards cannot exceed terminal positive reward. There is a balance between positive and negative rewards since they are both relative to the change in distance. The agent receives the highest reward when it moves closest to the target and the highest penalty when it moves furthest away. This means that the shaped reward function is policy invariant i.e. it does not alter the goal of the agent to learn the optimal path to the target.

Since the rewards are shaped exclusively based on distance metrics that do not take into account the specific dynamics of the environment, the same function can be used across different environments, and in general, for navigation tasks. A limitation of this approach is that the target location needs to be known. We have investigated using ray casts to find the location of the target if it is unknown, however, the scope of this research is to teach an agent to navigate past obstacles and find an optimal path, given a starting point and a destination. The definition of the task changes drastically, from a navigation-based one to

a goal-finding or search task, when the location is unknown. This is a possible area for future work.

3.2 Curiosity-Driven Exploration

Pathak et al. [17] formally defined a framework for training curious agents that involves training two separate neural-networks: a forward and an inverse model that form an Intrinsic Curiosity Model (ICM). The inverse model encodes the current and next observation into a feature space ϕ and learns to predict the action $\hat{a_t}$ that was taken between the occurrence of the two encoded observations. The forward model is trained to take the current encoded observation and action and predict the next encoded observation.

$$r_i^t = \frac{\eta}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2 \tag{1}$$

In order to generate a curiosity reward signal, the inverse and forward dynamics models' loss functions are jointly optimised i.e. curiosity is defined as the difference between the predicted feature vector of the next state and the real feature vector of the next state. η is a scaling factor.

As an agents explores, it learns more about its environment and becomes less curious. A major benefit of this approach is that it is robust: by combining the two models, the reward only captures surprising states that have come about directly as a result of the agents actions.

3.3 Intelligent Exploration

We propose hybridising curiosity [17] and distance-based reward shaping. Using reward shaping alone is flawed since the agent cannot navigate past obstacles to find a target. Using curiosity alone may cause the agent to spend too much time exploring, after the target has been found, and get trapped in a suboptimal state. By combining the two approaches the agent is able to explore and learn about the dynamics of the environment, while always keeping in mind its goal of finding an optimal path to the target. The agent learns in a more directed and intuitive manner. Curiosity enables the agent to find the target, while the shaped rewards provide feedback to the agent that enables it to learn a path to the goal.

Directed Curiosity simultaneously maximises two reward signals. The reward function components are somewhat conflicting so it is essential to find a balance between them. The agent needs sufficient time to explore the environment, while also ensuring that it does not converge to a suboptimal policy too quickly. This is similar to the exploration vs exploitation Problem in RL. We balance the reward by manually tuning weights attached to both the constituent reward signals. In future work, we wish to find a means of dynamically weighting the reward signals during training. We also wish to investigate alternative means of combining them.

Algorithm 2 Directed Curiosity-Driven Exploration

Input: Initial policy π_0, extrinsic reward weighting w_e, intrinsic reward weighting w_i, max steps T, decision frequency D
for $i \leftarrow 0$ to T **do**
Run policy π_i for D timesteps
Calculate distance-based shaped reward r_e^t (Algorithm 1)
Calculate intrinsic reward r_i^t (Equation 1)
Compute total rewards $r_t = w_i \cdot r_i^t + w_e \cdot r_e^t$
Take policy step from π_i to π_{i+1}, using PPO [20] with reward function r_t
end for

PPO [19] is a popular policy gradient method that is robust and simpler than alternative approaches. Our algorithm is trained using PPO though an arbitrary policy gradient method can be used.

4 Methodology

4.1 Learning Environment

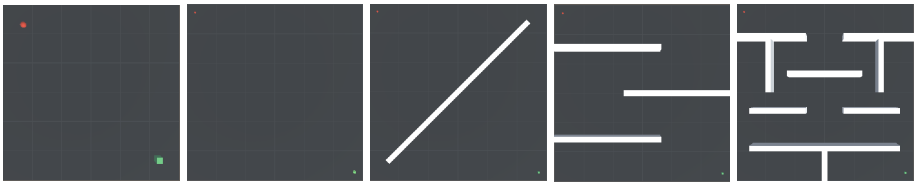
A custom testing environment was created to analyse the performance of our technique, based on the principal of pathfinding. It consists of a ball and a target. The ball is an agent that must learn to navigate to the target, in the shortest possible time (see Fig. 1). The agent is penalised every time it falls off the platform, since there are no walls along the boundaries and it receives a positive reward upon reaching the target. An episode terminates upon falling off the platform, reaching the target, or after a maximum number of steps.

The benefit of this environment is that it defines a simple base task of finding an optimal path to a target. This allows us to perform thorough analysis of the algorithm by continuously increasing the difficulty of the task. In this way, we are able to identify its limitations and strengths. The environment represents a generalisation for navigation tasks wherein an agent only receives positive feedback upon reaching its destination.

The agent is equipped with a set of discrete actions. Action 1 defines forward and backward movement while action 2 defines left and right movement. Simultaneously choosing the actions allows the agent to move diagonally. The agent's observations are vectors representing its current position and the target position. It is not given any information about the dynamics of the environment. The agent must learn an optimal policy that finds the shortest path to the target.

The baseline reward function was carefully tuned: a +100 reward is received for finding the target, -100 penalty for falling off the platform and -0.01 penalty every timestep. The reasoning behind the selected values is to remove bias from the experiments. An agent cannot fall into a local optimum by favouring a single suboptimal policy. This is because a policy that immediately falls off the platform and a policy that learns to remain on the platform for the entire episode, without

finding the goal, will both return roughly the same episodic reward. This function was used as a baseline that was tuned for each new environment.



(a) BasicNav (b) HardNav (c) ObstacleNav (d) MazeNav1 (e) MazeNav2

Fig. 1: Learning Environments. The agent is shown in the top left in red and the target is shown in the bottom right in green.

For the simplest version of the task, the agent and target are placed at fixed locations, on the opposite sides of the platform, without any obstacles between them. We term this an easy exploration task since it is possible for an agent trained with only the sparse reward function to find the target. This is achieved by tuning the floor to agent ratio and agent speed. The shaped reward coefficient C in Algorithm 1 was amplified to 0.1 due to the simplicity of the environment. This is referred to as BasicNav (see Fig. 1a).

The next environment, termed HardNav (see Fig. 1b), is significantly larger. It is a hard exploration environment [17], since an agent trained with a sparse reward function is never able to find the target. Due to the increased number of episode steps, the shaped reward coefficient C in Algorithm 1 was dampened to 0.001.

We also perform testing in environments with walls that block the direct path to the goal and make finding the target more difficult. ObstacleNav (see Fig. 1c) has a single obstacle that is deliberately placed perpendicular to the optimal path to the target, forcing the agent to have to learn to move around the obstacle. The agent is never explicitly given any information about the obstacle. This environment was designed to test the limitations of Directed Curiosity since shaping the reward to minimise the distance to goal is counter-intuitive because it leads the agent directly into the obstacle. The coefficient C in Algorithm 1 was dampened to 0.001.

The remaining set of environments contain multiple walls and obstacles in a maze-like structure. These environments were designed to investigate if the agent can learn to move further away from the target at the current timestep, in order to pass obstacles and reach the target at a later timestep i.e. it needs foresight to succeed. We term the first maze as MazeNav1 (see Fig. 1d).

The last environment is the most difficult version of the task since it has dead-ends and multiple possible paths to the goal. This allows us to investigate the robustness of Directed Curiosity. Even after finding the target, it is difficult to generalise a path from the starting point to the destination since it is easy for the

agent to get stuck in dead-ends or behind obstacles. We term this environment as MazeNav2 (see Fig. 1e). Due to the increased complexities, the terminal reward was increased to +1000 and the shaped reward coefficient C in Algorithm 1 was dampened to 0.000001.

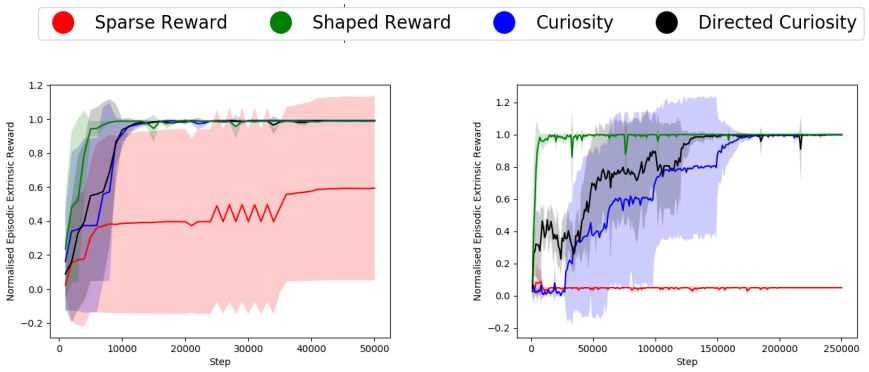
4.2 Hyperparameter Optimisation

It is important to carefully tune the hyperparameters for each environment. The success of the algorithms hinge on these values. Although literature guided this process, the hyperparameters were manually optimised, since the experiments were performed in custom environments. The base hyperparameters were found in BasicNav and then fine-tuned for all other environments, in order to cater for the increased complexities. PPO is a robust learning algorithm that did not require significant tuning [7], once the base hyperparameters were identified and this is a major reason for its selection.

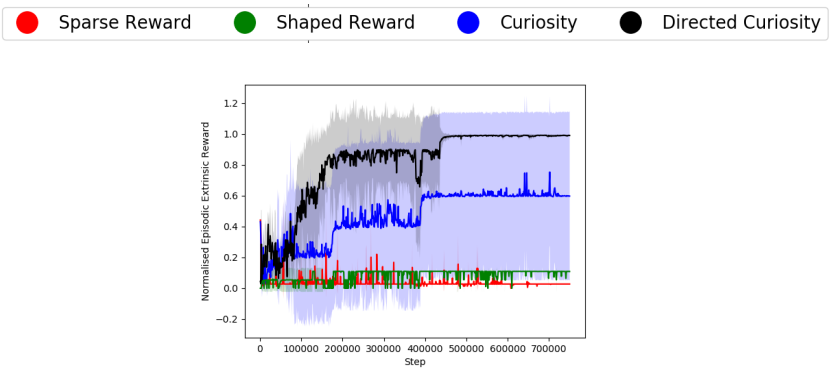
Hyperparameter tuning was essential in ensuring that the algorithms were able to perform meaningful learning. By attempting to tune the parameters to the best possible values, we were able to perform a fair comparison. The notable parameters are a batch size of 32, experience buffer size of 256 and a learning rate of $1.0e-5$. The strength of the entropy regularization β is $5.0e-3$ and the discount factor γ for both the curiosity and extrinsic reward is 0.99. The extrinsic reward weighting is 1.0 and the curiosity weighting is 0.1. The network has 2 hidden layers with 128 units. The baseline parameters were adjusted for each environment: the maximum training steps is 50000 in BasicNav, 250000 in HardNav, 750000 in ObstacleNav and 1000000 in MazeNav1 and MazeNav2.

5 Results

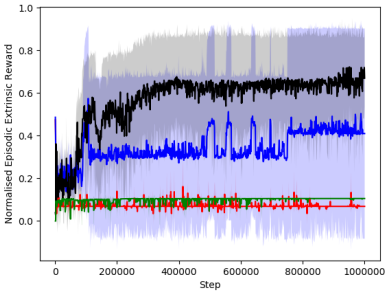
Each algorithm was run five times in every environment. 30 parallel instances of the same environment are used for data collection during training.



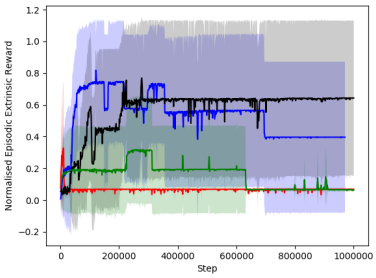
(a) BasicNav (b) HardNav



(c) ObstacleNav



(d) MazeNav1



(e) MazeNav2

Fig. 2: Learning curves for all environments. The average curve from the five runs is shown.

The sparse rewards agent does not perform consistently in BasicNav. The agent is able to find the target and learn an optimal policy on some runs only. This is the reason for the high variance in Fig. 2a. In the hard exploration environments, the agent learns to avoid falling off the platform but is unable to find the target on all runs and therefore receives no positive rewards in training. This highlights the need for an exploration strategy.

The reward shaping agent performs well in BasicNav. This is because the shaped rewards act as a definition of the task since there are no obstacles blocking the direct path to the goal. Continuously moving closer to the target on every timestep leads the agent to the goal in the shortest time. Even in HardNav, the agent is able to learn an optimal policy very quickly, for the same reasons.

The deficiencies of using the shaped reward only are exposed when obstacles are introduced (see Fig. 2c, Fig. 2d). The agent fails to find the target on all runs in ObstacleNav and MazeNav1 and gets stuck behind obstacles. This is because the shaped reward function is a greedy approach and the agent is not equipped with the foresight to learn to move around the obstacles. It cannot learn to move

further away from the target at the current time, in order to reach the target at a later stage.

In MazeNav2 (see Fig. 2e), the agent was able to find the target on two runs. Even though there are multiple obstacles, the optimal path to the goal in MazeNav2 is similar to that in HardNav. The agent "ignores" the obstacles and avoids dead-ends by acting simplistically. By the end of training, however, the agent was unable to converge to an optimal policy on any of the runs.

These results show that distance-based reward shaping provides the agent with some valuable feedback, but without an intuitive exploration strategy, the agent lacks the foresight needed to moves past obstacles that block it's path to the target.

The curiosity agent was able to consistently learn an optimal policy in the environments without obstacles. However, Fig. 2a shows that the curiosity agent takes longer to converge to an optimal policy in BasicNav. This highlights that curiosity is not necessary in environments that are not hard exploration. In HardNav (see Fig. 2b), the curiosity agent is still able to find an optimal policy on all runs, but it is significantly slower than the shaped reward function.

The necessity of the curiosity signal is highlighted when obstacles are introduced. Not only does it enable the agent to find the distant target, it also implicitly learns about the dynamics of the environment, allowing the agent to learn how to move past multiple obstacles.

In ObstacleNav (see Fig. 2c), the agent is still able to learn an optimal policy on most runs. The performance of the agent is not as successful in MazeNav1 (see Fig. 2d) and MazeNav2 (see Fig. 2e).The agent successfully learns an optimal policy on two of the runs. In these environments, it is difficult to converge to an optimal policy, once the target has been found. One reason for this is that the agent keeps exploring after initially finding the target and gets stuck behind obstacles and in dead-ends, eventually converging to an unsuccessful policy, without being able to reach the target again. The curiosity signal is insufficient to direct the agent back to the target and learn a path to the destination. This is the reason for the increase of the average reward in the early stages of training and the subsequent drop thereafter in Fig. 2e.

These results indicate that curiosity equips an agent with the ability to find a target in hard exploration environments with obstacles, but the agent requires additional feedback to consistently learn a path from the start point to the destination.

The Directed Curiosity agent is shown to be the most robust technique. Fig. 2a and Fig. 2b show that Directed Curiosity always finds an optimal policy in BasicNav and HardNav. It converges to a solution faster than the curiosity agent in HardNav, due to the additional shaped reward feedback.

The hard exploration environments highlight the benefits of the technique. It is the only technique that converges to an optimal solution on all runs in ObstacleNav (see Fig. 2c). Curiosity enables the agent to find the target and move past the obstacle, while the shaped rewards provide additional feedback

that allows the agent to learn an optimal path to the target, once it has been found.

MazeNav1 (see Fig. 2d) and MazeNav2 (see Fig. 2e) exhibit promising results since the Directed Curiosity agent learns an optimal policy on more runs than any other technique i.e. on three of the five runs. Training is more stable than the Curiosity agent. The agent always finds the target during training, however, it is unable to consistently find an optimal policy on all runs. A major reason is due to the limitations we have highlighted with the shaped reward function. In future work, we wish to investigate a more intuitive reward function that has foresight. Another reason is due to the complexities we have introduced in these environments. The reward feedback is not sufficient to guide the agent out of dead-ends back to the target. However, these results indicate that the two components of Directed Curiosity, when balanced correctly, allow the agent to learn in a more directed and intuitive manner.

For all algorithms, the variance of the results increase with the difficulty of the task since the agents do not always converge to an optimal policy i.e. when the agent does not learn a path to the target, it does not receive the terminal reward and hence its episodic rewards are significantly lower. PPO learns a stochastic policy, hence, even on the successful runs, the algorithms converge at different times. Due to the inherent randomness in the algorithm, the agent explores differently on every run and thus visits states in a different order.

6 Conclusions and Future Work

A new approach to learning in hard exploration, sparse reward environments, that maximises a reward signal made up of a hybrid of Curiosity-Driven Exploration [17] and distance-based reward-shaping, is presented. This algorithm is compared to baseline algorithms in a custom pathfinding environment and it is shown that the technique enables agents to learn in a more directed and intuitive manner.

The Directed Curiosity agent was the most robust technique. It was able to consistently learn an optimal policy in hard exploration environments with a single obstacle, and learned optimal polices more often then the other techniques, in hard exploration environments with multiple obstacles and dead-ends.

In future work, we wish to investigate alternative reward functions that are more flexible than the current greedy approach. We would like to perform further testing in existing benchmarked environments and in domains other than navigation. This requires further research into "intelligent exploration", through hybridising different shaped reward signals and exploration strategies. Another interesting direction is to create environments with multiple targets and agents.

References

1. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O.P., Zaremba, W.: Hindsight experience replay. In: Advances in Neural Information Processing Systems. pp. 5048–5058 (2017)

2. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep Reinforcement Learning: A Brief Survey. IEEE Signal Processing Magazine **34**(6), 26–38 (Nov 2017). https://doi.org/10.1109/MSP.2017.2743240, http://ieeexplore.ieee.org/document/8103164/
3. Badnava, B., Mozayani, N.: A new Potential-Based Reward Shaping for Reinforcement Learning Agent. arXiv:1902.06239 [cs] (May 2019), http://arxiv.org/abs/1902.06239, arXiv: 1902.06239
4. Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., Munos, R.: Unifying count-based exploration and intrinsic motivation. In: Advances in Neural Information Processing Systems. pp. 1471–1479 (2016)
5. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research **47**, 253–279 (2013)
6. Brafman, R.I., Tennenholtz, M.: R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. Journal of Machine Learning Research **3**(Oct), 213–231 (2002), http://www.jmlr.org/papers/v3/brafman02a.html
7. Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A.A.: Large-Scale Study of Curiosity-Driven Learning. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=rJNwDjAqYX
8. Burda, Y., Edwards, H., Storkey, A., Klimov, O.: Exploration by random network distillation. arXiv preprint arXiv:1810.12894 (2018)
9. Devlin, S.M., Kudenko, D.: Dynamic Potential-Based Reward Shaping (Jun 2012), http://eprints.whiterose.ac.uk/75121/
10. Gregor, K., Rezende, D.J., Wierstra, D.: Variational intrinsic control. arXiv preprint arXiv:1611.07507 (2016)
11. Hussein, A., Elyan, E., Gaber, M.M., Jayne, C.: Deep reward shaping from demonstrations. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 510–517. IEEE (2017)
12. Kang, B., Jie, Z., Feng, J.: Policy Optimization with Demonstrations p. 10 (2018)
13. Kearns, M., Singh, S.: Near-optimal reinforcement learning in polynomial time. Machine learning **49**(2-3), 209–232 (2002)
14. Marom, O., Rosman, B.: Belief reward shaping in reinforcement learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
15. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: ICML. vol. 99, pp. 278–287 (1999)
16. Oudeyer, P.Y., Kaplan, F.: What is intrinsic motivation? A typology of computational approaches. Frontiers in neurorobotics **1**, 6 (2009)
17. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-Driven Exploration by Self-Supervised Prediction. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 488–489. IEEE, Honolulu, HI, USA (Jul 2017). https://doi.org/10.1109/CVPRW.2017.70, http://ieeexplore.ieee.org/document/8014804/
18. Ravishankar, N.R., Vijayakumar, M.V.: Reinforcement Learning Algorithms: Survey and Classification. Indian Journal of Science and Technology **10**(1) (Jan 2017). https://doi.org/10.17485/ijst/2017/v10i1/109385, http://www.indjst.org/index.php/indjst/article/view/109385
19. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs] (Jul 2017), http://arxiv.org/abs/1707.06347, arXiv: 1707.06347

21. Suay, H.B., Brys, T.: Learning from Demonstration for Shaping through Inverse Reinforcement Learning p. 9 (2016)
22. Suay, H.B., Brys, T., Taylor, M.E., Chernova, S.: Reward Shaping by Demonstration. In: Proceedings of the Multi-Disciplinary Conference on Reinforcement Learning and Decision Making (RLDM) (2015)
23. Subramanian, K., Isbell Jr, C.L., Thomaz, A.L.: Exploration from demonstration for interactive reinforcement learning. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems. pp. 447–456. International Foundation for Autonomous Agents and Multiagent Systems (2016)
24. Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., Abbeel, P.: #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 2753–2762. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/6868-exploration-a-study-of-count-based-exploration-for-deep-reinforcement-learning.pdf
25. Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., Riedmiller, M.: Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. arXiv:1707.08817 [cs] (Jul 2017), http://arxiv.org/abs/1707.08817, arXiv: 1707.08817
26. Wiewiora, E., Cottrell, G.W., Elkan, C.: Principled methods for advising reinforcement learning agents. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03). pp. 792–799 (2003)