Advancing AI Capabilities for Dynamic Physical Environments: Transitioning from Closed-World Problem Solving to Open-World Challenges

A thesis submitted for the degree
Doctor of Philosophy in Computer Science

By:
Cheng Xue

Supervisor:
Prof. Jochen Renz

Australian National University

School of Computing
College of Engineering, Computing and Cybernetics (CECC)
The Australian National University

September 2024

Disclaimer:

Chapter 3 is curated from "Phy-Q as a measure for physical reasoning intelligence" [184]. As one of the first authors, I introduced the concept of assessing agents' physical reasoning capabilities, akin to those acquired by humans during infancy. Additionally, I proposed the idea and methods of evaluation of two distinct generalization capacities: narrow and broad generalization. Spearheading these efforts, I formulated, tested, and computed the Phy-Q measure.

Furthermore, I conceptualized and implemented the server-game-agent testing framework, overseeing the execution of experiments for all baseline agents. In addition to designing the experiments (collaboratively with Chathura and Vimu) related to the baseline agents, I authored the corresponding sections and meticulously proofread other parts, providing comprehensive comments for improvement.

Chapter 5 is curated from "NovPhy: A Testbed for Physical Reasoning in Open-world Environments" [60]. As one of the first authors, I proposed a novel framework for assessing novelty adaptation agents in diverse physical scenarios, aiming to accurately isolate and gauge the genuine extent of their adaptation capabilities. Collaboratively with Chathura and Vimu, I further honed the evaluation design. Taking a hands-on approach, I also implemented the server-game-agent testing framework and introduced and developed an innovative adaptation agent, the Naive Adapt agent, for comprehensive experimentation. My involvement extended to conducting experiments for the baseline agents and crafting the corresponding experiment sections. Additionally, I proofread various sections, providing extensive comments for refinement.

Acknowledgements

I am profoundly grateful to my parents, whose unwavering support and refusal to ever give up on me have been the bedrock of my journey. Your sacrifices, belief in my dreams, and endless encouragement have propelled me forward every step of the way.

To my dear grandma, although you are no longer with us, your generous spirit and love continue to inspire me. Your selfless act of providing the initial financial support enabled me to embark on my studies. Your belief in my potential laid the foundation for everything I have achieved.

To my beloved wife, Yuting, words cannot express the depth of my gratitude for your unwavering support and care, especially during the most challenging times. Throughout my PhD journey, and particularly during the tumultuous period of separation caused by the COVID-19 pandemic, you stood by me with unwavering love and strength.

And to my supervisor, Jochen Renz, your guidance, mentorship, and belief in my abilities have been instrumental in shaping my academic and personal growth. Your support provided me with the confidence to push my boundaries and strive for excellence.

Additionally, I extend my appreciation to DARPA for their partial support through the SAILON program, which has significantly contributed to the advancement of this work.

With heartfelt appreciation,
Cheng

Abstract

Contemporary artificial intelligence (AI) systems have excelled in well-defined tasks within controlled environments, often surpassing human performance, especially in complex games like Go, StarCraft, and Dota. However, transitioning to dynamic and unpredictable real-world scenarios presents a formidable challenge. AI systems must respond to unforeseen events, adapt to environmental changes, and navigate unfamiliar terrains. To facilitate their deployment across diverse applications, AI must move beyond closed-world problem-solving and confront open-world challenges.

Addressing this pivotal issue requires a multifaceted approach. Firstly, the development of robust evaluation methodologies is important to accurately assess an agent's performance in a pre-novelty environment. This baseline assessment distinguishes whether an agent's success or failure in a novel task is attributable to its intrinsic capabilities or merely a matter of chance. In Chapter 3, we discuss our solution to this challenge: Phy-Q, a metric for measuring physical reasoning intelligence. Secondly, it is crucial to create environments that enable the introduction of novel elements, thus facilitating comprehensive evaluations of agents facing unforeseen challenges. To tackle this issue, in Chapter 4, we introduce ScienceBird Novelty, a framework that empowers users to inject a wider range of novelties and evaluate agents specializing in novelty adaptation. Furthermore, designing a benchmark tailored specifically to evaluate an agent's performance in handling novelty is a non-trivial endeavour. Such a benchmark is essential for isolating the reasons behind an agent's success or failure in various novel situations. To this end, in Chapter 5, we present NovPhy, a benchmark that assesses the performance characteristics of novelty adaptation agents. Drawing upon a comprehensive toolkit of methodologies and approaches, we introduce our novel adaptation framework, NAPPING (Novelty Adaptation Principles Learning) in Chapter 6. Through a series of empirical demonstrations, we illustrate how NAPPING empowers deep reinforcement learning agents to rapidly adapt to a wide spectrum of novel situations, devoid of any prior knowledge regarding the specific novelty at hand. This breakthrough in AI adaptation promises to significantly enhance the flexibility and utility of intelligent systems across diverse real-world applications, paving the way for more effective collaboration between AI and human counterparts in the face of ever-changing conditions.

Table of Contents

Chapter 1

Introduction

Imagine an AI-powered drone created and trained for search and rescue missions. This drone undergoes extensive training in a computer simulation that replicates various disaster scenarios, such as earthquakes and floods. Throughout its training, the drone becomes highly skilled at finding survivors, avoiding obstacles, and delivering supplies to those in need.

Nevertheless, when this drone is dispatched to a real disaster scenario, it faces a range of challenges that hinder its operation. These challenges could arise from unanticipated issues that were not incorporated into its training, including:

1. Unpredictable Terrain and Weather: Real-life disasters often come with complex terrain and challenging weather conditions. The drone may have to deal with strong winds, heavy smoke, and extreme weather that it did not prepare for. These conditions disrupt the drone's ability to navigate and sense its surroundings, making it difficult for the drone to avoid obstacles and locate survivors accurately.

2. Communication and Connectivity Issues: In the simulated training, communication with the drone is straightforward. However, in real search and rescue situations, the drone may encounter difficulties maintaining a stable connection due to interference or poor signal quality. This can lead to delays or errors in the commands it receives.

3. Difficulty Recognizing Novel Situations: The simulation does not account for the many ways people may appear, behave, or signal for help during a disaster. Consequently, the drone may struggle to identify survivors who do not conform to its expectations.

4. Unexpected Challenges: Real disasters often present unforeseen obstacles, such as unstable structures, debris, or even frightened animals. The drone finds it challenging to cope with these unexpected hurdles, and this may result in damage to the drone or failure to locate survivors.

The challenges underscore a fundamental truth: it is exceedingly difficult and cost-prohibitive, if not outright impossible, for developers to predict and account for every conceivable real physical world scenario within a simulator or for the agent. Thus, it is important to develop AI agents capable of dynamic adaptation to unforeseen challenges encountered in the inherently unpredictable real world. Such agents are commonly known as open-world learning agents, and their success hinges on their ability to adapt and perform effectively in unpredictable environments.

However, before developing such AI agents, the need to accurately evaluate their performance becomes evident. Assessing an agent's capacity to adapt to novel elements is a complex task, particularly when distinguishing between the agent's adaptability limitations and its fundamental inability to execute its core task. Equally challenging is determining whether an agent's success in handling novel scenarios arises from a genuine understanding of adaptation or mere luck. Consequently, we confront the critical need for innovative evaluation methodologies and benchmarking frameworks that comprehensively assess an agent's competence in novel scenarios, illustrating its true capabilities. The availability and diversity of novelties, as well as the establishment of standardised benchmarks, are pivotal concerns in this context.

This thesis undertakes a multifaceted journey. It begins by addressing the challenges of evaluation, focusing on physical domains and the identification of the true reasons behind an agent's success or failure in a given task. Subsequently, it explores the creation of a versatile test bed for open-world learning, a foundation that provides the means to introduce and evaluate a wide range of novelties systematically. With the establishment of robust benchmarking methodologies, the thesis then focuses on the development of AI agents that can adapt rapidly and effectively to novel situations.

In the pages that follow, we outline our contributions to this evolving field, explaining the methodologies we employ, and showcasing their impact through empirical evidence. We hope to shed light on the path forward for AI systems that can thrive in open and unpredictable environments.

1.1 Contributions and Thesis Outline

In this thesis, we embark on a journey that encompasses three distinct phases, each contributing to the advancement of AI systems in the context of open-world learning.

1.1.1 Addressing Evaluation Challenges in Physical Domains

We begin by tackling the fundamental challenge of evaluating AI agents' performance, with a particular focus on physical reasoning. Our quest to understand why AI agents succeed or fail in various tasks begins by isolating the core competencies of physical reasoning. Inspired by the innate physical knowledge acquired during infancy and the essential skills required by robots operating in the real world, we craft a diverse array of tasks with low dexterity requirements. These tasks are generated automatically in

the popular video game, Angry Birds. Our approach involves designing a set of task templates, comprising 15 distinct physical scenarios, each structured around a common strategic physical rule. We also introduce the *Phy-Q* score, a quantitative metric tailored to measure an agent's physical reasoning intelligence, discussed in detail in Chapter 3.

1.1.2 Creating a Versatile Benchmark for Open-World Learning

The journey continues with the exploration of a test bed designed to facilitate open-world learning. We believe that an ideal test bed for open-world learning should possess five key characteristics. Firstly, it should be uncomplicated enough to enable agents to concentrate on confronting novel elements within the domain, without being loaded by unrelated AI research challenges, such as object tracking in crowded scenes. Secondly, the test bed should exhibit versatility, allowing for the introduction of a wide range of novelties. Thirdly, it should be under precise control, granting users the authority to systematically determine the timing, location, and nature of novelties in the environment for the evaluation of open-world learning systems. Additionally, it should facilitate the detection and characterization of novelties, and the recording of these observations alongside performance data. Lastly, an open-world learning test bed should incorporate a set of baseline agents that either possess expertise or access to a sufficiently extensive training dataset to ensure satisfactory agent performance in a standard environment. We delve into the Science Birds Novelty framework in Chapter 4, which encompasses all the aforementioned characteristics and serves as a foundation for evaluating novelties in open-world learning scenarios.

Expanding upon this *Science Birds Novelty* and the *Phy-Q*, we introduce an open-world learning benchmark *NovPhy* in Chapter 5. Here, we evaluate open-world learning agents that focuses on physical domains, across five common physical scenarios: applying a single force, applying multiple forces, and dealing with rolling, falling, and sliding objects. Within this benchmark, we have created eight distinct novelties, each representing a different facet of the novelty space. We've designed 40 novel task templates by incorporating these novelties into the five physical scenarios separately. Each task template serves as the basis for generating related tasks by adjusting parameters like object locations. Additionally, we've created 40 corresponding normal task templates without novelties to support our evaluation protocol. Our task variation generator allows us to generate an unlimited number of tasks from these templates. To provide a comprehensive evaluation of novelty detection and adaptation in AI systems operating in open-world physical environments, we've established a robust evaluation setup. This setup enables us to assess agents in two contexts: first, where the same novelty is applied separately to tasks across multiple physical scenarios, and second, where multiple novelties are applied individually to tasks within the same physical scenario. The former evaluates an agent's ability to handle the same novelty across different physical reasoning tasks, while the latter assesses an agent's performance in the same physical reasoning task under various novel circumstances.

1.1.3 Developing Adaptive AI Agents with NAPPING

Expanding upon the foundation established in the previous phases, we introduce a novel learning algorithm called NAPPING (Novelty Adaptation Principles Learning) in Chapter 6. NAPPING works on top of deep reinforcement learning agents, which are trained exclusively in pre-novelty environments, to enable rapid adaptation to novel situations when exposed to post-novelty environments. NAPPING identifies regions within the state space where the previous policy fails to perform and systematically adjusts actions within these regions. Notably, NAPPING leverages embedded representations of states, enhancing the speed and efficiency of adaptation. We proceed to demonstrate the effectiveness of NAPPING in four distinct environments: the CartPole and MountainCar control domains, the CrossRoad path-finding domain, and the AngryBirds physical reasoning domain. Our evaluation methodology includes a comparison with the online learning and fine-tuning versions of baseline agents in the first three domains and with two state-of-the-art novelty adaptation agents in the AngryBirds domain. In all cases, NAPPING exhibits a remarkable ability to adapt rapidly and effectively to novel situations, accommodating a wide range of novelties in just a few episodes.

The journey outlined in this thesis aims to contribute to the field of AI systems capable of thriving in open and unpredictable environments, addressing the critical challenges of evaluation, benchmarking, and adaptation along the way.

Chapter 2

Background and Related Work

2.1 Evaluation of AI Agents' Physical Reasoning Abilities

Physical reasoning is a set of cognitive processes that humans and intelligent systems use to understand, predict, and manipulate the physical world. It involves the extraction and application of knowledge about fundamental physical principles, such as Newtonian mechanics, to reason about the behavior of objects and systems in various environments. What sets physical reasoning apart from other forms of cognition is its foundation in the laws of physics and its focus on modelling dynamics and causality.

The ability to reason about objects' properties and behaviours in physical environments lies at the core of human cognitive development [186, 47]. A few days after birth, infants understand object solidity [165] and within the first year of birth, they understand notions such as object permanence [14], spatiotemporal continuity [95], stability [18], support [16], causality [145], and shape constancy [48].

However, mastering physical reasoning is challenging for AI agents. Physical environments are characterized by variability and uncertainty, arising from reasons such as noise, disturbances, and incomplete information [140]. Uncertainty can exist in various forms, such as sensor noise in robotic perception or stochasticity in the behaviour of dynamic systems. Addressing uncertainty requires robust reasoning mechanisms capable of incorporating probabilistic inference and adapting to uncertain conditions [24, 4]. Effective physical reasoning requires abstracting complex phenomena into symbolic representations that capture essential characteristics and relationships. This process involves identifying relevant variables, defining meaningful abstractions, and reasoning symbolically about physical concepts. While traditional symbolic approaches encounter difficulties in scaling to complex physical domains, recent advancements in neuro-symbolic methods [186, 38] offer promising avenues for progress.

As the reliance on autonomous AI systems grows in various day-to-day operations, phys-

ical reasoning has become a crucial aspect of AI research. To enhance AI systems' physical reasoning capabilities for secure real-world performance, numerous physical reasoning benchmarks and testbeds have been developed. In this section, we conduct a comparative analysis of ten related physical reasoning benchmarks and two physics-based AI game competitions, demonstrating how the Phy-Q (Chapter 3) testbed advances upon existing work. Our comparison considers six key criteria:

1. Measuring Broad Generalization: Assessing an agent's ability to generalize to tasks that require the same physical rule within individual physical scenarios.

2. Categorization of Tasks: Evaluating agents for individual scenarios to identify scenarios in which they excel.

3. Procedural Task Generation: Algorithmically generating tasks and task variations in the test environment to enable the generation of ample data.

4. Destructible Objects: Introducing objects in the environment that can be destroyed upon the application of forces, making the environment more realistic.

5. Observing Action Outcomes: Allowing agents to physically interact with the environment and observe the outcomes of their actions.

6. Inclusion of Human Player Data: Incorporating results from human players in the evaluation.

We consider PHYRE [19], Virtual Tools game [4], and OGRE [3], which are game-based benchmarks, IntPhys [142], CLEVERER [186], CATER [63], and Physion [25] which are video based benchmarks, COPHY [21] which is an image-based benchmark, CausalWorld [1] and RLBench [74] which are robotic benchmarks. The AI game competitions we consider are Computational Pool [8], and Geometry Friends [133]. We also included the AIBIRDS [2] competition for the comparison to show what properties in Phy-Q facilitate the systematic evaluation of AIBIRDS competition agents. Table 2.1 summarises the comparison.

The most closely related benchmark to Phy-Q is PHYRE [19], which measures generalization in two levels: within-template and cross-template. However, cross-template evaluation in PHYRE doesn't ensure that the required physical rules are covered in training tasks. This leads to uncertainties in understanding agents' performance: inferior performance may not be an indicator of inferior physical reasoning but of a difficult training and testing split. *Phy-Q*'s broad generalization evaluation guarantees coverage of physical rules in training tasks, thus offering a more systematic assessment of agents' physical reasoning capabilities.

In the task design of PHYRE, tasks typically require a trial-and-error approach for solutions. Even when the underlying physical rule is known, multiple attempts are often necessary to complete the tasks. As a result, PHYRE primarily encourages the development of agents with refined physical dexterity.

Table 2.1: Comparison of Phy-Q with related physics benchmarks and competitions

Test Environment	generalization to individual physical scenario/s	categorization of tasks to physical scenarios	procedurally generated tasks/variations	destructible objects	observe outcome of a desired physical action	human player data
PHYRE [19]	✗	✗	✓	✗	✓	✗
Virtual Tools [4]	✗	✓	✗	✗	✓	✓
OGRE [3]	✗	✗	✓	✗	✓	✗
IntPhys 2019 [142]	✓	✓	✓	✗	✗	✓
CLEVERER [186]	✓	✗	✓	✗	✗	✗
CATER [63]	✓	✓	✓	✗	✗	✗
Physion [25]	✓	✓	✓	✗	✗	✓
COPHY [21]	✓	✓	✓	✗	✗	✓
CausalWorld [1]	✓	✓	✓	✗	✓	✗
RLBench [74]	✗	✗	✓	✗	✓	✗
Computational Pool [8]	✗	✗	✗	✗	✓	✗
Geometry Friends [133]	✗	✗	✓	✗	✓	✗
AIBIRDS [2]	✗	✗	✓	✓	✓	✓
Phy-Q (ours)	✓	✓	✓	✓	✓	✓

In contrast, our approach centres on strategy-based physical reasoning tasks that can be successfully solved in a single attempt once the appropriate physical rule is comprehended. Our emphasis lies in promoting agents capable of understanding these physical principles, allowing them to approach tasks strategically rather than relying solely on precise actions in the physical environment. In essence, we foster the development of agents with strategic physical reasoning capabilities.

Moreover, the limited diversity in terms of object shapes, motion characteristics, and material properties in scene dynamics can pose limitations on the comprehensiveness of the evaluation process. Excelling in such tests may not necessarily reflect superior physical reasoning abilities applicable to more varied and realistic scenarios (Physion [25]). Therefore, in contrast to PHYRE, our Phy-Q testbed offers several enhancements:

1. We introduce three additional object shapes, namely rectangles, squares, and triangles, thereby broadening the spectrum of physical dynamics.

2. Destructible objects are incorporated into our environment, providing a more realistic setting that necessitates agents to consider the magnitude of force applied to objects, mirroring real-world scenarios.

3. Our testbed includes objects with three distinct materials, each possessing different characteristics such as density, bounciness, and friction. This variation enables agents to engage in physical reasoning within a more realistic context, expanding the scope of evaluation.

Physion [25], a recent benchmark for visual and physical prediction, assesses algorithms' capabilities in predicting physical events using videos of eight distinct physical scenarios. In contrast, the Phy-Q testbed offers a more extensive range of 15 physical scenarios, allowing for a more diverse evaluation of agents' performance across a wider spectrum of physical challenges. Notably, the Phy-Q testbed places a unique emphasis on the practical application of acquired physical knowledge, as agents are required to interact

with the environment and select actions to accomplish various physical tasks.

Despite the simplified and controlled nature of the physics in the Angry Birds environment, it is worth noting that, thus far, no AI system has come close to achieving human-level performance in this context. To foster the development of AI agents capable of reasoning about physics in a manner akin to human cognition, the AIBIRDS competition has been held annually since 2012, with most of its iterations taking place at the International Joint Conference on Artificial Intelligence (AIBIRDS [2]). Over the years, a diverse array of AI approaches has been proposed to tackle the AIBIRDS competition, encompassing modern techniques rooted in deep reinforcement learning, as well as more traditional heuristic methods, such as qualitative physical reasoning strategies.

However, despite these varied approaches, none have succeeded in reaching the significant milestone of achieving human-level performance. One of the primary challenges lies in the fact that the current methods employed in the competition do not facilitate an agent developer in identifying the specific physical scenarios where the agent falls short. This issue arises due to the inherent complexity of the tasks featured in the competition, which often encompass multiple intricate physical scenarios within a single task. In our work, we address this critical limitation by demonstrating how the Phy-Q testbed can serve as a valuable tool for guiding competition agents through a systematic and comprehensive evaluation of their performance.

2.2 Open-World Learning

The Formation of Open-World Learning The genesis of open-world learning can be traced back to its initial proposal in [148]. Over the years, this field has flourished, yielding a plethora of theoretical perspectives and practical methodologies. Notably, researchers have delved into formulating precise problem definitions [90, 91, 72], developing taxonomies and definitions for novelties [90, 31, 116], and crafting a spectrum of agent architectures [90, 91, 116, 101, 120, 34]. These advancements collectively lay the groundwork for AI agents to effectively navigate the intricate landscape of open worlds. A salient feature of open-world learning is the clear distinction from related paradigms, such as transfer learning, meta-learning, open-set recognition, incremental learning, and zero-shot learning. This demarcation is thoughtfully explored in [90, 72, 64, 120, 20], further highlighting the unique challenges and opportunities that open-world learning presents.

2.2.1 Novelty Theories

AI systems have indeed demonstrated remarkable capabilities in surpassing human performance across a wide spectrum of closed-world domains, as substantiated by prior research [152, 151, 169]. Nevertheless, the transition to open-world scenarios presents a formidable and distinct challenge. In open-world settings, AI agents often grapple with unexpected hurdles when confronted with novel situations. While developers may

attempt to anticipate and account for some of these novel scenarios, the inherently unpredictable nature of open worlds makes it exceedingly impractical to comprehensively encompass all potential novel situations within an agent's model. To tackle this challenge, DARPA initiated the Science of Artificial Intelligence and Learning for Open-world Novelty (SAIL-ON) program. This program is dedicated to the investigation and advancement of the fundamental scientific principles, general engineering techniques, and algorithms that are indispensable for enabling AI systems to adapt effectively to unforeseen circumstances [148].

Researchers have explored various avenues to formalize the concept of novelty. Novel situations have been referred to as anomalies or out-of-distribution data by some researchers [31, 54]. Within the SAIL-ON program, novelty is defined as situations that deviate from implicit or explicit assumptions within an agent's model of the external world, encompassing elements like other agents, the environment, and their interactions [50, 80]. Alternatively, Langley characterizes novelty as transformations of elements in the environment [90]. These transformations can manifest as spatio-temporal changes, structural alterations, process modifications, and constraint adjustments. Furthermore, Molineaux and Dannenhauer have provided a formal definition of various environmental transformations [116]. In contrast, Boult and colleagues have introduced a comprehensive framework for understanding novelty in the context of AI. Their framework delves into the world space, observation space, and agent state to formally define a diverse array of novelties.

Within the SAIL-ON program, a dedicated working group, known as the novelty working group (refer to [50] for a list of participants), has developed a novelty hierarchy. This hierarchy serves as a valuable tool for categorizing novelties based on their specific properties. This categorization framework not only facilitates a robust evaluation of novelty but also aids in the design of novelties that cover a broad spectrum of possibilities. Furthermore, it assists in identifying different categories of novelties that an agent model might struggle to handle. In this paper, we leverage the novelty hierarchy crafted by the SAIL-ON novelty working group. This hierarchy consists of various levels, including objects, agents, actions, relations, interactions, environments, goals, and events [50]. Please refer to Table 2.2 for an overview of the novelty hierarchy levels.

2.2.2 Novelty-centric Domains and Benchmarks

The presence of domains that revolve around novelty is paramount for fostering the advancement of open-world learning (OWL). These domains provide a space where agents can evaluate and compare their performance in the face of novel challenges. Within this domain, various testbeds, frameworks, and benchmarks have been created, with an explicit focus on making novelties central to their design. In this section, we explore the body of research concerning domains tailored for OWL, providing the necessary context for understanding NovPhy's place within this evolving landscape.

Table 2.2: SAIL-ON Open-world novelty hierarchy [50]

Novelty Level	Description
1. Objects	New classes, attributes, or representations of non-volitional entities.
2. Agents	New classes, attributes, or representations of volitional entities.
3. Actions	New classes, attributes, or representations of external agent behavior.
4. Interactions	New classes, attributes, or representations of dynamic properties of behaviors impacting multiple entities.
5. Relations	New classes, attributes, or representations of static properties of the relationships between multiple entities.
6. Environments	New classes, attributes, or representations of elements independent of specific entities.
7. Goals	New classes, attributes, or representations of external agent objectives. pushing objects down.
8. Events	New classes, attributes, or representations of series of state changes.

GNOME, a specialized simulator tool, concentrates on generating and evaluating AI systems in multi-agent environments, including strategic board games such as Monopoly (as detailed in [80]). GNOME introduces novelties that span the initial three tiers of the novelty hierarchy, as mentioned earlier. These novelties encompass modifications like the addition of extra dice to the game or the rearrangement of slots on the board.

NovGrid [20] and NovelGridworlds [65] are two frameworks specifically designed for grid environments. NovGrid is a toolkit designed for generating novelties within the MiniGrid environment [41]. It enhances MiniGrid by expanding the capabilities of existing objects, empowering agents to detect and adapt to novel situations. NovelGridworlds, on the other hand, implements a grid world inspired by the popular game Minecraft [112] for the study of novelties. In NovGrid, novelties are introduced based on the first two levels of the novelty hierarchy, while in NovelGridworlds, novelties pertain to the first three levels.

NovelCraft [54] stands out as a benchmark dataset for the detection and adaptation of novelties, using a modified version of Minecraft as its backdrop. NovelCraft creates a 3D environment in which an agent is tasked with selecting a sequence of actions to craft a pogo stick from available resources [156]. This benchmark primarily focuses on assessing agent performance with respect to the first level of the novelty hierarchy, namely objects [54].

However, none of the aforementioned novelty domains is rooted in physics, which is a significant consideration when developing agents intended for real-world applications. Cartpole with Novelty [35] and Science Birds Novelty [185] are two domains that specifically centre around physics and have been designed to evaluate novelty detection and adaptation. In Cartpole, the agent's objective is to maintain balance by controlling a pole's position, whereas in Science Birds Novelty, based on the physics game Angry Birds [143], the agent is tasked with eliminating pigs by launching birds from a slingshot. Our work [60] draws from the Science Birds framework [185] and introduces a range of realistic physical reasoning tasks, focusing on the evaluation of agents' physical reasoning capabilities in the context of novelty.

Despite the advancements achieved by the benchmarks mentioned earlier in fostering AI systems with sophisticated physical reasoning capabilities, none of them primarily addresses physical reasoning in the context of novel situations—scenarios that real-world agents would frequently encounter. In Chapter 5, we bring together physical reasoning scenarios with novel challenges to create an environment that closely mirrors real-world situations. To introduce these novel situations, we draw inspiration from the first five physical reasoning scenarios from the Phy-Q testbed: single force, multiple forces, rolling, falling, and sliding.

Novelty adaptation, a fundamental aspect of open-world learning, has emerged as a focal point of research in recent years. This section examines the evolving landscape of open-world learning, encompassing theoretical foundations, definitions, and diverse agent frameworks that empower AI agents to navigate unpredictable environments filled

with unexpected novelties. The multidimensional nature of open-world learning, its distinction from related paradigms, and its critical role in contemporary AI research are underscored.

2.2.3 Adapting to Novelty

Reinforcement Learning and Deep Q Network

Similar to [114], in this thesis, we examine tasks that involve an agent interacting with an environment, denoted as E, by taking a sequence of actions, observations, and rewards. At each time step, the agent picks an action a_t from a set of allowable actions, A, which is represented by numbers from 1 to K. The chosen action changes the internal state of the environment, and it's essential to note that the environment may show random behaviour. The representation of the environmental state may or may not be directly observed by the agent, with the possibility of either raw pixel values representing the agent's visual perception or a more abstract state representation. Moreover, the agent receives an immediate reward r_t at each time step, which provides feedback on the outcomes of the chosen action. It's important to note that, in general, the reward may depend on the entire history of actions and observations that occurred before, and it might be the case that feedback about specific action consequences is only received after a significant number of time steps have passed.

This framework provides a way to model and comprehend the interaction between an agent and its environment. The environment can be represented as a Markov decision process (MDP), which is defined by the tuple (S, A, P, R).

S : State space
A : Action space
P : Transition probability function
R : Reward function

Action Space (A): The action space A is the set of all feasible actions available to the agent within the environment. In a grid world, these actions may include movements such as going up, down, left, or right.

Transition Probability Function (P): The transition probability function P quantifies the probability of moving from one state to another when a specific action is taken. It can be expressed as $P(s'|s, a)$, where s' represents the next state, s is the current state, and a denotes the action taken.

Reward Function (R): The reward function R specifies the immediate reward that the agent receives when moving from one state to another after taking a particular action. This function can be represented as $R(s, a, s')$, where s is the current state, a is the action taken, and s' is the next state.

In reinforcement learning, the agent aims to maximize the cumulative rewards over time by interacting with the environment. Future rewards are often discounted with a discount factor γ, and the future discounted return at time t is defined as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the time-step at which the interaction with the environment ends.

The optimal action-value function, denoted as $Q^*(s, a)$, represents the maximum expected return achievable by following a certain strategy, given a sequence s and an action a. It can be formally defined as $Q^*(s, a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$, where π is a policy mapping sequences to actions (or probability distributions over actions).

The optimal action-value function follows the Bellman equation. Based on this identity, it can be expressed as follows: if the optimal value $Q^*(s', a')$ for the next sequence s' were known for all possible actions a', then the optimal strategy is to select the action a' that maximizes the expected value of $r + \gamma Q^*(s', a')$:

$$Q^*(s, a) = \mathbb{E}_{s' \sim S}\left[r + \gamma \max_{a'} Q^*(s', a')\right]. \tag{2.1}$$

Many reinforcement learning algorithms aim to estimate the action-value function, using the Bellman equation as an iterative update: $Q_{i+1}(s, a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s', a') | s, a\right]$. These value iteration algorithms converge to the optimal action-value function, $Q_i \to Q^*$ as $i \to \infty$.

In practice, function approximation methods are often employed to estimate the action-value function, represented as $Q(s, a; \theta) \approx Q^*(s, a)$. In this context, a neural network is utilized to approximate the Q function, which is called a Q-network. A Q-network can be trained by minimizing a sequence of loss functions $L_i(\theta_i)$ that evolve at each iteration i. These loss functions are given by:

$$L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot)}\left[(y_i - Q(s, a; \theta_i))^2\right], \tag{2.2}$$

Where $y_i = \mathbb{E}_{s' \sim S}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})\right]$ is the target for iteration i, and $\rho(s, a)$ is a probability distribution over sequences and actions referred to as the behaviour distribution. The weights from the previous iteration, denoted as θ_{i-1}, remain fixed while optimizing the loss function $L_i(\theta_i)$. The targets depend on the network weights, which is different from supervised learning, where targets are typically fixed prior to training.

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s, a \sim \rho(\cdot); s' \sim S}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right) \nabla_{\theta_i} Q(s, a; \theta_i)\right].$$

To update the network weights, stochastic gradient descent is often used, replacing the full expectations in the gradient with single samples from the behaviour distribution ρ

and the environment E. This approach leads to the well-known Q-learning algorithm, which is model-free and operates directly on samples from the environment without explicitly constructing an estimate of the environment E. It's also an off-policy method, meaning it learns about the optimal strategy while following a behaviour distribution that ensures adequate exploration of the state space. In practice, the behaviour distribution is often determined using an ϵ-greedy strategy, which follows the optimal strategy with probability $1 - \epsilon$ and selects random actions with probability ϵ.

Domain Adaptation in Reinforcement Learning

Domain Adaptation: In traditional machine learning, it is assumed that the training and test data come from the same distribution. However, this can be problematic in real-world scenarios where there are various sources of data disparity, such as domain shifts or evolving statistical properties. In response to this challenge, domain adaptation has emerged as a crucial research effort. Its objective is to bridge the distribution gap between source and target domains. Domain adaptation creates a model based on source data and adapts it for effective performance on target data with distinct distributions, mitigating distribution disparities. This is especially useful when getting new, accurately labelled training data that aligns perfectly with the target domain is impractical or time-consuming.

Notably, domain adaptation operates within the broader context of transfer learning, a fundamental element of machine learning. This subfield is integral to enhancing model performance in scenarios with limited labelled data [128].

Domain adaptation involves three key elements: the input or feature space (X), the output or label space (Y), and an associated probability distribution $p(x, y)$. These elements can be represented as $D = \{X, Y, p(x, y)\}$. The feature space X is a subset of a d-dimensional space, while the label space Y can be binary $(-1, +1)$ or multi-class $(1, ..., K)$. The probability distribution $p(x, y)$ describes a joint distribution over the feature-label space pair $X \times Y$. This joint probability distribution can be decomposed into either $p(x)p(x|y)$ or $p(y)p(y|x)$, where $p(.)$ represents a marginal distribution and $p(.|.)$ represents a conditional distribution.

Domain adaptation refers to a scenario where there is a source domain (S) and a target domain (T). The source dataset consists of feature-label pairs, denoted as $(x_i, y_i)_i = 1^n$, where n is the number of samples. On the other hand, the target dataset contains $\{(z_i, u_i)\}$, with m samples, where u_i is unknown in unsupervised domain adaptation due to the absence of labels in the target domain. When the source and target domains are related but differ in their distributions, it is not recommended to transfer knowledge straightforwardly from one domain to another since it can negatively impact the model's performance. Therefore, domain adaptation aims to reduce the distribution gap between these domains and train a model that performs well on target samples. Essentially, the goal is to learn a generalized classifier.

Domain adaptation can be applied to classification, a machine learning task involves

learning a function that maps input data to output labels. This function, denoted as $h : X \to Y$, belongs to a hypothesis space H, which includes all possible functions. The main objective of a classical classification task is to minimize the expected risk associated with the labelled data during model training. This risk, also known as $R_S(h)$, measures the disagreement between the classifier's predictions and true labels in the source domain. On the other hand, in domain adaptation, the focus shifts to learning a model with the lowest risk when applied to the target domain. This risk, referred to as $R_T(h)$, calculates the disagreements between the classifier's predictions and true labels in the target domain, taking into account the differences in probability distributions between the source and target domains.

In classical machine learning, it is assumed that the training and test data come from the same distribution. However, domain adaptation challenges this assumption, as explained in subsequent sections.

Farahani et al. [53] discussed four different categories of domain adaptation, each with its unique characteristics and implications:

Closed Set Domain Adaptation: Within this category, the source and target domains share the same classes, yet a domain gap persists between them. This category encompasses traditional domain adaptation methods.

Open Set Domain Adaptation: Related domains have some overlapping labels while also possessing private labels. An adapted version of open set adaptation considers the source label set as a subset of the target label set. Open-set domain adaptation is particularly useful when multiple source domains each encompass a subset of the target classes. These approaches leverage source domain information from the shared classes to enhance model performance in the target domain.

Partial Domain Adaptation: In this scenario, the target label set is a subset of the source label set. The source domain is viewed as a comprehensive domain with a multitude of classes, whereas the target domain represents only a subset of the source label set, comprising fewer classes.

Universal Domain Adaptation (UDA): UDA, or unsupervised domain adaptation, is a flexible approach that doesn't require prior knowledge about the source and target label sets. It's particularly useful in scenarios where source and target domains may overlap, but may also contain private labels. The goal of UDA is to identify shared labels and align data distributions, similar to open set and partial domain adaptation techniques. Once the alignment is complete, a classifier is trained on the source-labeled data, which can be confidently used to label the unlabeled target data during testing. The trained classifier should accurately assign labels to target samples from shared labels and designate outliers as "unknown" in both open set and universal domain adaptation scenarios.

Reinforcement Learning: There have been approaches proposed in the literature, but many of them primarily focus on creating shared state representations between domains. For instance, one approach to domain adaptation in RL is the use of a two-stage agent that initially learns a Latent Unified State Representation (LUSR) shared across multiple domains [182]. This representation enables the agent to adapt to different domains without additional training. Similar to the first approach, another approach focuses on alignment between domain-invariant state representations [98]. It employs adversarial losses at both the feature level and the pixel level to constrain the learning of domain-invariant state representations. The RL agent is then trained based on these learned domain-invariant state representations. Since the adapted observations are domain-invariant, the learned policy exhibits strong cross-domain generalization performance. This method, referred to as Adversarial-based Domain Invariant State Representation (Ad-DISR), is evaluated in various Car-Racing games and the CARLA autonomous driving simulator. Results show improved performance in terms of reward scores and living time across both source and target domains. One notable challenge in RL is the difficulty of training robotic policies entirely on real hardware due to low sample efficiency and unsafe exploration. To tackle this issue, [9] proposes a novel approach that combines gradient-based meta-learning with generative trajectory models, enabling policies to adapt to various dynamic conditions in simulation and real-world scenarios.

However, as discussed by Velasquez in [168], the current neural-based approaches face limitations when it comes to meeting the demands of practical applications as these methods lack efficiency, transparency, and robustness. These approaches often assume a small gap between simulated environments and target environment, for example, the real world, and fail when otherwise [190]. For instance, consider recent research in the domain of transferring skills from simulated environments to the real world, particularly in scenarios involving bipedal walking while carrying objects. It has been observed that when the simulated environment doesn't accurately model the carried object, the success rate of this transfer drops drastically from 96% to as low as 0.1%. This highlights the significant challenges in adapting and applying these approaches to complex real-world situations where there is a notable gap between simulation and reality [46].

Moreover, deep Reinforcement Learning (DRL) has historically emphasized adaptation to gradual environmental changes in many existing approaches [83, 127, 40]. However, the problem addressed in Chapter 6 revolves around sudden, long-term transformations of the environment. Meta-reinforcement learning enables AI agents to rapidly acquire new skills by learning how to learn based on prior experiences. However, conventional meta-reinforcement learning algorithms require training AI agents on multiple distinct tasks to identify an appropriate initialization of weights or hyperparameters before these agents can tackle new tasks [187]. Our focus, on the other hand, is on scenarios where an agent is trained solely on a single specification of the environment, with baseline agents already exhibiting acceptable performance in the pre-novelty environment [90]. In the context of novelty adaptation in open worlds, transfer reinforcement learning agents often face the challenge of: 1) selecting appropriate source tasks for a given target task, 2)

understanding the relationships between the source and target tasks, and 3) transferring knowledge from the source task to the target task [162]. The most analogous method to our proposed approach, "transfer learning via policy reusing," aims to repurpose policies learned from source tasks to construct a policy for the target task [191]. While we also reuse portions of a policy learned from a trained agent, we partition the policy into multiple regions, reusing those regions that remain effective, and relearning the regions that do not perform as expected. Transfer learning via policy reuse methods, in contrast, generally entails learning multiple policies or Q-value functions for source tasks, with the agent's action determined by the policy that yields the highest Q-value for a given state in the target task [56, 22].

Open-world Learning Approaches

A range of approaches has been developed to address the challenge of open-world learning. Some predominantly focus on novelty detection [110, 32, 99], while others are geared toward novelty adaptation [64, 88, 160, 86, 122]. Among the novelty adaptation approaches, [88] introduced "Rapid Motor Adaptation" (RMA), which enables legged robots to adapt to shifting terrains, payloads, and wear and tear. Similar to our method, RMA comprises a base policy and an adaptation module. However, their base policy requires training in a simulator that varies in environmental parameters, such as robot mass, friction, terrain height, and motor strength. In other words, their method deals with known unknowns, whereas our approach does not assume the nature of novel situations it may encounter, thus making it capable of handling unknown unknowns. HYDRA [86, 160] is a planning-centric agent that can detect and adapt to novelties in open worlds by employing a compositional model of the environment, defined using PDDL+. It harnesses a range of AI techniques, including domain-independent automated planning, anomaly detection, model-based diagnosis, and heuristic search. OpenMIND [122] shares similarities with HYDRA but employs a different planning formalism. OpenMIND consists of a goal-oriented agent that plans to achieve its objectives, dynamically revising its goals and planning models when novel and unexpected situations arise. Similar to [86, 160], [122] also introduces several heuristics proven to be effective in handling novelty. A hybrid planning and learning method called RAPid-Learn [64] has been recently introduced to tackle the same problem, grounded in domain knowledge through PDDL. This method is evaluated in a grid world environment inspired by Minecraft. For RAPid-Learn, the agent must possess knowledge of known entities, predicates, symbolic states in the environment, and the preconditions and effects of available actions. The primary distinction between our method and HYDRA, OpenMIND, and RAPid-Learn is that our approach does not assume an explicit understanding of the environmental structure. This includes the absence of a model of the environment, a predefined set of novelties, handcrafted predicates, preconditions, or results of available actions. Additionally, our method is not based on planning or formal languages, such as PDDL.

2.3 Discretization in Reinforcement Learning

In response to the challenges posed by computation and storage limitations on controller devices, which demand algorithms to excel in three critical aspects: efficient learning, minimal computation, and reduced storage, discretization-based approaches in reinforcement learning have been systematically investigated across various scenarios.

Earlier techniques employing non-uniform discretization are known as variable resolution techniques [117]. Moore [118] introduced the Parti-game algorithm, which automatically generates a variable resolution discretization of a continuous, deterministic domain based on observed data. Unlike standard reinforcement learning algorithms that aim to maximize total reward, Parti-game requires the problem to have a goal state. This can be viewed as a subset of the general reinforcement learning problem, where the agent receives a positive reward only in the goal state. Parti-game also necessitates a greedy local controller to navigate the agent in the state space and a heuristic to determine task failure. The G algorithm [36] and the U Tree algorithm [108] are similar in that they automatically generate a variable resolution discretization by re-discretizing propositional state spaces. A policy for the new discretization can then be derived using traditional techniques. Like Parti-game, both algorithms start with the world as a single state and recursively split it as needed. The Continuous U Tree algorithm [164] extends these methods to continuous state spaces rather than propositional ones. To refine the resolution of adaptive discretizations for solving continuous time-and-space problems, Munos and Moore [121] introduced the concepts of *influence* and *variance* related to a Markov Chain. Influence measures the extent to which a state contributes to the value function of other states. The variance of the Markov Chain derived from MDP provides a good estimation of the bias, indicating which parts of the discretization need refinement to improve approximation accuracy. The region is split when it has the highest product of influence and variance. However, calculating influence and variance requires a model of the world, and this method was applied to deterministic control problems.

Adaptive basis functions are explored as a research topic, where the parameters of the functional model - such as a neural network - are learned online while simultaneously adapting the basis functions. Basis functions are crucial for approximating the value function in reinforcement learning, particularly for problems with large state spaces. For instance, [81] looks into the automatic construction of basis functions using dimensionality reduction techniques like Neighborhood Component Analysis (NCA). Their method maps high-dimensional state spaces to lower-dimensional spaces based on Bellman error or temporal difference (TD) error. These basis functions are then integrated into a linear function approximator, as demonstrated in a high-dimensional inventory control problem. On a related note, [111] underscores the importance of selecting appropriate basis functions for algorithmic success. They present two algorithms for adapting basis functions during the learning process, effectively decoupling the optimization of linear weights from the adjustment of basis function parameters. The optimization is guided by the Bellman approximation error, and two distinct methods are proposed—one based on

gradient descent and the other on the Cross-Entropy (CE) method. Furthermore, [177] introduces the concept of evolutionary function approximation in reinforcement learning. This approach combines neuroevolutionary optimization techniques with Q-learning, resulting in NEAT+Q, a method for automatically discovering effective representations for neural network function approximators. [177] also presents online evolutionary computation, which enhances reinforcement learning performance and offers interpretable policies.

More recently, several works in this domain focus on tree-based partitioning rules. Researchers[135, 103] explore the hierarchical partitioning of the state and action space. This partitioning is refined over time and tested heuristically with various splitting rules, such as the Gini index. In contrast, [155] introduces an approach that splits regions based on the metric and statistical uncertainty in the estimates. This work addresses how to create discretizations, as well as when to refine them, offering a unified theoretical analysis of adaptive hierarchical partitioning methods. [155] demonstrate how these algorithms leverage problem structure to provide guarantees, scaling with respect to the "zooming dimension" rather than the ambient dimension. While [155] primarily focus on finite time horizon Markov Decision Processes (MDPs), [5] and [6] extend adaptive discretisation techniques to employ a single partition in infinite horizon time-discounted settings. Their algorithms were benchmarked on various control tasks in OpenAI, showcasing comparative performance between discretisation-based and deep learning techniques. Specifically, [5] addresses the exploration-exploitation trade-off using a combination of Upper Confidence Bounds (UCB) and Boltzmann exploration. Their SPAQL algorithm is compared with Adaptive Q-learning (AQL), highlighting scalability and sample efficiency. SPAQL's contribution lies in offering a novel reinforcement learning algorithm that adapts its discretisation on the fly, thereby improving both convergence speed and sample efficiency. [6] evaluates two reinforcement learning algorithms, Adaptive Q-learning (AQL) and Single-Partition Adaptive Q-learning (SPAQL), in classical control problems, including Pendulum and CartPole.

It is important to note that these methods differ from NAPPING in that they focus on learning the Q-value function from scratch by partitioning the state space. In contrast, NAPPING focuses on domain and novelty adaptation problems when a trained Q-value function is already available, requiring the algorithm to adapt to target environments with novelties using the baseline agent.

Chapter 3

Evaluating Physical Reasoning Intelligence

This chapter is curated from "Phy-Q as a measure for physical reasoning intelligence" [184]. As one of the first authors, I introduced the concept of assessing agents' physical reasoning capabilities, akin to those acquired by humans during infancy. Additionally, I proposed the evaluation of two distinct generalization capacities: narrow and broad generalization. Spearheading these efforts, I formulated, tested, and computed the Phy-Q measure. Furthermore, I conceptualized and implemented the server-game-agent testing framework, overseeing the execution of experiments for all baseline agents. In addition to designing the experiments related to the baseline agents, I authored the corresponding sections and meticulously proofread other segments, providing comprehensive comments for improvement.

3.1 Introduction

The ability to reason about objects' properties and behaviours in physical environments lies at the core of human cognitive development [186, 47]. Few days after birth, infants understand object solidity [165] and within the first year of birth, they understand notions such as object permanence [14], spatiotemporal continuity [95], stability [18], support [16], causality [145], and shape constancy [48]. Generalization performance on novel physical puzzles is commonly used as a measure of physical reasoning abilities for children [49, 37], animals [130, 51, 161, 75], and AI agents [19, 3, 4, 1].

Chollet's study [44] on the measure of intelligence proposes a qualitative spectrum of different forms of generalization that includes *local generalization* and *broad generalization*. Current evidence [89, 76, 106, 77] suggests that contemporary deep learning models are local-generalization systems, i.e., systems that adapt to known unknowns within a single task. *Broad generalization*, on the other hand, can be characterized as

'adaptation to unknown unknowns across a broad category of related tasks' and is being increasingly emphasized among the AI research community [44, 61, 58]. Moreover, when solving physics puzzles, it is common that a player must use a strategy to work out a plan and must use dexterity to accurately execute the strategic plan [71]. For instance, in a snooker game, a player needs to plan the path of the white cue ball, e.g., where it should go and where to stop, and then execute the strike that precisely produces to the planned path. One of the views of cognitive psychology researchers is that humans possess inaccurate forward physics prediction models [109, 157, 149, 79, 62, 97, 93, 134] and hence require practice to improve dexterity, high dexterity requirements of physics tasks make it unfair to compare AI agents' physical reasoning ability with average humans'. For example, when a human player fails a physics puzzle, it is hard to tell if it is due to incorrect physical reasoning or due to the inability of making precise actions. Despite the recent advancement in physical reasoning benchmarks and testbeds [19, 3, 4, 142, 186, 21, 1, 25, 45], there is a lack of a benchmark or a testbed with human comparable strategic physics puzzles and one that explicitly evaluates learning agents' local and broad generalization.

To close these gaps, we propose a new testbed Phy-Q and the associated Phy-Q score that measures physical reasoning intelligence using the physical scenarios we identified. Inspired by the physical knowledge acquired in infancy and the abilities required by the robots to operate in the real world, we have created a wide variety of tasks with low dexterity requirements in the video game Angry Birds. Also, we included a module to aid trajectory planning. We believe the contributions from this paper pave the way to develop agents with human level strategic physical reasoning capabilities. Our main contributions can be summarized as follows:

- **Phy-Q: A testbed for physical reasoning:** We designed a variety of task templates in Angry Birds with 15 physical scenarios, where all the task templates of a scenario can be solved by following a common strategic physical rule. Then we generated task instances from the templates using a task variation generator. This design allows us to evaluate both the local and the broad generalization ability of an agent. We also define the Phy-Q score, a quantitative measure that reflects physical reasoning intelligence using the physical scenarios we considered.
- **An agent-friendly framework:** We propose a framework, which allows training multi-agent instances simultaneously with accelerated game-play speed up to 50 times.
- **Establishing results for baseline agents:** The evaluation consists of nine baseline agents: four of our best performing learning agents, four heuristic-based agents, and a random agent. For each of the baseline agents, we present the Phy-Q score, the *broad generalization* performance, and the *local generalization* performance. We have collected human player data so that agent performance can be directly compared to human performance.
- **A guidance for agents in AIBIRDS competition:** Angry Birds is a popular physical reasoning domain among AI researchers with the long running AIBIRDS com-

petition since 2012 [140]. In 2016, Angry Birds was considered to be the next milestone in AI where AI will surpass humans [67]. A time horizon of four years was predicted and so far such a breakthrough seems very unlikely. In the AIBIRDS competition, heuristic methods generally perform better than their deep learning counterparts, but it remains unclear what has contributed to the gap in the performances. It has also not yet been analysed why current AI agents fall short when comparing to humans. By the systematic analysis of agents from the AIBIRDS competition, we show how they need to be improved to achieve human-level performance.

3.2 Phy-Q Testbed

In this section, we introduce our testbed, discuss the physical scenarios we have identified, and explain the evaluation settings we have used in the testbed.

3.2.1 Introduction to the Phy-Q Testbed

Based on the 15 identified physical scenarios (discussed in detail in Section 3.2.2), we develop a physical reasoning testbed using Angry Birds. In Angry Birds, the player interacts with the game by shooting birds at the pigs from a slingshot. The goal of the player is to destroy all the pigs using the provided set of birds. As the original game by Rovio Entertainment [52] is not open-sourced, we use a research clone of the game developed in Unity [57]. The game environment is a deterministic 2D world where objects in motion follow Newtonian physics. The game objects are of four types: birds, pigs, blocks, and platforms. There are five types of birds in which four of them have powers that can be activated once tapped in their flight. There are three types of pigs varying in size, the health points of the pigs increase with the increase in size. Blocks in the game are made of three materials (wood, ice, and stone) and each of them has 12 variations in shape. Platforms are static objects that remain at a fixed position and are not affected by forces and are indestructible. All other objects are dynamic, i.e., they can be moved by applying forces. Dynamic objects have health points that get reduced upon collisions with other objects and they get destroyed and disappear when health points reach zero. The initial state of a game level is physically stable (i.e., none of the objects is in motion) and the goal is not achieved. The action of an agent is to shoot the bird on the slingshot by providing the release coordinates relative to the slingshot and the tap time of the bird to activate powers (if available). This means the action space is essentially continuous. When playing, an agent takes a sequence of actions, i.e., shoots the birds in a predefined order. The agent passes a game level when it destroys all pigs with the provided set of birds, and fails otherwise. We do not provide the full world state that includes the exact location of objects in the simulator or their physical properties such as mass and friction to the agents as these properties are not directly observable in the real world. Instead, an agent can request screenshots and/or a symbolic representation of the game level at any time while playing. A game screenshot is a 480 x 640 coloured image and the symbolic representation is in JSON format containing all

objects in the screenshot represented as a polygon of its vertices (provided in order) and its respective colour map. The colour map provides the list of 8-bit quantized colours that appear in the game object with their respective percentages. Below is an example symbolic representation state:


```
[
  {
    "type": "FeatureCollection",
    "features": [
      {
        "type": "Feature",
        "geometry": {},
        "properties": {
          "id": "-472",
          "label": "Ground",
          "yindex": 300,
          "colormap": []
        }
      },
      {
        "type": "Feature",
        "geometry": {
          "type": "MultiPoint",
          "coordinates": [
            [195, 188], [391, 108], [591, 143]
          ]
        },
        "properties": {
          "id": "-18206",
          "label": "Trajectory",
          "colormap": []
        }
      },
      {
        "type": "Feature",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [
              [207, 216], [207, 167], [224, 167], [224, 216]
            ]
          ]
        },
        "properties": {
```

```
        "id": "-608",
        "label": "Slingshot",
        "colormap": [],
        "currentLife": 3.402823e+38
      }
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [587, 188], [587, 187], [587, 184], [583, 184], [581, 183]
        ]
      },
      "properties": {
        "id": "-716",
        "label": "pig_basic_medium_1",
        "colormap": [
          {"color": 0, "percent": 0.5},
          {"color": 4, "percent": 0.5}
        ],
        "currentLife": 2.5
      }
    }
  ]
}
]
```


This JSON structure represents a symbolic state representation of the Angry Birds game, using a combination of geometric shapes and properties to describe key elements within the game environment. It is structured as a "FeatureCollection", where each "Feature" contains a distinct object within the game. The first feature represents the "Ground" with an empty geometry, characterized by an "id" of "-472" and a vertical position ("yindex") of 300. The second feature describes the "Trajectory" of the bird, modeled as a "MultiPoint" geometry with three specific coordinates, indicating the path of the bird as it moves through the air. The "Slingshot" is represented as a polygon, defined by its coordinates to form a rectangular shape. Its properties include an "id" of "-608" and an extremely high value for "currentLife," indicating that the slingshot is essentially indestructible. Lastly, the pig, labelled "pig_basic_medium_1," is also represented as a polygon, with coordinates defining its location. The "colormap" attribute specifies color values and their percentages, while its "currentLife" is set to 2.5, indicating the pig's remaining health. This symbolic representation effectively encodes both the spatial and functional attributes of the game components, allowing for the analysis or manipulation of the game state.

3.2.2 Physical Scenarios in Phy-Q Testbed

In this section, we explain the 15 physical scenarios we consider in our testbed. Firstly, we consider the basic physical scenarios associated with applying forces directly on the target objects, i.e., the effect of a single force and the effect of multiple forces [144]. On top of the application of single force, we also include scenarios associated with more complex motion including rolling, falling, sliding, and bouncing, which are inspired by the physical reasoning capabilities developed in human infancy [29, 78]. Furthermore, we define the objects' relative weight [174], the relative height [15, 69], the relative width [170], the shape differences [123, 17], and the stability [180] scenarios, which require physical reasoning abilities infants acquire typically in a later stage. On the other hand, we also incorporate clearing path, adequate timing, and manoeuvring [82], and taking non-greedy actions [87], which are required to overcome challenges for robots to work safely and efficiently in physical environments. To sum up, the physical scenarios [1] we consider and the corresponding high-level strategic physical rules that can use to achieve the goal of the associated tasks are mentioned below. Each of these scenarios tests a different aspect of the agent's skill, physical understanding, and planning ability.

1. **Single force:** Target objects have to be destroyed with a single force.
2. **Multiple forces:** Target objects need multiple forces to destroy.
3. **Rolling:** Circular objects have to be rolled along a surface to a target.
4. **Falling:** Objects have to fall onto a target.
5. **Sliding:** Non-circular objects have to be slid along a surface to a target.
6. **Bouncing:** Objects have to be bounced off a surface to reach a target.
7. **Relative weight:** Objects with correct weight have to be moved to reach a target.
8. **Relative height:** Objects with correct height have to be moved to reach a target.
9. **Relative width:** Objects with correct width or the opening with correct width have to be selected to reach a target.
10. **Shape difference:** Objects with correct shape have to be moved/destroyed to reach a target.
11. **Non-greedy actions:** Actions have to be selected in the correct order based on physical consequences. The immediate action may be less effective in the short term but advantageous in long term. i.e., reach fewer targets in the short term to reach more targets later.
12. **Structural analysis:** The correct target has to be chosen to break the stability of a structure.

[1] Example tasks representing the 15 scenarios and the solutions for those tasks are available at: https://github.com/phy-q/benchmark

13. **Clearing paths:** A path have to be created before the target can be reached.
14. **Adequate timing:** Correct actions have to be performed within time constraints.
15. **Manoeuvring:** Objects have to be carefully guided to reach a target.

3.2.3 Task Templates and Task Generation

We design task templates in Angry Birds for each of the 15 physical scenarios mentioned above. A task template can be solved by a specific strategic physical rule and all the templates belonging to the same scenario can be solved by the high-level strategic physical rules discussed above. The task templates are handcrafted and we ensure that if an agent understood the strategic physical rule to solve the template, they can solve the template without requiring high accurate shooting, e.g., the template can be solved by shooting at a specific object rather than shooting a specific coordinate. This design criterion is followed to reduce the dexterity requirement when solving the tasks in our testbed. We have developed 2-8 task templates for each scenario totalling 75 task templates. Figure 3.1 shows example task templates for the 15 scenarios. These fifteen templates describe distinct strategies for solving a problem involving the destruction of a pig, each highlighting different physical and logical mechanisms. The first two templates, "Single force" and "Multiple forces," involve applying either one or several forces to directly target and destroy the pig. In "Rolling," the destruction occurs by rolling a circular object, and in "Falling," the object must fall onto the pig. "Sliding" requires sliding a square object to hit an otherwise unreachable pig. The "Bouncing" template involves bouncing a bird off a platform to reach the target. Templates such as "Relative weight," "Relative height," and "Relative width" emphasize the importance of size, height, and width in solving the game level, while "Shape difference" explores the impact of shape in determining which block can release the circular object to destroy the pig. "Non-greedy actions" suggests a counterintuitive approach where a less direct strategy, targeting a lower pig first, is more effective. "Structural analysis" identifies weak points in a structure to destabilize it and destroy multiple pigs with a single shot. "Clearing paths" focuses on positioning objects to open routes for destruction. "Adequate timing" stresses the importance of precise timing between two rolling objects to destroy the pig. Finally, "Manoeuvring" involves using a special capability, e.g. the blue bird split into three birds after a tap, to solve the game level.

We generate 100 game levels from each template and we refer to these game levels as tasks of the task template. All tasks of the same template share the same strategic physical rule to solve. Similar to [19], the tasks are generated by varying the location of the game objects in the task template within a suitable range. Furthermore, various game objects are added at random positions in the task as distractions, ensuring that they do not alter the solution of the task. Although we provide 100 tasks for each task template, we also provide a task variation generation module to generate more tasks if needed. Figure 3.2 shows task templates of relative height scenarios and example tasks generated from a single task template. All the 75 task templates and example task

variations can be found in Supplementary C.

3.2.4 Proposed Evaluation Settings

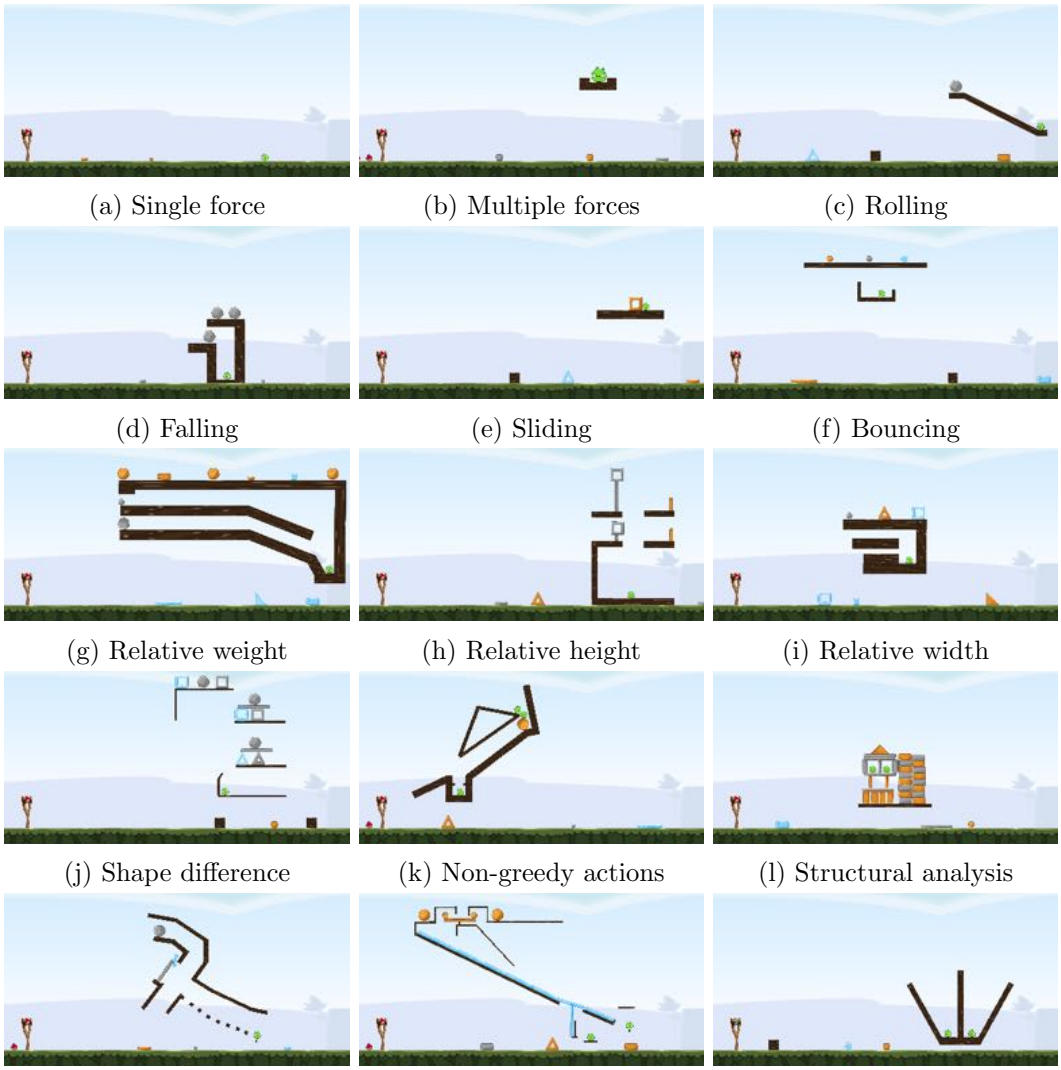
The spectrum of generalization proposed by Chollet [44] can be used to measure intelligence as laid out by theories of the structure of intelligence in cognitive psychology. There are three different levels in the spectrum: *local generalization*, *broad generalization*, and *extreme generalization*. Having 15 physical scenarios, a variety of task templates for each scenario, and task variations for each task template, our testbed is capable of evaluating all of the three different generalization levels. However, in this work, we focus on measuring the *local generalization* and the *broad generalization* of agents, as *local generalization* is the form of generalization that has been studied from 1950s up to this day and there is an increasing research interest in achieving *broad generalization* [44].

More formally, consider each scenario $scenario_i$ in the set of all scenarios $SCENARIO$, where $|SCENARIO| = 15$, we define template $template_j \in scenario_i$, where $|scenario_i| = NT_i$ and NT_i is the number of templates we included for $scenario_i$. As we have 100 tasks for each templates, we define $task_k \in template_j$, where $|template_j| = 100$ for all templates.

To evaluate *local generalization*, we train an agent on a subset of the tasks from a template j, $TrainTasks^j_{local} \subset template_j$ and test on the rest of the tasks within the same template as $TestTasks^j_{local} = template_j \setminus TrainTasks^j_{local}$. We evaluate the *local generalization* for all of the 75 templates.

For evaluating the *broad generalization*, we train an agent on the training tasks on a subset of templates, $TrainTasks^i_{broad} = \cup_{\{j|template_j \in scenario_i^{train}\}} TrainTasks^j_{local}$, and evaluate on the testing tasks of the rest of the templates within the same scenario, $TestTasks^i_{broad} = \cup_{\{j|template_j \in scenario_i^{test}\}} TestTasks^j_{local}$, where $scenario_i^{train}$ is the set of training templates and $scenario_i^{test}$ is the testing templates for $scenario_i$. We also make sure that $\forall i, scenario_i^{train} \cap scenario_i^{test} = \emptyset$. We evaluate the *broad generalization* performance for all 15 scenarios. We assume that if an agent learns the required strategic physical rule to solve a set of task templates, it should be able to apply the same strategic physical rule to solve unseen tasks from other templates within the same scenario. As opposed to this, the performance on *local generalization* evaluation may not represent an agent's physical rule generalizing capability but memorizing a special-purpose heuristic. Figure 3.3 is a diagrammatic representation of the two evaluation settings and Figure 3.4 shows an illustration of how generalizing a physical rule is evaluated in the *broad generalization* evaluation setting.

Our physical reasoning quotient (Phy-Q) is inspired by deviation IQ [176] for humans. We calculate Phy-Q of an agent using the results of our *broad generalization* evaluation since we consider that this evaluation measures the agent's ability in generalizing strategic physical rules. When calculating the Phy-Q we exclude the first two scenarios: single force and multiple forces, as the solution for the two scenarios is directly shooting the



(a) Single force (b) Multiple forces (c) Rolling

(d) Falling (e) Sliding (f) Bouncing

(g) Relative weight (h) Relative height (i) Relative width

(j) Shape difference (k) Non-greedy actions (l) Structural analysis

(m) Clearing paths (n) Adequate timing (o) Manoeuvring

Figure 3.1: 15 example tasks in Phy-Q representing the 15 physical scenarios. The slingshot with birds is situated on the left of the task. The goal of the agent is to kill all the green-coloured pigs by shooting birds from the slingshot. The dark-brown objects are static platforms. The objects with other colours are dynamic and subject to the physics in the environments. A detailed explanation of the 15 tasks and corresponding solutions can be found in our GitHub repository https://github.com/phy-q/benchmark

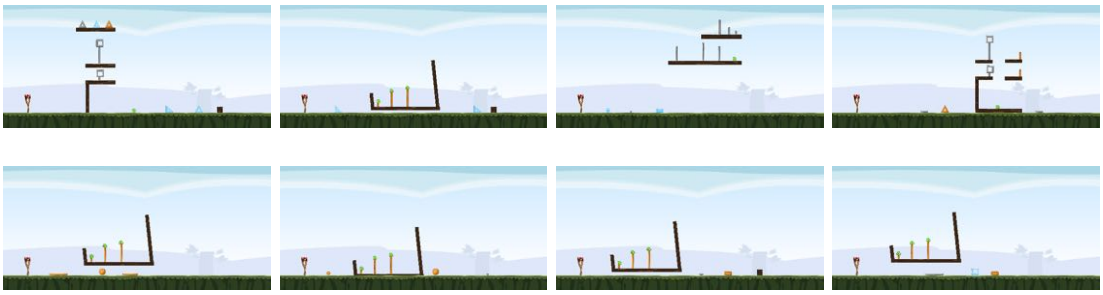


Figure 3.2: The first row shows the task templates of the relative height scenario and the second row shows the tasks generated using the second task template in the first row.

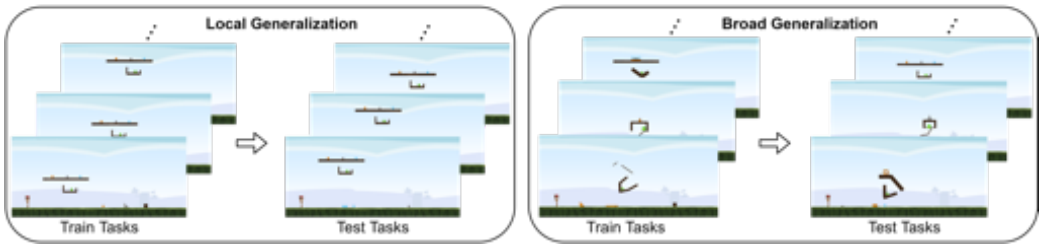


Figure 3.3: The *local generalization* and the *broad generalization* evaluation settings.

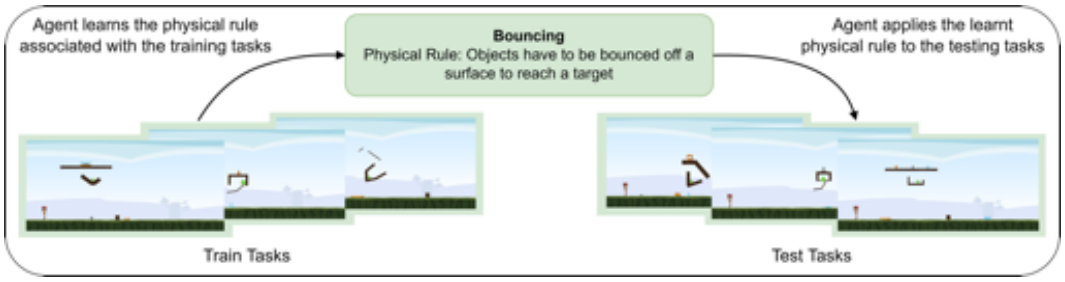


Figure 3.4: An illustration of how generalizing a physical rule is evaluated in the *broad generalization* evaluation using bouncing scenario as an example.

bird to the exact location of the pig. Given that we have provided a trajectory planner for both humans and agents, solving the tasks of the two scenarios is straightforward. This is also evident from the exceptionally high results (in Section 3.4.2) of the Pig Shooter agent that directly shoots at the pigs without doing any physical reasoning. We define the Phy-Q score as follows:

$$Z_{agent} = \frac{1}{|SCENARIO - \{scenario_1, scenario_2\}|} \sum_{m=3}^{|SCENARIO|} \frac{P_{agent,m} - P_{human,m}}{\sigma_{human,m}} \quad (3.1)$$

$$Phy\text{-}Q\ score_{agent} = 100 + Z_{agent} \frac{100}{|Z_{random}|} \quad (3.2)$$

where $P_{n,m}$ is the average pass rate of subject n in the m^{th} scenario. $\sigma_{human,m}$ is the standard deviation of human pass rate in scenario m and *random* represents the random agent which selects a random action (See Section 3.3.1). A Phy-Q score of 100 represents the agent has an average human level performance and if the score is less than 100, the agent's performance is less than the average humans' performance and vice versa. As the random agent does not have any physical reasoning capabilities, we bring the random agent's Phy-Q score to zero. Therefore, the interval for each standard deviation of Phy-Q is set to be 13.58 ($= 100/|Z_{random}|$) as compared to 15 of IQ. Therefore, a Phy-Q score more than zero indicates a performance better than an agent that selects a random action.

3.3 Experiments

We conduct experiments on baseline learning agents to measure how well they can generalise in two different settings: *local generalization* and *broad generalization*. We also conduct experiments using heuristic baseline agents in the two generalization settings. In addition, we establish human performance on the 15 scenarios. Further, we conduct an additional experiment using heuristic agents in AIBIRDS competition game levels to examine if the performance of agents in the testbed resembles the performance in the competition.

3.3.1 Baseline Agents

We present experimental results of nine baseline agents: two DQN agents - one uses screenshot as input and the other uses symbolic representation, two relational agents - one uses screenshot as input and the other uses symbolic representation, four heuristic agents from the AIBIRDS competition, and a random agent.

Learning Agents For learning agents, we evaluate value-based and policy-based reinforcement learning algorithms. We evaluate Double Dueling Deep Q-network (D-DDQN)

[172, 167] for value-based and Proximal Policy Optimization (PPO) [147] and A2C [113] for policy gradient agents.

- **Deep Q-network (DQN):** The DQN [115] agent collects state-action-reward-next state quadruplets at the training time following decaying epsilon greedy. The quadruplets are then stored in a *replay buffer* $E = \{e_1, ..., e_n\}$, where each $e_i = (s_i, a_i, r_{i+1}, s_{i+1})$ is an *experience*. We define the reward function as task pass status, meaning the agent receives 1 if the task is passed and 0 otherwise. The agent uses a discretized action space of 180 actions, where each corresponds to a release degree from the slingshot with a maximum stretch. At the end of each update step, the agent trains a Deep Q-network on the sampled experiences to predict the Q-value of actions for a given state. In our experiments, we use Double Dueling Deep Q-network [167, 172], which comprises of a state encoder that transforms an input state into a hidden representation through convolutional filters, which is then separated into a state-value and advantage streams. We tested four improvements upon a vanilla DQN: Double DQN [167], Dueling DQN [172], Prioritized Experience Replay [146] and Double Dueling DQN, each with two different input types: symbolic representation and screenshot. For symbolic representation, we map each game state to a $h \times w \times o_t$ tensor, where o_t is the number of object types. In our experiments, we set $h = 120$, $w = 160$, and $o_t = 12$. For the state encoder, we use a 1×1 convolutional filter to squeeze the channels to 1 and then use $h \times w$ filters to extract features from the representation that preserves crucial spatial information. For agents with screenshot state representation, we re-scale and normalise the image to $h = 120$, $w = 160$, and use ResNet-18 [68] to extract features and estimate Q-values. We trained both of the networks using an Adam optimizer [85] with a learning rate of 0.0003. We refer the Dueling Double DQNs we experimented with the two different input types as D-DDQN-Image and D-DDQN-Symbolic.

- **Policy Learners:** To support the development of the learning agents and to allow using existing RL libraries, our framework follows the OpenAI Gym [33] requirements. This allowed us to evaluate the two multi-processing agents from the Stable-Baselines3 [136]: A2C [113] and PPO [147]. We trained the two agents with both discrete and continuous action spaces, different training settings, and various hyper-parameters, but similar to [19], policy-gradient methods did not show good results compared to DQN and were unable to converge to the good policy in a reasonable time. Therefore, we exclude these two agents in the results section.

- **Relational Deep Q-network**: The relational agent consists of the relational module [188] that was built on top of the deep Q-network. The aim of this agent is to generalize over the presented templates/events by using the structured perception and relational reasoning. In our experiments, we wanted to test if the relational agent would be able to learn the important relations between the objects that could be generalized to other templates or events. We have tested the agent with symbolic and image input types and refer to them as Relational-symbolic and Relational-image agents respectively. For the image agent, we use ResNet-18 as the feature extractor and feed its output

to the Multi-head dot product attention module (MHDPA) [188]. The output of the MHDPA is then used to compute the policy. For the symbolic agent, the setting is similar to the symbolic DQN agent except that the output of the convolutional layer is passed to the relational module.

Heuristic Agents: The heuristic agents are based on hard-coded strategic physical rules designed by the developers. We included four heuristic agents from AIBIRDS competition. We compare heuristic agents' performance on our testbed with the generalization performance of the baseline learning agents.

- **Bambirds:** Bambirds is the winner of 2016 and 2019 AIBIRDS competitions. The agent chooses one of nine different strategies. The strategies include creating a domino effect, targeting blocks that support heavy objects, maximum structure penetration, prioritizing protective blocks, targeting pigs, and utilizing certain bird's powers [55].
- **Eagle's Wing:** Eagle's Wing is the winner of 2017 and 2018 AIBIRDS competitions. This agent selects action based on strategies including shoot at pigs, destroy most blocks, shoot high round objects, and destroy structures [171].
- **Datalab:** Datalab is the winner of 2014 and 2015 AIBIRDS competitions. The agent use strategies: destroy pigs, destroy physical structures, and shoot at round blocks. The agent selects the strategy based on the game states, possible trajectories, bird types, and the remaining birds [30].
- **Pig Shooter:** The strategy of the Pig Shooter is to directly shoot at the pigs. The agent shoots the bird on the slingshot by randomly selecting a pig and a trajectory to shoot the pig [159].

Random agent: For each shot, the agent selects a random release point (x, y), where x is sampled from $[-100, -10]$ and y from $[-100, 100]$ relative to the slingshot. It also provides a tapping time that is when the bird is in between 50% and 80% of the trajectory length where applicable.

3.3.2 Experimental Setups

Human experiment setup: Experiments were approved by the Australian National University committee on human ethics under protocol 2021/293. We recruited 20 volunteers for the experiment. The volunteers represented males and females and were around the age range 18-35. They were not experienced Angry Birds players. For each of them, we provided two tasks from each physical scenario for the 15 scenarios in Phy-Q (except the manoeuvring scenario which used four tasks representing the four types of birds with powers). We provide a trajectory visualizer of the bird to the participants to remove the need of precise shooting. If the participants solved a task or failed to solve a task in five attempts, they move to the next task. As humans acquire physical reasoning capabilities from their infancy [29, 78], using an evaluation setting we proposed for agents does not

exactly measure the generalization ability of humans. Therefore, we measure the task performance in humans using the pass rate.

D-DDQN and Relational DQN experimental setup: We conduct separate experiments on the D-DDQN and Relational agents in the two settings: *local generalization* and *broad generalization* setting. For the *local generalization* evaluation, to speed up the training, we run 10 sampling agents that use the same DQN model to collect experiences. Each sampling agent runs on the randomly selected task for 10 episodes. After the set of experiences is collected, the DQN model is trained for 10 epochs with randomly sampled batches of size 32. We train DQN until it either converges or reaches N update steps, where N is the number of training tasks per template divided by 5. Similar to [19], for each batch, we sample 16 experiences that solve a task and 16 that do not. We train our agent on 80% of the tasks of the task template and evaluate on the rest of the 20% tasks of the same template. As to discourage hyper-parameters tuning, we used the same training setting for all of the task templates. At the testing time, the agent runs on each of the testing tasks only once and selects the action that has the highest Q-value for a given state. For *broad generalization* evaluation, we use the same training and testing setting as in the *local generalization* evaluation, except we train our agents on the tasks in the training templates in each scenario and test on the tasks from the testing templates.

Heuristic agents experimental setup: We conduct two experiments using the AIBIRDS heuristic agents. The first experiment is to evaluate the *local and broad generalization* capabilities and the second is to evaluate the performance in the AIBIRDS competition game levels.

- ***Local and broad generalization setup:*** Due to the randomness in the heuristic agents, we allow them to have five attempts per task and calculate the task pass rate by averaging the result over these five attempts. For the *local generalization* setting, the agents were tested on the same 20% of the test tasks from each task template (1500 tasks in total) used for the D-DDQN evaluation. We report the *local generalization* performance by averaging the pass rates of all templates. For the *broad generalization* setting, the same testing templates used for the D-DDQN evaluation were used and the within scenario pass rate is calculated by averaging over all the tested templates within the scenario.
- **AIBIRDS competition setup:** We evaluate the AIBIRDS heuristic agents on 2021 AIBIRDS competition game levels to compare their performance in the competition game levels and the Phy-Q testbed tasks. We exclude the competition game levels with unrealistic effects as our focus in the testbed is scenarios with realistic physics. The game levels used for this evaluation are shown in Supplementary G. In the AIBIRDS competition, the agent with the highest score wins the competition. Therefore in this experiment, we record the score and pass rate of the agents. The agents are allowed to have five attempts per game level to account for their randomness. Altogether an

agent had 40 plays.

Random agent experimental setup: The random agent was tested on the same set of testing tasks from each task template with 1500 tasks. As to reduce the bias in pass rate, we run the random agent 50 times per task and report the pass rate using the average of these 50 attempts. Same as how we evaluate the heuristic agents, we further average the task performance within the same task template and average the pass rate of all the templates to present the *local generalization* performance. For the *broad generalization* setting, within scenario pass rate is calculated by averaging over all the tested templates within the scenario.

3.4 Results and Analysis

In this section, we first present and analyze the results obtained from our experiment with human players. Next, we present the results obtained from our experiments in measuring the local and broad generalization ability of agents and Phy-Q score. We further analyze the results and discuss what we can derive from the experiments. We also discuss the results obtained from the heuristic agents in the 2021 AIBIRDS competition levels and Phy-Q testbed tasks to show how the testbed can be used as a guidance for the competition.

3.4.1 Human Performance

Figure 3.5 presents the average pass rate, the pass rate the human players achieved within five attempts, the maximum number of attempts made, and the total thinking time of human players for the 15 capabilities. The average pass rate is calculated as 100% if the player passes in the first attempt and if the player passes in the fifth attempt, the pass rate is 20%. We record the thinking time of an attempt as the time between the task loading and the player making the first action. The total thinking time of a player is the sum of the thinking time of all his/her attempts. The number of attempts made and the total thinking time are scaled to 0-1 using min-max scaling in Figure 3.5. Charts with the real values are available in Supplementary F.

Overall, human players have passed almost all the tasks in each scenario within the five attempts. On average, they used 1.86 attempts per task and took 23.73 seconds to think per task. On average, the low number of attempts to pass the tasks shows that the dexterity required to solve the tasks when the strategy is determined is low. The average thinking time per task shows that humans have to carefully think about the strategic physical rule required to solve the task.

Humans have the longest thinking time for the tasks in the adequate timing scenario but the average pass rate for these tasks is the second lowest. Similarly, the tasks from the non-greedy actions scenario have the lowest average pass rate with the highest number of attempts while the thinking time is the second longest. This shows that figuring out

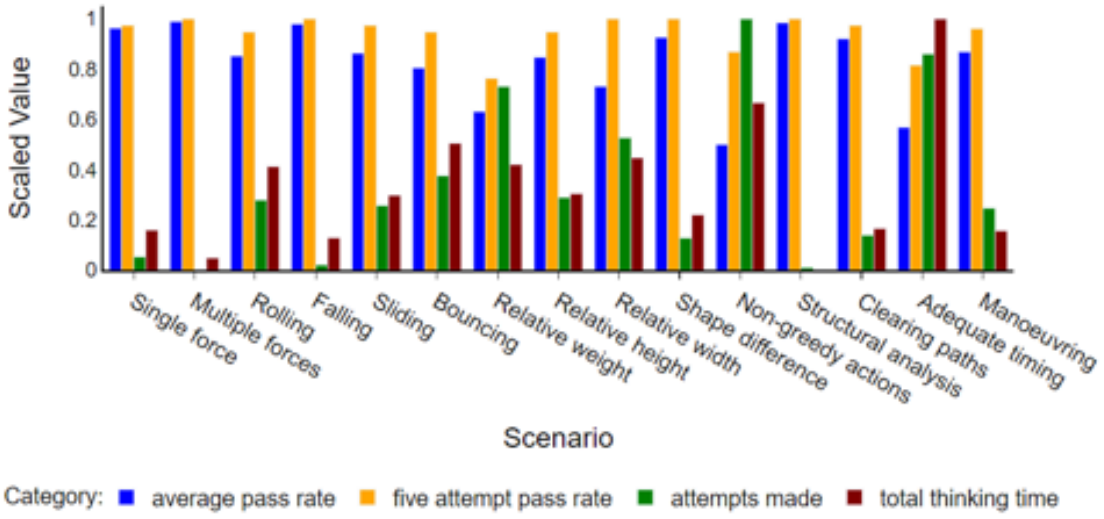


Figure 3.5: Performance of human players. The x-axis is the scenario and y-axis is the scaled value.

Table 3.1: Comparison of agents for *local generalization* and *broad generalization* (Phy-Q score). The results are shown for the nine agents: Bambirds (BB), Eagle's Wing (EW), Datalab (DL), Pig Shooter (PS), D-DDQN-Symbolic (D-DDQN-Sy), D-DDQN-Image (D-DDQN-Im), Relational-Symbolic (Rel-Sy), Relational-Image (Rel-Im), and Random. Learning agents have higher *local generalization* values and lower values in *broad generalization* than performance of heuristic agents. Human performance is way beyond agents.

	Human	BB	EW	DL	PS	D-DDQN-Sy	D-DDQN-Im	Rel-Sy	Rel-Im	Random
Local gen.	-	0.15	0.14	0.15	0.11	**0.34**	0.25	0.32	0.24	0.03
Broad gen.	0.83	0.18	**0.20**	0.19	0.13	0.12	0.10	0.14	0.09	0.04

the correct strategies for the tasks of these scenarios has been difficult for humans. In the relative weight scenario, the pass rate achieved within five attempts is the lowest. But the thinking time is average for this scenario. This suggests that some humans take the action without carefully thinking the strategy, and the strategy realized at a glance is not the correct strategy to solve the task. This also agrees with our observation that humans are overconfident in their wrong actions.

3.4.2 Local and Broad Generalization Performance and Phy-Q Score

***Local generalization* performance** Table 3.1 first row presents the average *local generalization* evaluation pass rate for all of our baseline agents, we have also included the full results of pass rate per agent per template in Supplementary D. The table shows that the four learning agents perform significantly better than their heuristic

counterparts. While both the symbolic learning agents and both the image learning agents on average pass approximately 33% and 24% of the test levels respectively, the previous champions in AIBIRDS competition-Bambirds, Eagle's Wing, and Datalab-pass around only half of the levels as compared with the learning agents, averaging 15%, 14%, and 15% respectively. This agrees with what is generally accepted that deep learning systems can perform a single narrow task much better than heuristic methods when enough densely sampled training data is available.

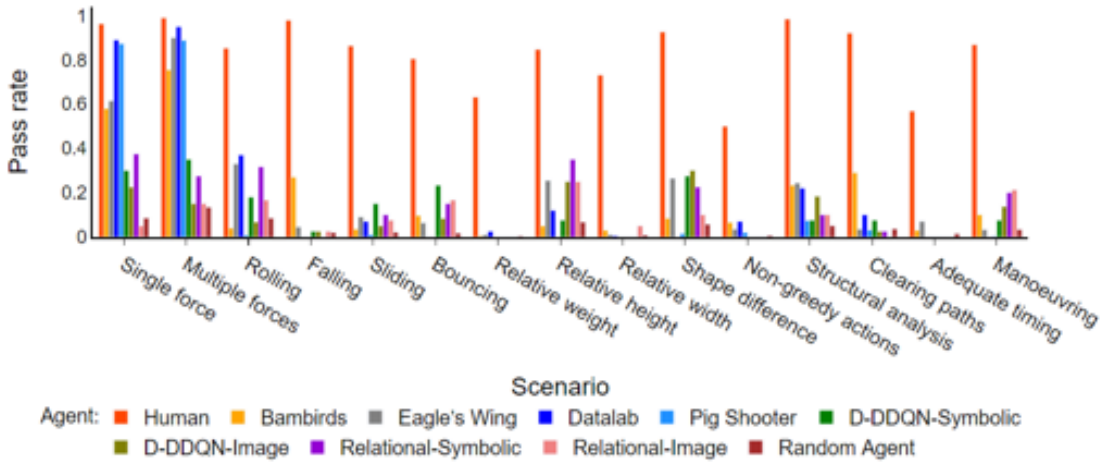


Figure 3.6: *Broad generalization* of the baseline agents. The x-axis is the scenario and y-axis is the passing rate.

***Broad generalization* performance** Figure 3.6 presents the average pass rate of test templates of the *broad generalization* evaluation of all the baseline agents and human players. It is clear that the humans substantially outperform all the other agents, while all the agents have above-chance performance compared to the random agent. Heuristic agents achieved a better pass rate in the single force scenario (scenario 1) and multiple force scenario (scenario 2) as these two scenarios correspond to the essential ability needed to play Angry Birds - shooting directly at pigs. It can be seen that the heuristic agents generally perform better if the physical scenario is covered in their built-in rules. For example, Datalab and Eagle's Wing have a built-in strategic physical rule to roll round objects and they have the highest pass rate in scenario 3 (rolling) among all agents. For scenario 4 (falling) and scenario 13 (clearing paths), Bambirds dominates the leaderboard of pass rate because it explicitly analyses spatial relationships between blocks and pigs and is the only heuristic agent with the 'prioritizing protective blocks' rule.

The second row in Table 3.1 shows the overall average pass rate for the *broad generalization* evaluation of the agents and humans. The heuristic agents' results were obtained in a similar way as the *local generalization* evaluation, except we only consider the tasks from the testing task templates given to the learning agents. In contrast to the *local*

generalization results, in this evaluation setup, the learning agents have worse results than all the heuristic agents. The D-DDQN-Symbolic and the D-DDQN-Image agents have an average pass rate of 12% and 10% respectively while Relational-Symbolic and Relational-Image have 14% and 9% respectively. The champions in the AIBIRDS competition have an almost double pass rate compared to the learning agents. Our result further advocates the claim that deep learning agents often exploit spurious statistical patterns instead of learning in a meaningful and generalizable way as humans do [44, 13, 27, 124, 107].

Phy-Q score As discussed in Section 3.2.4, the Phy-Q score of humans is set to 100 while the random agent is set to 0. A Phy-Q score above 100 indicates superhuman performance. Figure 3.7 shows the positions of agents and humans in the Phy-Q score distribution. Even though Eagle's Wing was the first in the *broad generalization* leader board (even after removing the results of the first two scenarios Eagle's Wing: 0.11, Bambirds: 0.10), Bambirds has taken the lead in the Phy-Q score dragging Eagle's Wing into the second place. This is because the Phy-Q score positions the agent with respect to human performance.

Interestingly D-DDQN-Image agent and Relational-Symbolic agents have higher Phy-Q values compared to Datalab. Similar to the above reason, this is due to the positioning of the agents with respect to human performance.

Moreover, the Phy-Q of the Pig Shooter is negative. This result is expected as the Pig Shooter only shoots at the pigs, thus having a below-chance performance compared to the random agent. Overall, it can be seen that all the agents are far below the humans' Phy-Q score.

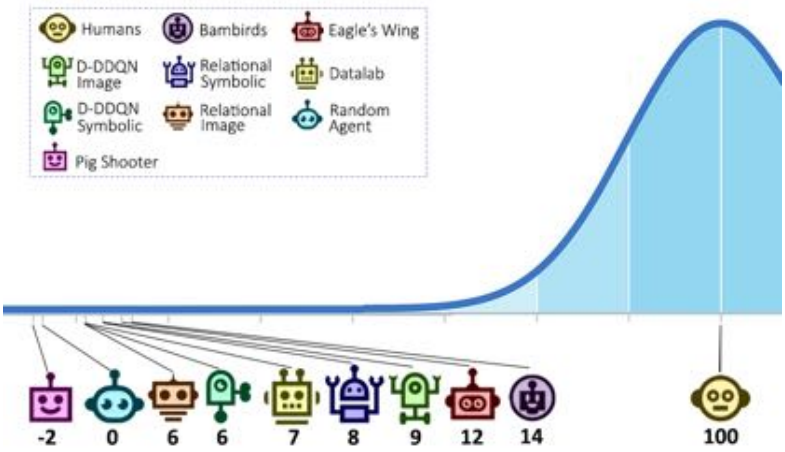


Figure 3.7: Phy-Q Score of humans and the agents

Table 3.2: Comparison of AIBIRDS competition performance with Phy-Q score of heuristic agents

Rank from Phy-Q score	1	2	3	4
Agent Name	Bambirds	Eagle's Wing	Datalab	Pig Shooter
Competition performance	0.27	0.20	0.20	0.00
Phy-Q score	14	12	7	-2

3.4.3 AIBIRDS Competition Performance

Table 3.2 presents the results of the AIBIRDS heuristic agents in the AIBIRDS competition game levels. As can be seen from the results, the pass rate of the agents in competition game levels agrees with the rank they achieved using the Phy-Q score. Eagle's Wing and Datalab have achieved the same pass rate of 0.2. However, considering the total score, Eagle's Wing has obtained 667,394 while Datalab has obtained 634,435 taking Eagle's Wing to the second position. Overall, these results portray that the tasks in the Phy-Q tested are representative of the tasks in the AIBIRDS competition.

From this result, we infer that the physical scenarios available in the Phy-Q tasks are also the scenarios that are commonly encountered in the AIBIRDS competition game levels. The within scenario (*broad generalization*) evaluation we have conducted can be used to identify an agent's ability to perform in those individual physical scenarios. Therefore one can use the Phy-Q testbed to thoroughly analyze the physical reasoning capabilities of their AIBIRDS agent and figure out where the agent falls short and improve on those capabilities. Additionally, the human performance results we have established in the 15 scenarios facilitate to compare of the agents' performance with humans' performance, allowing to set of targets for agents to achieve human-level performance in those scenarios. Thus, the Phy-Q testbed and the evaluation settings we proposed in the testbed can be used to better evaluate AIBIRDS agents and guide them towards achieving human-level performance in the competition.

3.5 Conclusion and Future Work

The goal of Phy-Q is to facilitate the development of physical reasoning AI methods with broad generalizing abilities similar to that of humans. As recent research suggests, humans possess inaccurate forward physics prediction models. Therefore, we focus on tasks that agents can solve by understanding a strategic physical rule instead of solving tasks that require accurate forward prediction. Therefore, towards the goal of developing AI methods similar to humans, we designed 75 task templates considering 15 different physical scenarios in our testbed. The tasks that belong to the same physical scenario can be solved by a specific strategic physical rule. Therefore, it enables us to measure the *broad generalization* of agents by allowing the agent to learn a strategic physical rule in the learning phase that can be used in the testing phase. Apart from the *broad*

generalization performance evaluation, the Phy-Q testbed also enables evaluating agents' *local generalization* performance. We have established baseline results from the testbed and have shown that even though current learning agents can generalize locally, the *broad generalization* ability of these agents is below heuristic agents and is far below human performance. Further, we have defined Phy-Q score, a score to reflect the physical reasoning ability of agents. In addition, we have shown how the testbed can be used towards the advancement of the AIBIRDS competition agents.

Although we discourage developing heuristic agents with hard-coded rules that apply only to Angry Birds, we believe the superior performance of these rule-based systems, given that none of the agent developers has seen the Phy-Q tasks previously, indicates that the human extracted strategic physical rules are highly generalizable. Therefore, we foresee several areas of improvement: 1) agents should learn and store generalizable abstract causal knowledge [107], e.g. strategic physical rules. For example, humans understand not only shooting a bird at a pig can destroy the pig, but also the pig is destroyed because when the bird hits the pig, a force is exerted by the bird onto the pig [96] and if the force is large enough, an object will be destroyed. One possible way to learn this abstract causal knowledge is through Explanation-Based Learning (EBL) [13], where an agent constructs an explanation for initial exemplars and then constructs a candidate rule that depends only on the explanation; if the rule is proven true for a small number of additional exemplars, the rule is adapted. As the representation of abstract and causal knowledge allows for symbolic manipulation [107], 2) it is also worthwhile to explore the possibility of combining deep learning techniques with reasoning over knowledge systems in physical domains, where Neural-Symbolic methods such as NS-DR [186] and NS-CL [105] have shown promising results on physical reasoning.

Phy-Q can be advanced in different directions. Characteristics such as deforming can be introduced to the objects in the tasks. Further, complex scenarios can be added to the testbed by combining the existing scenarios. This will also enable the measuring of the combinatorial generalization of the agents. Moreover, additional physical scenarios that are not covered in the testbed can be introduced such as shape constancy, object permanence, spatiotemporal continuity, and causality. We hope that Phy-Q can provide a foundation for future research on developing AI agents with human-level physical reasoning capabilities, thereby coordinating research efforts towards ever-new goals. Also, more human experiments can be conducted in the future to improve the accuracy of estimating human variabilities.

With Phy-Q establishing itself as a benchmark for assessing AI agents' physical reasoning abilities, the next step is to develop a testbed for open-world learning (OWL) that allows developers to introduce diverse challenges and evaluate agents' performance easily. An ideal OWL testbed should possess simplicity, versatility, precise control, novelty detection capabilities, and a set of baseline agents for standard performance evaluation. With considering these principles, we introduce the Science Birds Novelty framework in the following chapter.

Chapter 4

Science Birds Novelty: An Open-world Learning Test-bed for Physics Domains

This chapter is curated from "Science Birds Novelty: An Open-world Learning Test-bed for Physics Domains" [185].

4.1 Introduction

One of the ultimate goals in the AI field is to have systems that can safely work alongside humans in real-world environments. With the ongoing applications and transitions of AI systems from constrained lab environments to much messier real-world environments, the ability to handle unexpected events (novelties) has taken on new importance in recent years. As a field focusing on developing systems that can operate in such open worlds, open-world learning (OWL) has been proposed [90] recently; A successful open-world system has been defined as one that not only deals with in-distribution inputs, but also rapidly adapts to out-of-distribution inputs.

It is usually not feasible to develop and test OWL systems directly in real-world environments due to the limited opportunities to inject novelties [92, 43]; therefore, simulated test-beds that allow AI agents to learn and enables to evaluate agents are essential to advance the research in OWL [90]. We believe an ideal test-bed for OWL should have 5 characteristics. 1) it should be **simple** enough to allow agents to specifically focus on dealing with novelties in the domain. This ensures agents are not affected by problems that involve other bodies of AI research (e.g. object tracking in crowed scenes). 2) The test-bed should be **versatile** enough. This means the test-bed should enable to inject different kinds of novelty ([90, 31]). 3) The test-bed should be **well-controlled**. That is, in order to systematically evaluate the performance of OWL systems, users of the test-bed should be able to decide precisely what, when, where, and how novelties will appear in the environment. 4) It should **allow agents to report novelty detection**

and characterization, and record them together with the performance of the agents. Although the novelty adaption performance is what we care about, it is still useful to understand if the agent adapts to the novelty because it successfully detects and characterize the novelty or just because of luck. 5) It is desirable for an OWL test-bed to come with **a set of baseline agents that have the expertise or a large enough normal training data-set to support acceptable agent performance in the normal environment**. These baseline agents would help OWL agent developers to get started by studying the baseline agents and they help to evaluate how good OWL agents can adapt to novelties by comparing the performance of OWL agents with the baseline agents.

4.2 Test-bed: *Science Birds Novelty*

After extensively modifying the original Science Birds [57], which is an open-source Angry Birds research clone using Unity, we present our test-bed: the *Science Birds Novelty* [1].

4.2.1 Simplified Inputs

Recent research [25] shows that the physical reasoning performance of AI agents can be significantly affected owing to the errors coming from computer vision components. As *Science Birds Novelty* is a test-bed for physics domains, we encourage agents to develop physical reasoning abilities when interacting with novelties. Therefore, in addition to standard screenshot state representation, we also provide ground truth state representation to avoid the need for computer vision.

A screenshot state representation is a 480 x 640 coloured image and the ground truth representation is in JSON format containing all foreground objects in a screenshot. Each object in the ground truth representation is represented as a polygon of its vertices (provided in order) and its respective colour map containing a list of 8-bit quantized colours that appear in the game object with their respective percentages. An agent can request screenshots and/or ground truth representations of the game level at any time while playing. To further save agents from the object tracking task, we provide an object ID in the ground truth for each object. Hence, an agent can request a batch of ground truth with a desired frequency after a bird has been released and calculate the trajectories of objects of interest without using any advanced computer vision techniques.

4.2.2 Versatile Possible Novelty Types

As an open-source project written in C# using Unity, Science Birds allows us to modify the game in almost any way we want and hence to introduce a large variety of novel situations. This can be as easy as changing physical parameters and colours of existing game objects, to more difficult modifications such as introducing a new class of objects

[1] https://gitlab.com/aibirds/sciencebirdsframework

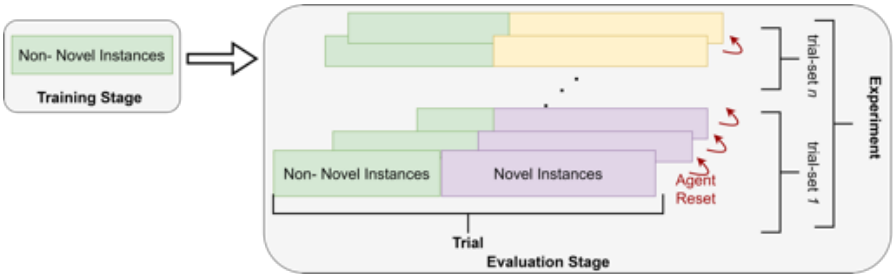


Figure 4.1: Training stage followed by the evaluation stage. An experiment contains *trial-sets* that are drawn from different novel distributions. A trial contains variable lengths of problem instances drawn first from a pre-novelty distribution and then from a post-novelty distribution. We assume the agent is an online learner, and we record its performance throughout a trial. The agent's model is reset at the end of each trial to eliminate model transfer. Reproduced from [131].

(e.g. hostile external agents that hinder the agent) or changing the game goals (e.g. instead of killing the pigs, the goal changes to destroy all wood blocks).

4.2.3 Controlled Novelty Injection

Controlled novelty injection is particularly important when it comes to systematically evaluating novelty detection and reaction performance. For example, being able to decide when a novel situation is appearing in the testing environment helps researchers to measure the timeliness of OWL agents' responses: how long does it take to detect the novelty and how long does it take to adapt to the novelty? Moreover, controlling what, where, and how novelties appear in the environment can help a more detailed evaluation of OWL agents and hence provide clear insights regarding which further improvements are required. For example, an agent might adapt to tasks much faster where the novelty is a reduced mass of objects, rather than to tasks that require the agent to deal with increased friction of objects as novelty. This may suggest that the agent's ability to reason with friction should be improved.

In *Science Birds Novelty*, users can specify the exact type and location of novelty to be included and decide the exact order when the novel game level will appear in a trial (discussed in section 4.2.7).

4.2.4 Baseline Agents

Together with the test-bed, we include 3 heuristic-based agents: Eagle Wings, Datalab, Naive Agent, and 1 deep-learning-based agent, the DQ-Birds.

- **Eagle Wings:** This is the winner of 2017 and 2018 competitions. The agent selects action based on strategies including shooting at pigs, destroying most blocks,

shooting high round objects, and destroying structures [171].

- **Datalab:** Datalab is the winner of the 2014 and 2015 competitions. The agent uses the following strategies: destroy pigs, destroy physical structures, and shoot at round blocks. The agent selects the strategy based on the game states, possible trajectories, bird types, and the remaining birds [30].
- **Naive Agent:** The strategy of the Naive Agent is to directly shoot at the pigs. The agent shoots the bird on the slingshot by randomly selecting a pig and a trajectory to shoot the pig [159].
- **DQ-Birds**: A Double Dueling Deep Q-network trained on over 115,000 Angry Birds game frames using greedy epsilon and partially random policy [126].

4.2.5 Integrating Game Level Generation

To help agents develop an acceptable level of performance in a normal game environment, we integrated the winner of the 2017 and 2018 Angry Birds Level Generation Competitions, Iratus Aves [158], into our test-bed to make it possible to create an unlimited number of new game levels for training. Other procedural game-level generators can be used as well.

4.2.6 Illustrative Examples of Novelties

In this section, we present a comprehensive collection of 12 sample novelties, each categorized within the first four levels of the Open World Novelty Hierarchy [148].

Novelty Level 0, referred to as "Instance Novelty," involves previously unseen instances, which can be likened to entirely new game levels, unencountered before in the gaming environment.

Moving on to Novelty Level 1, known as "Class Novelty," novelties in this level manifest as previously unencountered classes of objects or entities. These novelties introduce unexplored game elements with distinct effects, including novel types of blocks that exhibit different behaviours compared to the familiar block classes. These newly introduced game objects can be easily distinguished from pre-existing objects based on their visual appearance, but their behaviour remains a mystery at first glance. Within the class novelty level, we have devised five sample novelties, which include: 1) a new block type with a significant increase in linear drag (from 1 to 25), while maintaining other parameters identical to wood blocks, 2) a new block type with double the score compared to wood blocks, with all other parameters mirroring those of wood blocks, 3) a new block type with three times the health points of ice blocks, retaining other parameters identical to ice blocks, 4) a new bird type that resembles dark pigs while maintaining the characteristics of red birds, and 5) a pig with a novel appearance, while all other parameters remain consistent with small pigs.

Novelty Level 2, known as "Attribute Novelty," refers to modifications made to the

attributes of objects or entities, such as changes in colour, shape, or orientation, which were previously not relevant for classification or action. Many of these novelties may not be visually apparent but significantly alter gameplay dynamics. In this category, we introduce five sample novelties, including 1) a change in the red bird's bounciness from 0.3 to 0.9, 2) a modification of the red bird's linear drag from 0 to 0.2, 3) a threefold increase in the health points of wood blocks, 4) a doubling of the score associated with wood blocks, and 5) an adjustment in the linear drag of wood blocks from 1 to 25.

The final novelty level, Novelty Level 3, is referred to as "Representation Novelty." This level primarily encompasses alterations in how entities or features are specified, entailing transformations of dimensions or coordinate systems. For this level, we have devised two sample novelties, which include: 1) the transformation of colours in the screenshot and ground truth representations to grayscale, and 2) a rotation of the screenshot and ground truth by 180 degrees around the centre point (420, 240) of the image.

4.2.7 Evaluation Protocol

In *Science Birds Novelty*, we follow the same evaluation protocols (Fig. 4.1) that is described in [131].

1. Agents are first exposed to a sequence of normal (i.e., no novelties presented) game levels. The number of normal game levels is not known to the agent. Agents can attempt to solve each game level only once in the given order.
2. At some point, the novelty occurs, and all subsequent game levels after that point include a certain type of novelties. The number of novel game levels is unknown to the agent. Agents can attempt to solve the game level only once in the given order as well.
3. For every instance (normal or novel) i an agent attempts to solve, we record its task performance (e.g., score) TSP_i and p_i, the probability that the agent believes novelty has occurred. Task performance reflects how well the agent solves a game level.

We refer to the above sequence of normal and novel game levels as a single *trial*. T_j^a is the j^{th} *trial* for a given novelty a. We refer to a set of trials with the same post-novelty distribution as a *trial-set*. An *experiment* is a set of trial-sets. When an agent completes a trial, it is reset to its initial state before it begins the next trial (i.e., agents are permitted to learn throughout a trial, but learned models are not transferred between trials. The agent also reports a *detection threshold* where each p_i exceeding the threshold indicates a predicted distribution change (i.e., the agent predicts that a novelty has occurred).

4.3 Use-case: AIBIRDS Novelty Track

In this section, we provide a demonstration of how our test-bed can be used to evaluate OWL agents' performance with the AIBIRDS Competition Novelty Track.

Table 4.1: Novelties in the AIBIRDS 2021 Novelty Track

1. Novelty Level 1: (Class)	1.1. New egg-shaped object which gives $-10,000$ points when hit. 1.2. New bird with low friction and low bounciness that can slide on the ground.
2. Novelty Level 2: (Attributes)	2.1. Pig color changed to red. 2.2. Launch force of red bird increased.
3. Novelty Level 3: (Representation)	3.1. The game is flipped upside down, agents need to shoot downwards. 3.2. Changed color map from RGB to BGR, a different color.

During the AIBIRDS competition at IJCAI 2021, we introduced a new element called the **AIBIRDS Novelty Track** for the very first time. This track focused on the four novelty levels discussed in section 3 and introduced a fresh set of evaluation novelties for each of these levels (refer to Table 1). These novelties were kept secret from the participants. For each novelty introduced, we created game levels, each containing that specific novelty. It's important to note that a single game level could include multiple instances of the same novel object.

The competition's evaluation involved multiple trials, each designated as T_i^a, for a certain level of novelty, "a," and followed the protocol outlined in section 3.7. Each trial consisted of a sequence of "n" distinct Angry Birds games, with the first "m_i" games being pre-novelty games, or games without any novelties, and the remaining "$n - m_i$" games introducing novelties. The exact values of "n" and "m_i" were unknown to the participants, and "m_i" could range from 0 to "n". If "m_i" was 0, the trial could only consist of games without novelties. If "m_i" was greater than 0, the trial could consist of either only novel games or a sequence of standard games followed by novel games. The games within a trial could only be played once and had to be played in the specified order, with each trial having a time limit.

In each trial, the agents were required to indicate if they thought the novelty switch had happened or not by providing a value between 0 and 1. If the value was greater than 0.5, it meant that the trial had switched to novel games. We recorded the solved score achieved by the agents for each game. The solved score for a game was the same as the game score if all pigs were eliminated; otherwise, it was 0.

We measured several aspects for each agent:

- For each level of novelty, we calculated the aggregated solved score, which served as the task performance measure. This score was the sum of the solved scores for each game that contained novelty, or for all games for novelty level 0.
- We used a metric called "percentage of correctly detected trials" (%CDT) to determine the novelty levels 1-3. This metric evaluated trials where the agent detected the introduction of novelty for a new game correctly (with no false positives and at least one true positive if new games were present).

- We also computed the "average number of novel games needed" (abbreviated as #NGN) for each novelty level. This calculation was based only on the trials in which novelty was correctly detected for novel games. The #NGN represents the average number of novel games played before detecting novelty in a trial.

We identified two agents for each novelty level: the agent with the highest aggregated solved score and the agent with the highest novelty detection score. The novelty detection score was calculated as a product of the percentage of correctly detected novelties and the difference between the maximum number of novelties and the actual number of novelties detected, averaged across all trials where novelty was detected in the last game.

The winner of the competition was the agent who had the highest total score across all novelty levels 1-3. There were also subcategories for the best-performing agent at each of the four novelty levels. Furthermore, there was a special award for the agent with the best novelty detection score across all novelty levels 1-3 and subcategories for each of the three novelty levels.

The primary task for the agents remained the same: to solve each game level by eliminating all the pigs using as few birds as possible. However, the introduction of novelty could significantly alter the strategies needed to accomplish this task. Agents needed to adapt and detect the novelty to continue solving games. Each trial comprised between 0 and 10 non-novel games, followed by precisely 40 novel games. Participants were not informed about these settings. Agents were evaluated based on two key aspects: (1) their ability to detect novelty, which considered the percentage of trials where they correctly detected novelty and the number of novel games they needed to detect it, and (2) their performance in reacting to novelty, which was assessed by the overall game score they achieved in the novel games. Given the use of six novelties, ten trials per novelty, plus ten trials with no novelty, and each trial consisting of around 50 games, each agent had to play approximately 3500 games in total.

4.3.1 Competition Results

Six teams participated in this highly challenging competition. These teams included the following:

- **BamBirds** from the University of Bamberg, who had previously won the standard track in the 2019 competition.
- **CIMARRON** from the University of Massachusetts Amherst.
- **Dongqing 1**, a collaborative team from Bytedance and Monash University.
- **HYDRA** from the Palo Alto Research Center and the University of Pennsylvania.
- **OpenMIND** from Smart Information Flow Technologies.
- **Shiro** from NIAD-QE.

Table 4.2: AIBIRDS 2021 Novelty Track Overall Results

Agent	#NGN	%CDT	Detection Score	Total Solved Score
BamBirds	1	6%	2.44	36192880
CIMARRON	3.23	67%	24.52	**49653730**
Dongqing 1	3.13	67%	24.58	41231190
HYDRA	9.86	83%	25.12	37766360
OpenMIND	1.49	68%	**26.32**	28427980
Shiro	1	5%	1.95	19488940

Table 4.3: AIBIRDS 2021 Novelty Track Detection and Reaction Results by Novelty Level. Detection performance is followed by the reaction performance after the comma ",".

Agent	Level 1	Level 2	Level 3
BamBirds	1.95 , **25265200**	2.79 , 6563240	2.79 , 4364440
CIMARRON	**36.40** , 24020150	18.25 , **16386620**	18.90 , **9246960**
Dongqing 1	21.00 , 24139240	**34.75** , 9674290	18.00 , 7417660
HYDRA	21.45 , 19042980	27.05 , 10504580	26.85 , 8218800
OpenMIND	19.5 , 17651480	20.45 , 6919110	**39.00** , 3857390
Shiro	1.95 , 13247470	1.95 , 4335210	1.95 , 1906260

In Table 2, the results indicate that **CIMARRON** achieved the best performance in reacting to novelties. **Dongqing 1** secured the second position, followed by **HYDRA** in the third place. For the best novelty detection performance, **OpenMIND** emerged as the leading agent in this competition.

It is also interesting to notice that **CIMARRON** dominates the novelty reaction performance over all three novelty levels achieving 3rd place in level 1 and 1st place in levels 2 and 3, while there's no dominating winner in novelty detection (Table 3). Different agents were good at detecting each different novelty level. For example, **CIMARRON** is much better than others in detecting level 1 novelties while **Dongqing 1** and **OpenMIND** are better in levels 2 and 3.

4.4 Conculsion

In conclusion,as AI applications increasingly transition from controlled settings to real-world scenarios, the ability to handle unforeseen events, or novelties, becomes paramount. Open-world learning (OWL) has emerged as a framework to address this challenge, emphasizing systems that not only perform well on familiar data but also rapidly adapt to novel inputs.

However, testing OWL systems directly in real-world environments is often impractical due to limited opportunities to introduce novelties. Therefore, simulated test-beds play a crucial role in advancing OWL research.

In response to these needs, we introduced the Science Birds Novelty test-bed based on the Angry Birds domain. This test-bed facilitates research by providing a controlled

environment for assessing AI agents' ability to adapt to unforeseen circumstances. Additionally, we presented the AIBIRDS Competition Novelty Track as a use-case of our test-bed, showcasing its effectiveness in evaluating OWL systems.

Building upon the Science Birds Novelty framework and Phy-Q (Chapter 3), there arises a critical need for a robust benchmarking system. Such a system must enable the systematic introduction and evaluation of AI agents on a wide array of novelties, while also providing clear metrics for assessing performance in tasks involving these novelties. This poses a significant challenge, as it is important to distinguish whether an agent's success in handling novel scenarios arises from a genuine understanding of adaptation or mere chance.

To address these questions, we introduce the NovPhy open-world learning benchmark in Chapter 5. This benchmark aims to provide a comprehensive framework for evaluating AI agents' ability to adapt to novel situations effectively in physical domains. By defining standardized procedures for introducing novelties and measuring performance, NovPhy offers a valuable tool for advancing research in open-world learning and enhancing our understanding of AI systems' capabilities in real-world environments.

Chapter 5

NovPhy: A Testbed for Physical Reasoning in Open-world Environments

This chapter is curated from "NovPhy: A Testbed for Physical Reasoning in Open-world Environments" [60]. As one of the first authors, I proposed a novel framework for assessing novelty adaptation agents in diverse physical scenarios, aiming to accurately isolate and gauge the genuine extent of their adaptation capabilities. Collaboratively with Chathura and Vimu, I further honed the evaluation design. Taking a hands-on approach, I implemented the server-game-agent testing framework and introduced an innovative adaptation agent, the Naive Adapt agent, for comprehensive experimentation. My involvement extended to conducting experiments for the baseline agents and crafting the corresponding experiment sections. Additionally, I meticulously proofread various sections, providing extensive comments for refinement.

5.1 Introduction

A key aspect of human intelligence is the ability to reason about the physical behaviour of objects and make decisions in the physical environment [47]. Research suggests that humans develop physical reasoning capabilities within just the first year of birth [12, 17]. Even though it has been challenging to develop AI systems that can do general physical reasoning as good as humans do [44, 184], there is work that shows AI systems that could achieve human performance in some physical reasoning tasks [4, 25]. But the question is: is it adequate to only have physical reasoning capabilities to successfully work in the real physical world?

Encountering novel situations is an inherent characteristic of the real world (Figure 5.1). As humans, we are adept at working in such novel situations that we constantly face in our day-to-day lives. For example, consider a person who can ride a bicycle on a normal road. On a rainy day when the roads are slick, the person can still ride the bicycle

safely by adjusting the speed and applying the brakes smoothly without slipping. Even though sometimes a human may initially make a suboptimal decision when facing a novel situation, they usually quickly recover successfully after continuing to work for a short period under the influence of novelty. Similarly, for an agent that operates in an open-world physical environment, along with physical reasoning capabilities, it is crucial to possess capabilities that are required to handle novel situations [189], i.e., novelty detection and adaptation capabilities.

Figure 5.1: Example novelties that could be encountered in the real world. Left: self-checkout machines started appearing in supermarkets after the early 2000s, until then customers were used to traditional checkout methods, hence using self-checkout machines was a novelty to the customers [179]. Right: traffic accidents are generally novel situations for self-driving cars as such incidents are rare in the training data and usually visually unique in each incident [178].

In contrast to the intelligence of humans, current AI systems tend to struggle when they are presented with situations that were not available during their training stage or if the situation was not anticipated by the developers [66]. This could be due to the fact that the research field, Open World Learning (OWL), attempting to address this issue is relatively new [148, 90, 80]. Apart from that, not having adequate testbeds to experiment and evaluate such AI systems also hinders their advancement. There are frameworks such as Monopoly [80], Polycraft [54], Cartpole [35], etc, that treat novelties as first-class citizens and facilitate agent experimentation. Even though some of them are physics based environments [35], none of them specifically focus on introducing novelties to the physical scenarios that an agent would encounter in the real world. Also, it is out of their context to evaluate agents in real-world physical tasks in the presence of novelties.

To fill the above gaps, we propose a new novelty-centric testbed, NovPhy, where agents need to perform in real-world physical scenarios in the presence of novelties. NovPhy includes a wide variety of novelties applied to different physical scenarios. We implement

our testbed on the physics-based video game Angry Birds as it has realistic physics and it is a versatile domain to introduce physics-based novelties. Moreover, Angry Birds is popular in both physical reasoning research [184, 139, 140] and OWL research [185, 59, 132]. The main contributions of this work can be summarized as follows:

- **NovPhy - A testbed for novelty detection and adaptation in physical environments:** We consider five commonly encountered physical scenarios in NovPhy: applying a single force and multiple forces, and rolling, falling, and sliding objects. We developed eight novelties representing a diverse novelty space. We designed task templates by applying the eight novelties to the five physical scenarios separately, resulting in 40 novel task templates. A task template is used to generate related tasks by varying task template parameters such as the locations of the objects. We also created 40 corresponding normal task templates without novelties to facilitate our evaluation protocol. Further, we developed a task variation generator that can generate an unlimited amount of tasks from these task templates.
- **Agent evaluation setups for open-world physical environments:** We propose a comprehensive evaluation setup to evaluate the novelty detection and adaptation of AI systems in open-world physical environments. In this setup the novelties are orthogonal to the physical scenarios, hence facilitating us to evaluate agents in two settings: first, the same novelty is applied separately to the tasks of multiple physical scenarios, and second, multiple novelties are applied separately to the tasks of the same physical scenario. The former is used to evaluate how well an agent can deal with the same novelty in different physical reasoning tasks and the latter is used to evaluate how well an agent can perform the same physical reasoning task under different novel situations.
- **Establishing results for baseline agents:** We evaluate 11 baseline agents. Three heuristic-based agents, two standard online learning agents, two standard offline agents, three adaptive learning agents, and a random agent. We report their novelty detection and novelty adaptation performance.
- **Establishing baseline human performance:** In order to show our novelties are adaptable for humans, we conducted an experiment using human players in our testbed. These results also show that humans can detect and adapt to novelties better and faster compared to AI agents, thus acting as a milestone performance for AI to achieve.

5.2 Designing Novel Tasks and Agent Evaluations in Open-world Physical Environments

In this section, we discuss the desiderata we satisfied when designing tasks in our testbed and when setting up agent evaluations for those tasks. We term the tasks that have novelties in them as novel tasks and the tasks without novelties (i.e., tasks in the normal

environment) as normal tasks. In our testbed, we consider a constrained environment setting where an agent's action only applies to the state of the environment when the action is taken, and that action determines the subsequent states of the environment (i.e., the agent cannot control the subsequent states of the environment after the action is taken).

5.2.1 Designing Novel Tasks

An agent working in a physical environment has to encounter different physical scenarios such as applying a force to an object, moving an object from one place to another, avoiding an obstacle in its path, etc. We introduce novelties to these scenarios that an agent could encounter in the physical environment.

When designing novel tasks, **we ensure that the agent has to work under the effects of the novelty to solve the task**. In other words, in the novel tasks, there are no solutions to the task that skip the effects of the novelty. For example, consider a bowling game where the player has to roll a ball on a surface to knock over the pins. Assume that when novelty occurs, the surface on which the ball rolls becomes slippery. In this scenario, the player is required to interact with the novel element (the surface) in order to complete the task. Therefore, the only way to successfully perform this task is to adapt to roll the ball on the slippery surface.

To ensure that the agent has to go through the novelty when completing a task, when designing novelties we consider the physical interactions in the solution of a physical scenario. We categorize these physical interactions into three phases: the initial phase, the middle phase, and the final phase. We only design novelties that at least affect one of these three interaction phases, to guarantee that the agent has to work along with the effects of the novelty. The initial phase includes the immediate impact on the objects by the agent's action, the middle phase includes the consequences of the immediate impact of the action, and the final phase includes the interactions that complete the task. In all the phases, the objects that are involved in those physical interactions are also considered.

For example, consider the previously mentioned bowling game. In this scenario, the possible physical interactions are, the player throws the ball giving a starting velocity to the ball, the ball rolls on the surface, and the ball hits the pins knocking them down. In this instance, the initial interaction phase includes the ball and the velocity the player applies to the ball. The middle phase includes the ball and the surface, and the rolling movement of the ball. The final phase includes the ball and the pins, and the collision between the ball and the pins. Therefore, here, the novelty can be applied to the ball, to the surface, to the pins, or to something that affects the rolling of the ball and collision of the ball and pins, in order to make sure the agent has to bowl under the effect of the novelty.

5.2.2 Designing Agent Performance Evaluations

In OWL, agent evaluations are conducted to measure two capabilities of agents: the novelty detection capability and the novelty adaptation capability [131, 73, 129]. The novelty detection measures evaluate whether an agent could successfully detect a novelty in the environment and the novelty adaptation measures evaluate whether an agent could successfully perform the task in the presence of the novelty. Novelty adaptation is generally more emphasized than novelty detection, as performing the task under the influence of a novelty is more important than merely detecting something novel for an agent that actually works in an open-world environment. In this work, we also prioritize evaluating agents' novelty adaptation performance.

The standard agent evaluation setup in OWL consists of a set of trials, where each trial consists of a sequence of normal tasks followed by a sequence of novel tasks [131, 129, 120]. We follow the same setup in NovPhy. We believe that, in order to measure whether an agent genuinely adapts to a novelty, **there should be a change in the solution path of the tasks when moving from the normal tasks to the novel tasks**. In an abstract form, we define a solution path as a sequence of physical interactions including the associated objects, initiated by an agent's action, that leads to solving the task. When designing a novel task for a physical scenario we also design a corresponding normal task that has a different solution path compared to the novel task. Then, when defining the trials for the evaluation, we select these normal and novel task pairs to guarantee that there is a solution path change from normal to novel tasks.

In this setting, since there is an obvious change in the solution path from the normal tasks to the novel tasks, novelty detection becomes trivial. To detect whether there is a novelty, the agent has to simply monitor whether the solution in the normal tasks is no longer working. To avoid this consequence, one could define separate trials that are only used to evaluate the novelty detection performance by including the tasks that have the same solution path in both normal and novel tasks. In this work, we do not include such trials in the evaluation as our main focus is evaluating the novelty adaptation performance of the agents.

We consider another desideratum when evaluating the performance of an agent in an open-world physical environment. From the perspective of OWL, we believe that, **if an agent can truly perform under a novelty, the agent should be able to perform with that novelty when the novelty is applied to different physical scenarios**. Also, from the perspective of physical reasoning, we believe that, **if an agent is robust at performing in a physical scenario, that agent should be able to perform in that scenario under the effect of different novelties**. To achieve these two evaluation setups, we designed the novelties orthogonally to the physical scenarios, such that the same novelty can be applied to multiple scenarios and the same scenario can be affected by multiple novelties.

5.3 NovPhy Testbed

In this section, we introduce our testbed, the physical scenarios we consider, the novelties we designed, the tasks in the testbed, and explain the evaluation settings we have used in the testbed.

5.3.1 Introduction to NovPhy

Based on different physical scenarios and novelties, we develop the novelty-centric testbed NovPhy using Angry Birds. We use an open-source research clone of the game developed in Unity called Science Birds [57]. Our testbed is adapted from a framework that can be used to inject novelties and conduct agent evaluations, developed from Science Birds [185].

In Angry Birds, the goal of the player is to kill all the pigs in the game level by shooting a given number of birds from a slingshot. In the normal game environment, along with the slingshot, the player will encounter four types of game objects: birds, pigs, blocks, and platforms. Additionally, we have also introduced an external agent to the normal environment called Air Turbulence that applies an upward force to any object that travels through it. An external agent is an agent with goal-oriented behaviour and having external agents enables situations that hinder or support the action that a player takes. In the game, birds, pigs, and blocks are dynamic objects, which behave according to Newtonian physics, while platforms are static and are not affected by external forces. The dynamic objects have health points that get reduced in the collisions and they get destroyed when the health points become zero. The blocks have 12 variations in shape and they are made of one of 3 types of materials: wood, stone, and ice. There are three types of pigs with three different sizes; the larger the size the higher the health points are. All objects in NovPhy are shown in Appendix Figure 5.10.

An agent playing the game can request the current game state anytime as a screenshot and/or as a symbolic representation. The screenshot is a 480x640 coloured image of the game. The symbolic representation is in JSON format and contains all the objects in the screenshot. Here, an object is represented as a polygon of ordered vertices along with the percentages of its 8-bit quantized colours. The full world state is not provided to the agent such as the exact positions of the objects and their physical parameters such as mass, coefficient of friction, etc. as they are not directly observable in the real world. The action of an agent is the release point of the bird relative to the slingshot. Sometimes when there is more than one bird in the game level the agent takes a sequence of actions. We provide a trajectory planner that can be used to calculate the release point of the bird to reach a target, under the normal settings in the environment, when the target point is given. The agent passes the game level if it destroys all the pigs with the provided number of birds or fails if not.

The physical scenarios and the novelties we consider in this testbed are discussed in the next subsections 5.3.2 and 5.3.3 respectively.

5.3.2 Physical Scenarios in NovPhy

We use the first five physical scenarios introduced in Phy-Q as they are the most basic and frequently encountered scenarios in a physical environment. The scenarios include applying forces directly on target objects - the effect of a single force and the effect of multiple forces. The motion-related scenarios: rolling, falling, and sliding, inspired by the physical reasoning capabilities developed in human infancy [29]. The five scenarios and the corresponding physical rules that can be used to achieve the goal of the associated tasks are:

1. Single force: Target objects have to be destroyed with a single force.
2. Multiple forces: Target objects have to be destroyed with multiple forces.
3. Rolling: Circular objects have to be rolled along a surface to a target.
4. Falling: Objects have to fall onto a target.
5. Sliding: Non-circular objects have to be slid along a surface to a target.

5.3.3 Novelties used in NovPhy

We design a representative novelty for each hierarchy level in the open-world novelty hierarchy proposed by the SAIL-ON program novelty working group. The novelty hierarchy consists of eight novelty levels that cover a wide range of novelty types that could occur in an open-world environment. Table 5.1 shows the open-world novelty hierarchy and descriptions of representative novelties in NovPhy. Appendix Figure 5.11 shows the new game objects that are introduced to the game for the novelties associated with a game object.

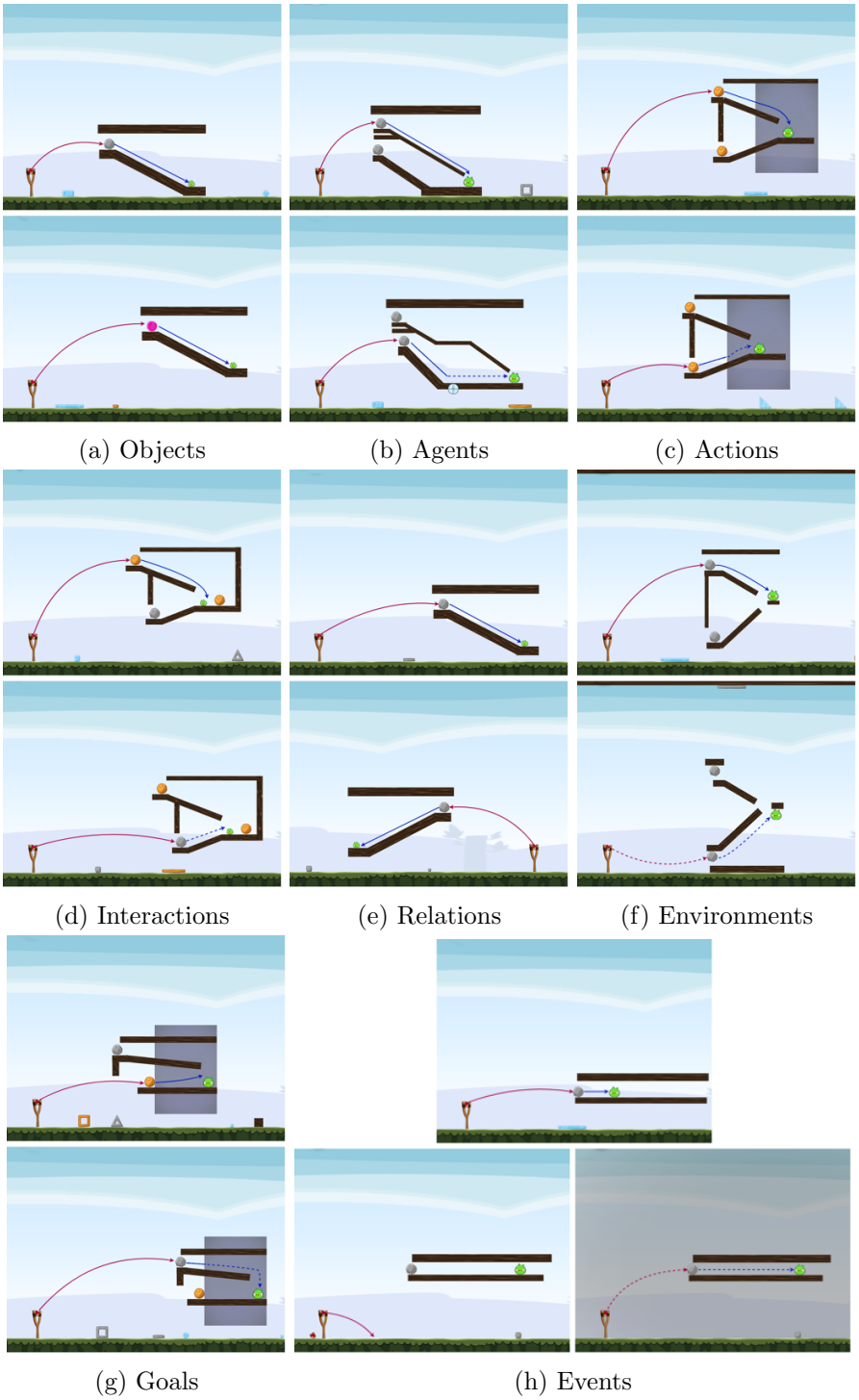
5.3.4 Task Templates

A task template defines a set of related tasks that can be created by varying task template parameters such as the locations of the objects. We design task templates by applying each of the eight novelties to each of the five physical scenarios discussed in the above sections. For example, for the rolling scenario we design templates that require rolling an object when: it is a new object, there is an effect from the Fan, the Air Turbulence agent changes the magnitude of the upward force, the slingshot is in the right side of the object, there is an effect from a magnetic field, the gravity is inverted, Air Turbulence pushes the object down, and there is a storm after shooting the first bird. We term a physical scenario with a novelty applied as a novelty-scenario. Since we consider eight novelties and five scenarios, we have 40 novelty-scenarios.

When designing novel task templates for NovPhy, we follow the desiderata discussed in Section 5.2. For each novelty-scenario, we designed and handcrafted a novel task template. A novel task template is a task template where a novelty is present. In the

Table 5.1: SAIL-ON Open-world novelty hierarchy [50] and the representative novelties in NovPhy for each hierarchy level.

Novelty Level	Description	Representative Novelty
1. Objects	New classes, attributes, or representations of non-volitional entities.	A new pig/block that has a different colour to the normal pigs/blocks.
2. Agents	New classes, attributes, or representations of volitional entities.	A novel external agent, Fan, that blows air (horizontally from left to right) affecting the moving path of objects.
3. Actions	New classes, attributes, or representations of external agent behaviour.	The non-novel external agent, Air Turbulence, increases the magnitude of its upward force.
4. Interactions	New classes, attributes, or representations of dynamic properties of behaviours impacting multiple entities.	Existing circular wood object now has magnetic properties: repels objects of its type and attracts other object types.
5. Relations	New classes, attributes, or representations of static properties of the relationships between multiple entities.	The slingshot which is at the left side of the tasks is now at the right side of the tasks (i.e., the spatial relationship between the slingshot and other objects is changed).
6. Environments	New classes, attributes, or representations of elements independent of specific entities.	The gravity in the environment is now inverted, which affects the behaviour of the dynamic objects.
7. Goals	New classes, attributes, or representations of external agent objectives.	The non-novel external agent, Air Turbulence, changes its goal from pushing objects up to pushing objects down.
8. Events	New classes, attributes, or representations of series of state changes.	When the first bird is dead, a storm occurs that affects the motion of the objects (by applying a force to the right direction).



(a) Objects (b) Agents (c) Actions

(d) Interactions (e) Relations (f) Environments

(g) Goals (h) Events

Figure 5.2: Example tasks of the rolling scenario for the eight novelties. In each subfigure, the top figure is the normal task and the bottom figure is the corresponding task with the novelty. The arrows show the trajectories of the objects when the solution is executed.

design, we ensure the novel tasks meet our desideratum, that it is necessary to work under the effects of the novelty to complete the task. Appendix Table 5.4 contains more details on how this desideratum is satisfied when designing the tasks, considering the physical interaction phases of the solution that are affected when the novelty is introduced. Then, for each novel task template, we design a corresponding normal task template as well. This is done by removing the novelty from the novel task template and adjusting the template accordingly to make it solvable without the novelty. Considering our next desideratum, when designing the normal task templates, we also ensure that there is a solution path change from the normal task to the novel task.

In the task template design, we also ensure that all the templates of a given scenario can be solved by the associated physical rule of that scenario discussed in Section 5.3.2. Figure 5.2 shows how the eight novelties are applied to the tasks of the rolling scenario. In each subfigure, the top figure is the normal task and the bottom figure is the corresponding task with the novelty. The arrows show the trajectories of the objects when the solution is executed (in red: bird's trajectory, in blue: other objects' trajectories when the bird is hit). Dotted arrows represent the trajectories affected by the novelty. To solve the novel task shown in each subfigure:

(a) objects: the new pink coloured object has to be rolled.

(b) agents: the force of the new Fan agent has to be used to roll the object further.

(c) actions: the magnitude of the upward force of the Air Turbulence agent is increased, which helps to roll the ball upwards in the ramp.

(d) interactions: the circular wood (brown) objects have magnetic properties, thus repels the object of the same type and attracts the stone (grey) object helping to roll the stone upwards in the ramp.

(e) relations: the slingshot is now placed in the right side of the task, instead of shooting left to right now the shooting has to be done from right to left to roll the object to the left direction.

(f) environments: the inverted gravity makes the objects attract towards the sky, hence objects can be rolled upwards in ramps.

(g) goals: the goal changed Air Turbulence agent (from the goal of pushing objects up to pushing objects down) hinders rolling on flat surfaces while helps rolling on inclined surfaces.

(h) events (the novel template has two birds as shown in the first bottom figure and when the first bird dies it activates the storm as shown in the second bottom figure): when the first bird is wasted, the storm occurs which applies a force to the right direction of the moving objects, hence shooting the second bird to the circular object makes the object to roll further to reach the pig.

Figure 5.3 shows how the inverted gravity novelty is applied across the tasks of the five

physical scenarios. All 40 task templates in NovPhy can be found in 5.7.2.

5.3.5 Task Generation

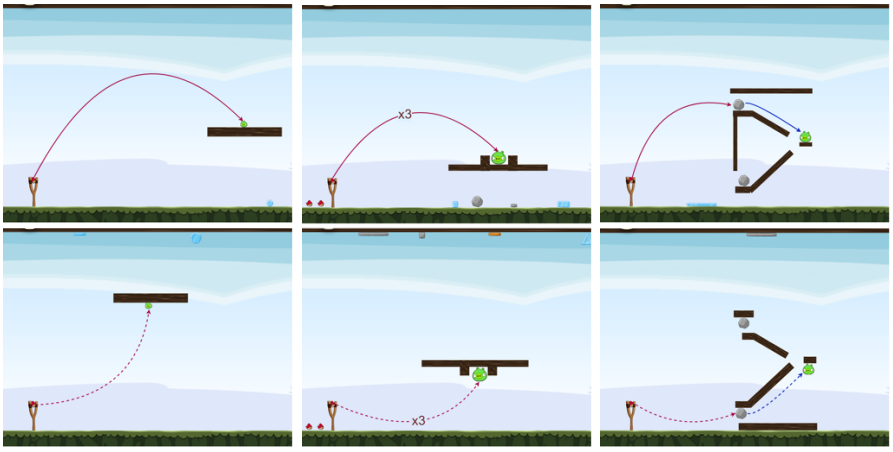
We developed a task generator that can generate an unlimited number of tasks from a given template. The game levels generated from a task template are termed as the tasks of that template. When generating the tasks we vary the locations of the game objects within a suitable range in the level space. Additionally, some random game objects are added as distraction objects at random positions of the game level to trick the agents. In the generation, we ensure that the task can still be solved by the solution path in the original template. To achieve this, we define template specific constraints such as, which game objects are reachable/unreachable to the bird, which objects should be excluded from the distraction objects, what are the feasible regions to place specific objects, etc. These constraints are determined by the template designers and are input to the task generator.

We provide 350 generated tasks for each task template, but we also provide the task generator in case it is necessary to generate more tasks. Appendix Figure 5.17 shows the variations of the tasks generated from the rolling scenario template with the inverse gravity novelty applied.

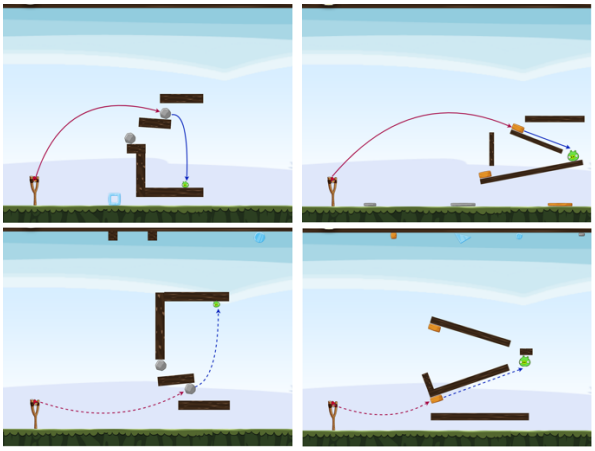
5.3.6 Evaluation Protocol

In NovPhy testbed, as in a standard OWL evaluation, we evaluate the novelty detection and novelty adaptation capabilities of the agents. In the novelty detection evaluation, we measure if an agent can detect if a novelty is present in the given task. In novelty adaptation evaluation, we measure the task performance of the agent in the presence of a novelty. Both novelty detection and novelty adaptation evaluation are done by using a trial setting [131]. A trial is a sequence of tasks, which starts from normal tasks and after a random number of normal tasks switches to novel tasks. After switching to novel tasks, all the subsequent tasks until the end of the trial are novel tasks. Figure 5.4 shows how evaluations are done through the trial setup. A trial-set is a set of trials. A given trial set consists of trials of the same novelty-scenario. i.e., all the trials of a given trial-set only have normal and novel tasks from the same novelty-scenario. The agent is not allowed to share knowledge in between trials, i.e., at the start of each trial of a novelty-scenario, the agent is in the same initial state, as the agent was at the beginning of the evaluation.

To evaluate an agent on a novelty-scenario, the agent is trained on the normal task distribution of that novelty-scenario and tested using a trial-set of that novelty-scenario. This evaluation resembles the local generalization evaluation of the agents, i.e., the agent trains on the tasks of a normal template and is tested on the tasks of the same template and the corresponding novel template. Even though the testbed facilitates broad generalization evaluations (i.e., the agent trains on the tasks of a normal template of a physical scenario. Then the agent is tested on the tasks of a different template of



(a) Single force (b) Multiple forces (c) Rolling



(d) Falling (e) Sliding

Figure 5.3: Example tasks of the inverted gravity novelty applied to the five physical scenarios. In each scenario, the top figure is the normal task and the bottom figure is the novel task. The arrows show the trajectories of the objects when the solution is executed. Solid arrows are the trajectories of the objects that are not affected by the novelty and the dotted arrows are the trajectories of the objects that are affected by the novelty. The inverted gravity has made all the dynamic objects attract towards the sky, hence they have been stopped from platforms to avoid flying away. The motion of the dynamic objects is also affected by the inverted gravity.

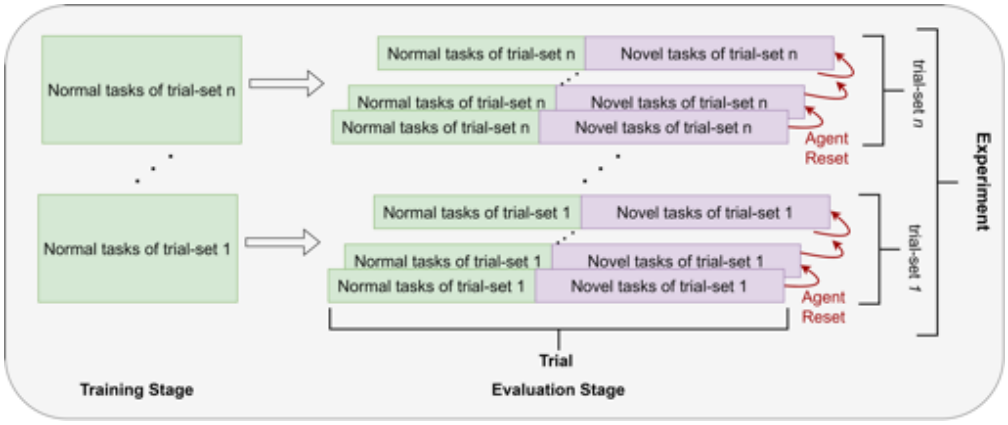


Figure 5.4: The trial-based evaluation protocol used in NovPhy. The evaluation stage follows the training stage. An experiment contains *trial-sets* where a trial-set contains multiple trials. A trial contains a variable number of tasks drawn first from the normal task distribution and then from the novel task distribution. In a given trial-set the agent is only evaluated on a single novelty-scenario.

the same scenario and its corresponding novel template), we use this local generalization based evaluation setup as it is proven that learning agents still struggle to generalize broadly in physical reasoning [184]. Moreover, it is a precondition for agents to have a good normal task performance before adapting to novel tasks.

When the agent is playing a trial, the task completion status (whether the task is passed/failed) and the score the agent achieved at the end of the task are recorded. This data is used to calculate the novelty adaptation performance of the agent. For the novelty detection performance calculation, the agent has to inform in which task of the trial it believes the novelty occurred.

In this work, we focus on evaluating agents in the below two evaluation settings.

1. Novelty Informed Evaluation: In this evaluation, an agent will be informed when the novelty appears in each trial. The agent will only be evaluated on the novelty adaptation ability.

2. Novelty Uninformed Evaluation: In this evaluation, an agent will be evaluated on both novelty detection and novelty adaptation. The agent will not be informed when the novelty appears in a given trial.

5.3.7 Evaluation Measures

For novelty detection, we use standard OWL measures used in the SAIL-ON program: the percentage of correctly detected trials (CDT) and the detection delay (DD) calculated using the average number of tasks taken to detect the novelty [131]. Consider a

trial $t \in T$, where T represents a set of trials for a novelty-scenario, FP_t represents the number of normal tasks in trial t where the agent incorrectly detected as a novel task. TP_t represents the number of novel tasks in trial t where the agent correctly detected as a novel task. A correctly detected trial is a trial where the agent detected novelty only after entering the novel task sequence. *CDT* is defined in Equation 5.1. CDT varies between 0 and 1 and 1 is the best result.

$$CDT = \frac{1}{|T|} \sum_{t=1}^{|T|} \begin{cases} 1, & \text{if } FP_t = 0 \text{ and } TP_t \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

DD quantifies the delay in detection using the number of tasks required to correctly detect the novelty. *DD* is defined in Equation 5.2. The lower the DD, the better the detection performance (in terms of timeliness), and the best possible DD is 1.

$$DD = \frac{1}{N_{cdt}} \sum_{t=1}^{|T|} \begin{cases} d_t, & \text{if } FP_t{=}0 \text{ and } TP_t \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

where,

$$N_{cdt} = \sum_{t=1}^{|T|} \begin{cases} 1, & \text{if } FP_t{=}0 \text{ and } TP_t \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

and d_t is the number of novel tasks taken until the agent informs a novelty detection in trial t (including the task that the agent detected novelty).

To measure novelty adaptation performance, we use the area under the pass rate curve (success curve). First, in a given novelty-scenario, to measure the performance of the agent after adapting to novelty, we use the pass rate of the asymptotic tasks. We refer to this measure as the asymptotic performance (AP). In equation 5.4 for AP, n represents the length of the novel task sequence and m represents the asymptotic length we consider. The asymptotic length can be adjusted based on the percentage of novel tasks in the trial.

$$AP = \frac{1}{m} \sum_{i=n-m}^{n} \left(\frac{1}{|T|} \sum_{t=1}^{|T|} \begin{cases} 1, & \text{if } i^{th} \text{ novel task in } t^{th} \text{ trial is passed} \\ 0, & \text{otherwise} \end{cases} \right) \tag{5.4}$$

Second, as AP does not capture the timeliness of adaptation, we compute the total area under the success curve (AUS). AUS can be defined as follows.

$$AUS = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{|T|} \sum_{t=1}^{|T|} \begin{cases} 1, & \text{if } i^{th} \text{ novel task in } t^{th} \text{ trial is passed} \\ 0, & \text{otherwise} \end{cases} \right) \tag{5.5}$$

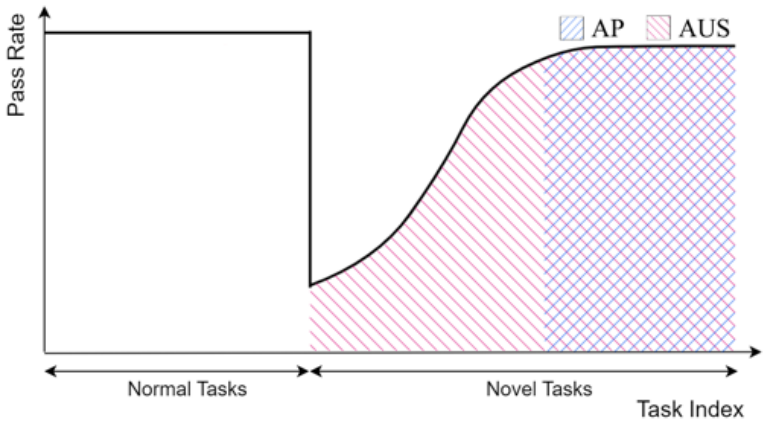


Figure 5.5: An example pass rate curve of an agent that played a set of trials. The area shaded in blue is considered for the asymptotic performance (AP) and the area shaded in red is considered for the area under success curve performance (AUS).

Both AP and AUS vary between 0 and 1 and 1 is the best achievable adaptation performance. Figure 5.5 graphically illustrates the areas considered in the pass rate curve of an agent when calculating AP and AUS.

Deriving from the above measures for novelty detection and novelty adaptation, we define *NPS* (Novelty Performance in Scenarios) and *SPN* (Scenario Performance under Novelties). *NPS* measures how agents perform in a single novelty, when the novelty is applied to different physical scenarios. *SPN* measures how agents perform in a single physical scenario, when different novelties are applied to the scenario. To calculate the *NPS* measures, the average performance is taken across all scenarios for a single novelty. Formally, consider i to represent the i^{th} novelty, and j to represent the j^{th} scenario. The NPS measures, $NPS_{CDT,i}$, $NPS_{DD,i}$, $NPS_{AP,i}$, and $NPS_{AUS,i}$ are defined for a given novelty i as the average of CDT_{ij}, DD_{ij}, AP_{ij}, and AUS_{ij} respectively for all the scenarios j with the novelty i. Similarly, for *SPN* measures, for a given scenario j, the average performance is taken across all the novelties i for the scenario j. i.e., *SPN* measures $SPN_{CDT,j}$, $SPN_{DD,j}$, $SPN_{AP,j}$, and $SPN_{AUS,j}$ are calculated from the average performance taken for CDT_{ij}, DD_{ij}, AP_{ij}, and AUS_{ij} respectively for all the novelties i in the scenario j.

5.4 Experiments

We conduct experiments on baseline agents on the 40 novelty-scenarios to measure how agents detect and adapt to novelty in each of those novelty-scenarios. In addition, we establish human performance on all the novelty-scenarios. This section describes the

baseline agents we provide and the experimental setups used for each experiment.

5.4.1 Baseline Agents

We include experimental results of 11 baseline agents which consist of three heuristic agents, seven learning agents, and a random agent.

Heuristic Agents The heuristic agents are based on hard-coded physical rules developed by agent developers. All the agents were participating agents from the AIBIRDS competition [2], which is an annual competition held to find the best Angry Birds game-playing AI agent. Following is the list of heuristic agents evaluated in NovPhy.

- Datalab: Datalab is a planning agent that has six strategies. The strategies include destroying pigs, destroying physical structures, and shooting at round blocks. The agent selects which action to take based on the game objects available, possible trajectories, the bird on the sling, and the birds remaining [30].
- Eagle's Wing: Eagle's Wing agent selects from a suit of five strategies based on structural analysis. The five strategies include: shooting at unprotected pigs, destroying many blocks as possible, and shooting at objects close to round objects [171].
- Pig Shooter: Pig Shooter has only one strategy: shooting at pigs. The agent randomly selects which pig to shoot and which trajectory to use [159].

All these heuristic agents work under the uninformed evaluation setting, in which the agent is not informed when the first novel task appears in the trial.

Learning Agents In this work, we evaluate seven learning agents/versions of agents. All seven learning agents we present here work under the informed evaluation setting in which the agent is informed when the novelty appears in the trial. Therefore, we do not evaluate the novelty detection performance of these agents. The seven agents are *DQN Offline/ Online/ Adapt*, *Relational Offline/ Online/ Adapt*, and *Naive Adapt*. Same as the deep reinforcement learning agents used in [184], we train a DQN [173, 167] agent and the Relational agent that contains a relational module [188]. Both agents are trained on the tasks generated from a normal task template and are evaluated on the trials that contain tasks from the corresponding novel task template. We evaluate the DQN and Relational agents in three versions: offline, online, and adapt. With the offline version, the two deep reinforcement learning agents DQN and Relational, always select the action with the highest q-value throughout the trial. On the other hand, online learning agents update the q-network after novelty is introduced and try to relearn the policy to solve novel tasks.

We also evaluate the recently developed open-world learning component **NAPPING** [183], which is also discussed in Chapter 6, together with DQN and Relational agents. We call these agents **DQN Adapt** and **Relational Adapt**.

The *Naive Adapt* is built on top of the *Pig Shooter* agent in [184], which shoots only at the pigs. *Naive Adapt* uses the strategy of the *Pig Shooter* in the pre-novelty game tasks. After the agent is informed that the novelty has occurred, it searches for a combination of (objects, trajectories, and delays) that solve a game level and keeps a record of each triplet tried. Once a solution triplet (e.g., a solution triplet could be (pig_2, high trajectory, delay 5 seconds)) is found for a trial, the *Naive Adapt* will keep using the triplet until it does not solve the tasks anymore, where the agent starts to search for another triplet.

Random Agent The Random Agent selects a random release point (x,y) relative to the slingshot. The x is sampled from [-200, 200] and y is sampled from [-200, 200]. This agent works under the uninformed evaluation setting.

5.4.2 Experimental Setups

Human Experiment Setup

The experiments with human participants were approved by the Australian National University committee on human ethics under the protocol 2021/293. Participation was entirely voluntary, and no monetary compensation was provided. There were 47 participants with ages ranging from 20 to 35 years and there were both males and females. They were not experienced Angry Birds players. Some of the participants have never played the game and some of them knew the general game mechanics and had played the game on an occasional basis in the past, but did not have an extensive understanding of the game's strategies. Participants provided their consent to use their play-data.

For a single participant, we provided 10 trials from 10 novelty-scenarios. We had four such trial sets to cover all 40 novelty-scenarios. In a single trial, there were 1-4 normal tasks and 4 novel tasks. On average participants spent 25-30 minutes to complete the experiment. Participants attempted to solve the tasks and at the end of each task they indicated if they detected a novelty or not.

Agent Experiment Setup

We use the standard SAIL-ON evaluation setup for all agents. As mentioned previously, we have eight novelties and five physical reasoning scenarios which results in 40 novelty-scenarios. For a single novelty-scenario, we test the agent on 40 trials. A trial consists of 1-40 normal tasks and 40 novel tasks. All heuristic agents and the Random agent do not require any training. However, learning agents are trained on the normal tasks of the corresponding novelty-scenario and then the agent is evaluated on the trial set.

5.5 Results and Analysis

In this section, we present and discuss the results of the experiments we conducted: the human player experiment and the baseline agent experiment. For both experiments, we

report the novelty detection and novelty adaptation performance.

5.5.1 Human Performance

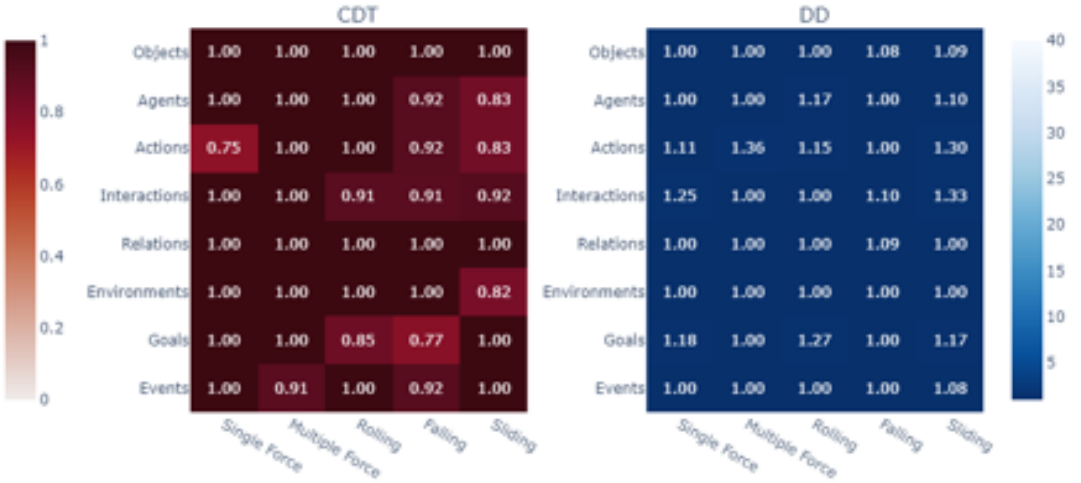


Figure 5.6: CDT (left) and DD (right) results of the human players. In the heat maps, the x-axis is the physical scenario and the y-axis is the novelty applied.

Figure 5.6 shows CDT and DD results of the human players. Overall, the participants were able to correctly detect the novelties in almost all the trials (CDT is close to 1). The lowest CDT is for the novelty-scenario actions-single force, in which the upward force of the Air Turbulence agent is increased. The reason for this is likely because this novelty is not visually detectable until interacted with. Also, when this novelty is applied to the single force scenario, the player has only to slightly adjust the shooting angle of the bird compared to the shooting angle in the normal tasks. As the impact of this novelty is subtle, it might not be perceivable to humans. This is also likely to be the case with the novelty-scenario that has the second lowest CDT: goals-falling. When we look at the DD results, in most cases it can be seen that the participants could detect the novelty in the first game level where the novelty was encountered (DD is close to 1). Generally, it can also be seen that for the novelties that are not visually detectable without interaction (actions, interactions, and goals), humans have a higher detection delay compared to the other novelties that can be visually detected before interaction.

The AP and AUS performances of the human players are shown in Figure 5.7. For the AP calculation, we used the asymptotic length as 2 (i.e., the performance of the last two tasks in the trial). As the AP results depict, the participants obtained above 80% performance for most of the novelty-scenarios. Generally, for both the AP and AUS, participants showed lower results for the novelty-scenarios which required highly accurate actions. For example, interactions-single force, goals-single force, events-multiple forces, interactions-sliding, and actions-falling have slightly lower results for both the measures as it is required to shoot the bird with high accuracy to solve the tasks in those novelty-

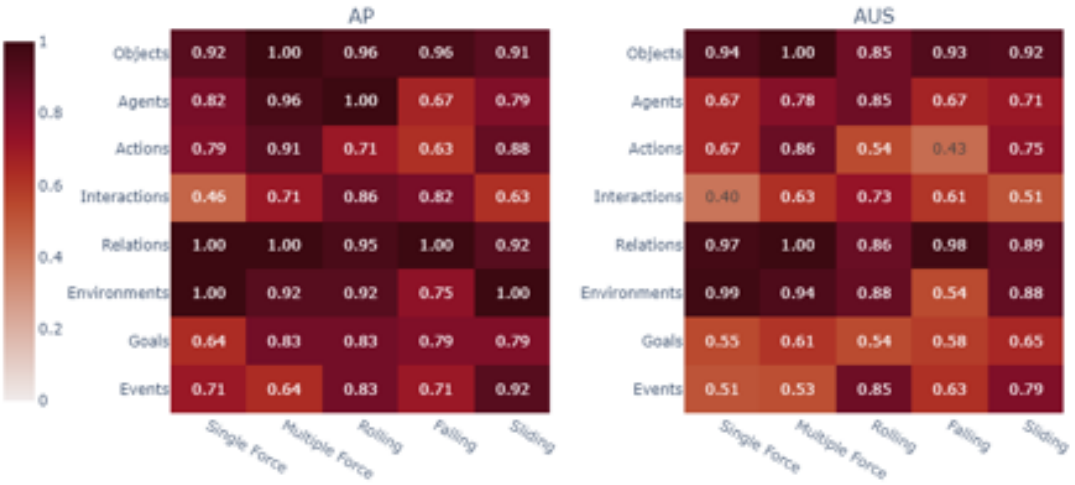


Figure 5.7: AP (left) and AUS (right) results of the human players. In the heat maps, the x-axis is the physical scenario and the y-axis is the novelty applied. The asymptotic length considered for the AP calculation is 2.

scenarios. The rate of adaptation is captured from the AUS results. When the accuracy requirements are higher, it takes more tasks for humans to adapt to the novelty. This is reflected in the relatively lower AUS results for the novelties that demand accurate actions: interactions, goals, and events. On the other hand, novelties that do not require accurate actions such as objects and relations have nearly perfect AUS results, as humans can adapt to such novelties straight away.

Moreover, we analyzed the relationship between the detection performance and adaptation performance of human players using the non-parametric Mann-Whitney test and Spearman's rank correlation (see 5.7.6 for more details). Considering the relationship between CDT and adaptation (both the AP and AUS), only the results of the actions-single force novelty-scenario shows that, the adaptation performance is independent of whether participants detected it or not. As discussed in the above paragraphs, this is because, even though humans have successfully adapted to the actions-single force, the novelty may not be perceivable to humans. The correlation between detection delay and adaptation performance (both the AP and AUS) shows that there are some novelty-scenarios such as goals-single force, goals-rolling, and interactions-sliding have moderately negative correlations implying that the longer a player takes to detect the novelty, the lower the adaptation performance. The correlation plots are shown in Appendix Figure 5.44.

5.5.2 Baseline Agent Performance

In this section, we discuss the performance of the 11 baseline agents and humans in terms of novelty detection and novelty adaptation.

Baseline Agent Novelty Detection Performance

As discussed in Section 5.4.1 some of the agents in NovPhy work under the uninformed evaluation. We compute the detection performance measures for those agents. The detection modules in those baseline agents are based on the pass rate deviation (discussed in detail in 5.7.4). The results of the detection measures per novelty are presented in Table 5.2 and detection measures per scenario are presented in Table 5.3 for all the heuristic agents, the random agent, and humans.

As the results indicate, Datalab has the highest overall CDT while Pig Shooter has the lowest. It is expected for Pig Shooter to have the lowest CDT as the agent only directly shoots at the pigs, generally resulting in the same outcome in the tasks of a given trial, hence does not show a significant deviation in pass rates. Considering the overall DD, Random Agent has the highest DD. This is because the agent's randomness in the actions results in novelty detection at random positions of a trial causing the overall DD to fall around the mean number of the tasks in the trial. The Pig Shooter has the best (lowest) DD, this is because, for the few trials it correctly detects, it is done rapidly due to the deterministic nature of action selection. All the agents are far below the humans' novelty detection performance which is near perfect (CDT = 0.96 and DD = 1.07).

Baseline Agent Novelty Adaptation Performance

The adaptation plots per novelty are shown in Figure 5.8 and the adaptation plots per scenario are shown in Figure 5.9. The novelty adaptation measures derived from the adaptation curves, AP and AUS results per scenario are presented in Table 5.2 and per novelty results are presented in Table 5.3. The AP results are based on the last 50% of the novel tasks (m=20).

Ideally, an agent would have a higher pass rate in normal tasks and when novel tasks begin (at index 0 in Figures 5.8 and 5.9), the performance would drop and recover its performance to reach the normal task performance within a few tasks. Therefore, an ideal agent would have AP and AUS results close to 1.

In Figure 5.9, the agents DQN Adapt and Relational Adapt show a better adaptation behaviour compared to other agents in four scenarios out of five, the only exception is multiple forces. In those four scenarios, out of the two agents, DQN Adapt shows slightly better performance (at 10% level of significance) than Relational Adapt when we look at the $SPN_{AP,j}$, and $SPN_{AUS,j}$. Similarly as shown in Figure 5.8, the two agents adapt in five novelties except the novelties Relations, Environments, and Events. This is because that to adapt to the Relations and Environments novelties, the agent needs to adjust the pre-defined trajectory planner. For example, in the Environments novelty, the trajectory is upside down. However, the -Adapt agents currently use the provided trajectory planner. Similarly, Naive Adapt agent also shows the adaptation behaviour in all scenarios except for single force. This agent shows an adaptation behaviour in all the novelties except Relations, Environments, and Events. However, as the Naive Adapt agent searches for just one solution tuple through a trial, it can not adapt to novelty

Table 5.2: Results of the NPS measures of the agents and humans for the eight novelties.

Novelty	Measure	Human	DQN Offline	DQN Online	DQN Adapt	Rel Offline	Rel Online	Rel Adapt	Naive Adapt	Datalab	Eagle'sWing	PigShooter	Random
1. Objects	$NPS_{CDT,1}$	1.00 ± 0.00	-	-	-	-	-	-	-	0.65 ± 0.13	0.57 ± 0.08	0.25 ± 0.08	0.23 ± 0.09
	$NPS_{DD,1}$	1.03 ± 0.02	-	-	-	-	-	-	-	7.85 ± 4.04	5.46 ± 1.17	6.07 ± 2.53	12.79 ± 4.31
	$NPS_{AP,1}$	0.95 ± 0.02	0.65 ± 0.17	0.66 ± 0.14	0.79 ± 0.10	0.54 ± 0.10	0.50 ± 0.10	0.80 ± 0.06	0.61 ± 0.19	0.37 ± 0.19	0.40 ± 0.21	0.02 ± 0.02	0.01 ± 0.01
	$NPS_{AUS,1}$	0.93 ± 0.02	0.66 ± 0.17	0.67 ± 0.16	0.81 ± 0.10	0.56 ± 0.11	0.55 ± 0.10	0.80 ± 0.06	0.61 ± 0.19	0.38 ± 0.19	0.41 ± 0.21	0.02 ± 0.02	0.01 ± 0.01
2. Agents	$NPS_{CDT,2}$	0.95 ± 0.03	-	-	-	-	-	-	-	0.49 ± 0.17	0.42 ± 0.12	0.17 ± 0.07	0.24 ± 0.07
	$NPS_{DD,2}$	1.05 ± 0.03	-	-	-	-	-	-	-	10.96 ± 4.29	10.01 ± 2.25	4.57 ± 0.39	19.59 ± 1.42
	$NPS_{AP,2}$	0.85 ± 0.02	0.06 ± 0.03	0.21 ± 0.07	0.73 ± 0.13	0.07 ± 0.04	0.11 ± 0.04	0.67 ± 0.09	0.11 ± 0.06	0.05 ± 0.03	0.05 ± 0.03	0.01 ± 0.01	0.01 ± 0.00
	$NPS_{AUS,2}$	0.73 ± 0.03	0.06 ± 0.03	0.18 ± 0.06	0.62 ± 0.10	0.07 ± 0.04	0.10 ± 0.04	0.56 ± 0.09	0.09 ± 0.05	0.05 ± 0.03	0.05 ± 0.03	0.00 ± 0.00	0.01 ± 0.00
3. Actions	$NPS_{CDT,3}$	0.90 ± 0.05	-	-	-	-	-	-	-	0.44 ± 0.15	0.49 ± 0.19	0.24 ± 0.02	0.17 ± 0.06
	$NPS_{DD,3}$	1.19 ± 0.07	-	-	-	-	-	-	-	10.11 ± 1.69	8.12 ± 2.20	5.67 ± 0.57	19.27 ± 1.73
	$NPS_{AP,3}$	0.78 ± 0.05	0.07 ± 0.06	0.13 ± 0.08	0.56 ± 0.17	0.05 ± 0.03	0.12 ± 0.07	0.63 ± 0.16	0.10 ± 0.07	0.03 ± 0.02	0.08 ± 0.04	0.00 ± 0.00	0.01 ± 0.00
	$NPS_{AUS,3}$	0.65 ± 0.08	0.08 ± 0.06	0.13 ± 0.08	0.53 ± 0.15	0.05 ± 0.03	0.10 ± 0.06	0.55 ± 0.14	0.08 ± 0.05	0.03 ± 0.02	0.08 ± 0.04	0.00 ± 0.00	0.01 ± 0.00
4. Interactions	$NPS_{CDT,4}$	0.95 ± 0.02	-	-	-	-	-	-	-	0.42 ± 0.15	0.46 ± 0.18	0.18 ± 0.05	0.55 ± 0.12
	$NPS_{DD,4}$	1.14 ± 0.07	-	-	-	-	-	-	-	5.87 ± 1.32	16.31 ± 7.38	5.25 ± 0.35	18.40 ± 5.13
	$NPS_{AP,4}$	0.69 ± 0.07	0.10 ± 0.04	0.24 ± 0.07	0.75 ± 0.16	0.08 ± 0.03	0.19 ± 0.05	0.72 ± 0.15	0.33 ± 0.14	0.10 ± 0.06	0.3 ± 0.17	0.03 ± 0.02	0.06 ± 0.02
	$NPS_{AUS,4}$	0.58 ± 0.05	0.10 ± 0.04	0.25 ± 0.08	0.64 ± 0.14	0.08 ± 0.03	0.19 ± 0.04	0.63 ± 0.13	0.27 ± 0.11	0.18 ± 0.07	0.35 ± 0.18	0.02 ± 0.02	0.06 ± 0.02
5. Relations	$NPS_{CDT,5}$	1.00 ± 0.00	-	-	-	-	-	-	-	0.27 ± 0.18	0.28 ± 0.17	0.08 ± 0.07	0.01 ± 0.01
	$NPS_{DD,5}$	1.02 ± 0.02	-	-	-	-	-	-	-	3.47 ± 0.40	7.49 ± 0.02	3.86 ± 0.54	4.00 ± 0.00
	$NPS_{AP,5}$	0.97 ± 0.02	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	$NPS_{AUS,5}$	0.94 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
6. Environments	$NPS_{CDT,6}$	0.96 ± 0.04	-	-	-	-	-	-	-	0.44 ± 0.15	0.35 ± 0.13	0.01 ± 0.01	0.40 ± 0.10
	$NPS_{DD,6}$	1.00 ± 0.00	-	-	-	-	-	-	-	4.75 ± 0.19	5.50 ± 1.03	2.50 ± 0.95	14.64 ± 0.83
	$NPS_{AP,6}$	0.92 ± 0.05	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.01
	$NPS_{AUS,6}$	0.85 ± 0.08	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.03 ± 0.01
7. Goals	$NPS_{CDT,7}$	0.92 ± 0.05	-	-	-	-	-	-	-	0.43 ± 0.13	0.55 ± 0.10	0.19 ± 0.05	0.39 ± 0.09
	$NPS_{DD,7}$	1.12 ± 0.05	-	-	-	-	-	-	-	11.62 ± 1.48	12.28 ± 2.99	5.73 ± 0.22	15.06 ± 2.40
	$NPS_{AP,7}$	0.78 ± 0.04	0.07 ± 0.02	0.09 ± 0.03	0.45 ± 0.15	0.04 ± 0.02	0.14 ± 0.08	0.54 ± 0.18	0.29 ± 0.12	0.08 ± 0.04	0.09 ± 0.03	0.01 ± 0.01	0.02 ± 0.01
	$NPS_{AUS,7}$	0.59 ± 0.02	0.08 ± 0.03	0.10 ± 0.03	0.42 ± 0.13	0.04 ± 0.02	0.12 ± 0.05	0.49 ± 0.16	0.29 ± 0.12	0.08 ± 0.04	0.09 ± 0.04	0.01 ± 0.01	0.02 ± 0.01
8. Events	$NPS_{CDT,8}$	0.97 ± 0.02	-	-	-	-	-	-	-	0.46 ± 0.13	0.46 ± 0.12	0.07 ± 0.03	0.66 ± 0.80
	$NPS_{DD,8}$	1.02 ± 0.02	-	-	-	-	-	-	-	7.36 ± 1.73	11.00 ± 2.36	4.80 ± 1.35	18.14 ± 2.18
	$NPS_{AP,8}$	0.76 ± 0.05	0.27 ± 0.12	0.21 ± 0.10	0.21 ± 0.1	0.29 ± 0.11	0.2 ± 0.09	0.2 ± 0.09	0.00 ± 0.00	0.13 ± 0.05	0.12 ± 0.04	0.01 ± 0.00	0.02 ± 0.01
	$NPS_{AUS,8}$	0.66 ± 0.07	0.27 ± 0.12	0.24 ± 0.11	0.23 ± 0.1	0.29 ± 0.11	0.22 ± 0.10	0.21 ± 0.09	0.00 ± 0.00	0.13 ± 0.05	0.12 ± 0.04	0.01 ± 0.00	0.02 ± 0.01

Table 5.3: Results of the SPN measures of the agents and humans for the five scenarios.

Scenario	Measure	Human	DQN Offline	DQN Online	DQN Adapt	Rel Offline	Rel Online	Rel Adapt	Naive Adapt	Datalab	Eagle'sWing	PigShooter	Random
1. Single Force	$SPN_{CDT,1}$	0.97 ± 0.03	-	-	-	-	-	-	-	0.59 ± 0.07	0.67 ± 0.05	0.08 ± 0.04	0.31 ± 0.10
	$SPN_{DD,1}$	1.07 ± 0.04	-	-	-	-	-	-	-	6.61 ± 0.88	7.42 ± 1.46	4.82 ± 1.02	14.21 ± 2.62
	$SPN_{AP,1}$	0.79 ± 0.07	0.11 ± 0.05	0.13 ± 0.05	0.37 ± 0.14	0.12 ± 0.06	0.13 ± 0.05	0.36 ± 0.13	0.13 ± 0.11	0.08 ± 0.04	0.16 ± 0.11	0.01 ± 0.00	0.03 ± 0.02
	$SPN_{AUS,1}$	0.71 ± 0.08	0.11 ± 0.05	0.13 ± 0.05	0.33 ± 0.12	0.12 ± 0.06	0.13 ± 0.06	0.33 ± 0.11	0.13 ± 0.12	0.08 ± 0.05	0.17 ± 0.12	0.01 ± 0.00	0.03 ± 0.02
2. Multiple Force	$SPN_{CDT,2}$	0.99 ± 0.01	-	-	-	-	-	-	-	0.55 ± 0.14	0.55 ± 0.10	0.14 ± 0.03	0.14 ± 0.09
	$SPN_{DD,2}$	1.05 ± 0.05	-	-	-	-	-	-	-	4.29 ± 1.28	4.88 ± 0.47	5.16 ± 0.38	17.48 ± 1.26
	$SPN_{AP,2}$	0.87 ± 0.05	0.07 ± 0.03	0.15 ± 0.06	0.11 ± 0.05	0.05 ± 0.03	0.06 ± 0.03	0.16 ± 0.08	0.17 ± 0.12	0.01 ± 0	0.08 ± 0.05	0.00 ± 0.00	0.01 ± 0.00
	$SPN_{AUS,2}$	0.79 ± 0.07	0.08 ± 0.03	0.14 ± 0.06	0.12 ± 0.06	0.04 ± 0.02	0.07 ± 0.03	0.15 ± 0.07	0.17 ± 0.12	0.05 ± 0.05	0.10 ± 0.07	0.00 ± 0.00	0.01 ± 0.00
3. Rolling	$SPN_{CDT,3}$	0.97 ± 0.02	-	-	-	-	-	-	-	0.26 ± 0.07	0.24 ± 0.08	0.20 ± 0.04	0.34 ± 0.07
	$SPN_{DD,3}$	1.07 ± 0.04	-	-	-	-	-	-	-	10.49 ± 3.11	12.02 ± 2.28	4.45 ± 0.42	16.13 ± 1.71
	$SPN_{AP,3}$	0.88 ± 0.03	0.24 ± 0.13	0.27 ± 0.12	0.59 ± 0.14	0.24 ± 0.11	0.22 ± 0.10	0.60 ± 0.14	0.18 ± 0.10	0.07 ± 0.03	0.07 ± 0.03	0.00 ± 0.00	0.02 ± 0.00
	$SPN_{AUS,3}$	0.76 ± 0.05	0.25 ± 0.13	0.28 ± 0.12	0.57 ± 0.13	0.25 ± 0.11	0.23 ± 0.11	0.57 ± 0.13	0.15 ± 0.08	0.07 ± 0.02	0.07 ± 0.03	0.00 ± 0.00	0.02 ± 0.00
4. Falling	$SPN_{CDT,4}$	0.93 ± 0.03	-	-	-	-	-	-	-	0.48 ± 0.11	0.44 ± 0.11	0.19 ± 0.05	0.36 ± 0.08
	$SPN_{DD,4}$	1.03 ± 0.02	-	-	-	-	-	-	-	10.04 ± 1.75	12.51 ± 2.37	4.89 ± 0.48	13.35 ± 1.98
	$SPN_{AP,4}$	0.79 ± 0.05	0.17 ± 0.11	0.20 ± 0.1	0.57 ± 0.14	0.14 ± 0.09	0.2 ± 0.08	0.55 ± 0.14	0.21 ± 0.09	0.19 ± 0.10	0.19 ± 0.10	0.02 ± 0.01	0.02 ± 0.01
	$SPN_{AUS,4}$	0.67 ± 0.07	0.18 ± 0.11	0.22 ± 0.10	0.51 ± 0.13	0.14 ± 0.09	0.19 ± 0.08	0.50 ± 0.13	0.19 ± 0.09	0.20 ± 0.10	0.19 ± 0.10	0.02 ± 0.01	0.02 ± 0.01
5. Sliding	$SPN_{CDT,5}$	0.93 ± 0.03	-	-	-	-	-	-	-	0.37 ± 0.13	0.33 ± 0.13	0.12 ± 0.06	0.25 ± 0.05
	$SPN_{DD,5}$	1.13 ± 0.04	-	-	-	-	-	-	-	9.10 ± 1.72	12.97 ± 4.41	6.99 ± 2.25	22.12 ± 2.83
	$SPN_{AP,5}$	0.85 ± 0.04	0.17 ± 0.11	0.21 ± 0.12	0.55 ± 0.15	0.13 ± 0.08	0.18 ± 0.07	0.55 ± 0.14	0.21 ± 0.10	0.13 ± 0.10	0.16 ± 0.11	0.02 ± 0.01	0.01 ± 0.00
	$SPN_{AUS,5}$	0.76 ± 0.05	0.18 ± 0.12	0.21 ± 0.12	0.49 ± 0.13	0.13 ± 0.08	0.18 ± 0.07	0.48 ± 0.12	0.19 ± 0.10	0.14 ± 0.10	0.16 ± 0.12	0.01 ± 0.01	0.01 ± 0.00

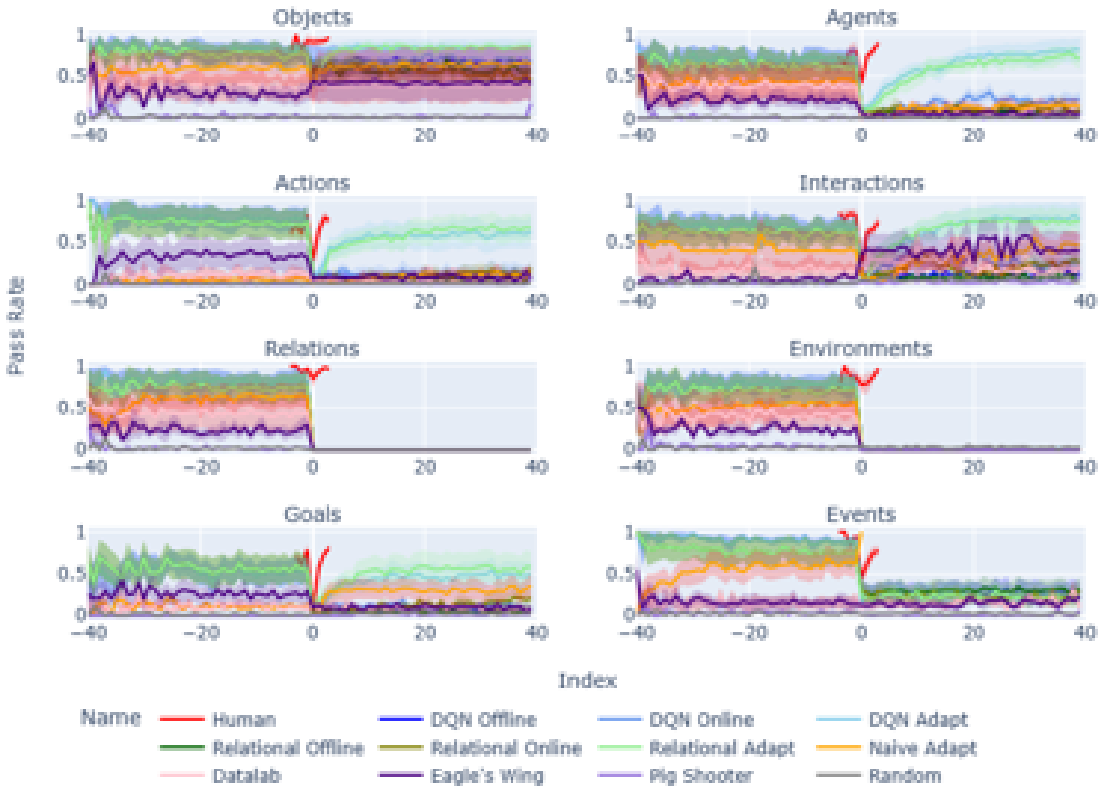


Figure 5.8: The pass rate of the agents per novelty. The x-axis represents the index of the task in trials. Indexes -40 to -1 represent normal tasks and 0 to 40 represent novel tasks. The y-axis shows the pass rate averaged across all trials relevant to the respective novelty. **DQN Adapt and Relational Adapt are the agents with NAPPING, which is discussed in Chapter 6.**

trials that require different solution tuples under different situations. As a result, the Naive Adapt agent does not reach the performance level of the two agents, DQN Adapt and Relational Adapt, who learn efficiently how to handle novelties in different scenarios through each trial. Overall, the agents show the best adaptation performance in the Objects novelty. This is because the Objects novelty tested here is a change of the colour of an existing object, which has not impacted the agents' actions drastically. It is interesting to note that none of the agents has reached the humans' pass rate in any scenario or in any novelty, except for interactions novelty where Relational Adapt and DQN Adapt exceed the humans' $NPS_{AP,j}$, and $NPS_{AUS,j}$. However, within the same number of tasks that were given to humans, those two agents have not reached the performance the humans could achieve, showing that there is room for improvement in terms of adaptation efficiency.

Figure 5.9: The pass rate of the agents per scenario. x-axis represents the index of the task in trials. Indexes -40 to -1 represent normal tasks and 0 to 40 represent novel tasks. The y-axis shows the pass rate averaged across all trials relevant to the respective scenario. **DQN Adapt and Relational Adapt are the agents with NAPPING, which is discussed in Chapter 6.**

5.6 Conclusion and Future Work

The objective of NovPhy is to facilitate the development of AI systems that can perform physical reasoning tasks in the presence of novelties, which is the condition that a system in an open-world physical environment would encounter. Towards this objective, NovPhy was designed to evaluate the abilities of an agent to detect novelties and adapt to perform under those novelties in a physical environment. We designed task templates for five commonly encountered real-world physical scenarios. Then, we designed novel tasks by introducing a diverse set of novelties to those task templates. This design enables to measure novelty detection and adaptation of agents in two directions: 1) how an agent performs in a novelty when the novelty is applied to different physical scenarios, and 2) how an agent performs in a physical scenario when different novelties are applied to it. To measure the true novelty adaptation performance of the agents, when designing the tasks we ensure that the agent has to work under the influence of the novelty rather than bypassing the novelties to solve the tasks. We evaluated the agents using a trial setting, in which the agent has to play a sequence of tasks of a scenario without novelties followed by a sequence of tasks of that scenario with novelties.

We have established the baseline results of the testbed using human players, learning agents, and heuristic agents. The results show that novelties affect the agents' performance severely and some agents can recover as they play more and more tasks. However, agents' solving rate and efficiency in adaptation are subpar compared to humans' performance. Although our results show that DQN Adapt and Relational Adapt agents are able to adapt to most novelties, there are still some novelties that the agents fail to adapt to. The main reason is that the agents still use the provided trajectory planner to plan for the shot (the release point). Future work on agents may focus on easing the need of using a trajectory planner to allow the agent to adapt to a wider range of novelties.

We foresee different directions of improvement for NovPhy. NovPhy can be advanced by introducing more novelties representing the levels of the novelty hierarchy. Further, more complex physical reasoning scenarios such as relative height, relative weight, and clearing paths can be introduced to the testbed after agents show efficient novelty detection and novelty adaptation in the existing scenarios. Moreover, the testbed can be extended to assess the novelty characterization ability of agents (i.e., to evaluate whether an agent correctly detects 'what is novel' in a task). Additionally, the concept in NovPhy, evaluating the physical reasoning capabilities under the influence of novelties, can be extended to other physical reasoning domains. For example, novelties can be introduced to physics-based robotic benchmarks such as CausalWorld [1] and RLBench [74], which will facilitate evaluating agents on physical scenarios such as pulling, picking and placing, and stacking, which are not seen in the Angry Birds domain. We believe that NovPhy builds the foundation for future research on developing agents that can efficiently detect and adapt to novelty in the physical world as humans do.

With the establishment of Phy-Q (Chapter 3) for physical reasoning assessment, the

Science Birds Novelty framework (Chapter 4) for introducing novelties and evaluating AI performance, and the introduction of NovPhy, a comprehensive benchmark for open-world learning agents, the focus now shifts to the direct development of AI agents equipped to tackle open-world learning challenges.

5.7 Appendix

5.7.1 The Objects in NovPhy Tasks

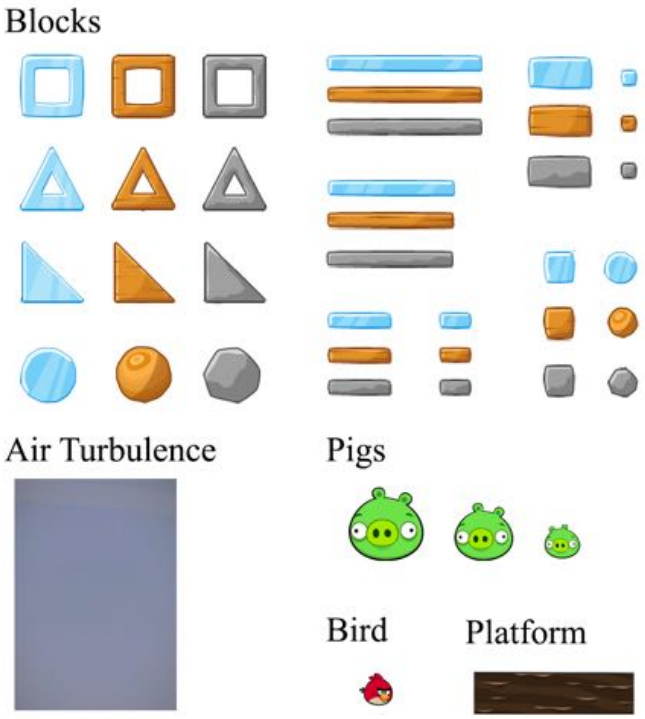


Figure 5.10: Objects that appear in normal tasks in NovPhy (not to the scale). The shape and the size of the objects are fixed, except the platform object that can appear in different shapes and sizes.

5.7.2 Tasks in NovPhy

NovPhy contains 40 novel task templates and corresponding 40 normal task templates designed for five physical scenarios by applying eight novelties. The figures 5.12, 5.13, 5.14, 5.15, and 5.16 shows the normal and novel task templates of the five scenarios: single force, multiple forces, rolling, falling, and sliding, respectively. Figure 5.17 shows tasks generated from five example task templates (of the objects novelty).

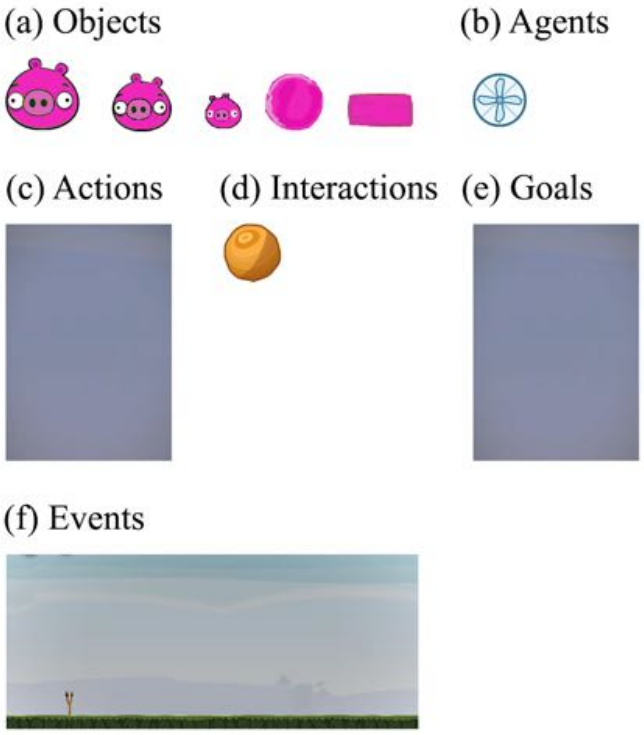
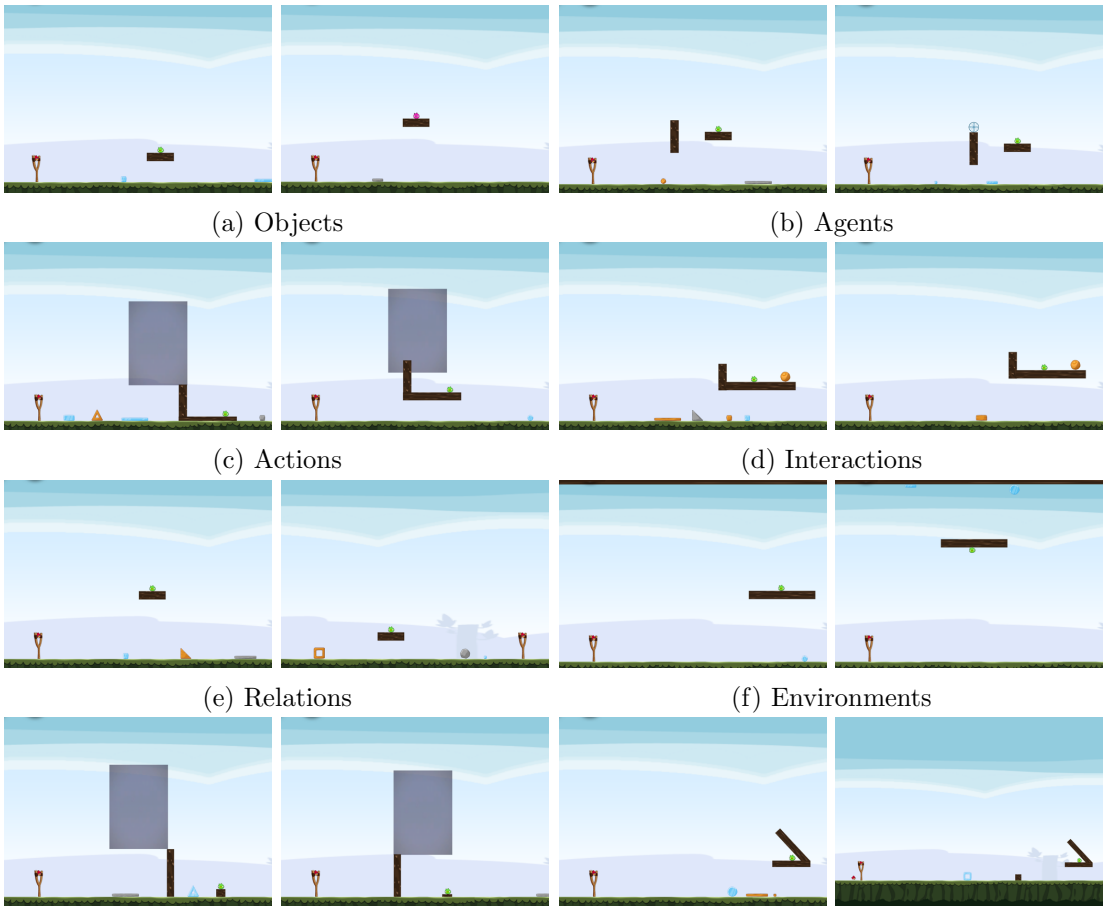


Figure 5.11: Objects associated with the novelties in NovPhy (not to the scale). Relations (the slingshot which is normally at the left side of the tasks is now at the right side of the tasks) and Environments (the gravity in the environment is now inverted) novelties are not shown here as they are not associated with a specific game object. Shown in the figures are, (a) Objects: a new pig/block that has a different colour (pink) to the normal pigs/blocks, (b) Agents: a novel external agent, Fan, that blows air (horizontally from left to right) affecting the moving path of objects, (c) Actions: the Air Turbulence external agent increases the magnitude of its upward force (has the same appearance of the Air Turbulence agent), (d) Interactions: existing circular wood object now has magnetic properties: repels objects of its type and attracts other object types (has the same appearance of the circular wood object), (e) Goals: the Air Turbulence external agent changes its goal from pushing objects up to pushing objects down (has the same appearance of the Air Turbulence agent), (f) Events: when the first bird is dead, a storm occurs that applies a force to the right direction affecting the motion of the objects (figure shows the appearance of the environment when the storm has occurred).



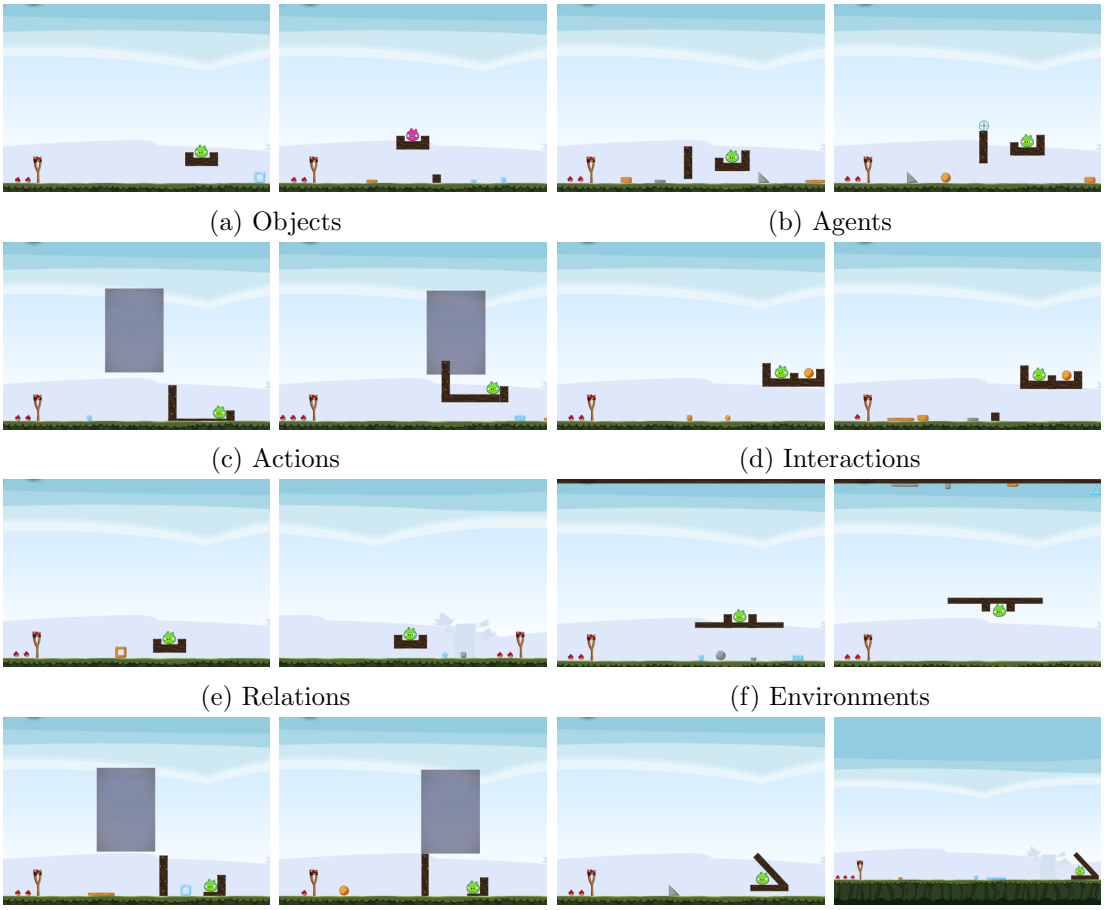
(a) Objects (b) Agents

(c) Actions (d) Interactions

(e) Relations (f) Environments

(g) Goals (h) Events

Figure 5.12: Task templates of the single force scenario with eight novelties applied to them. In each subfigure, the left figure is the normal task and the right figure is the corresponding novel task with the novelty applied.



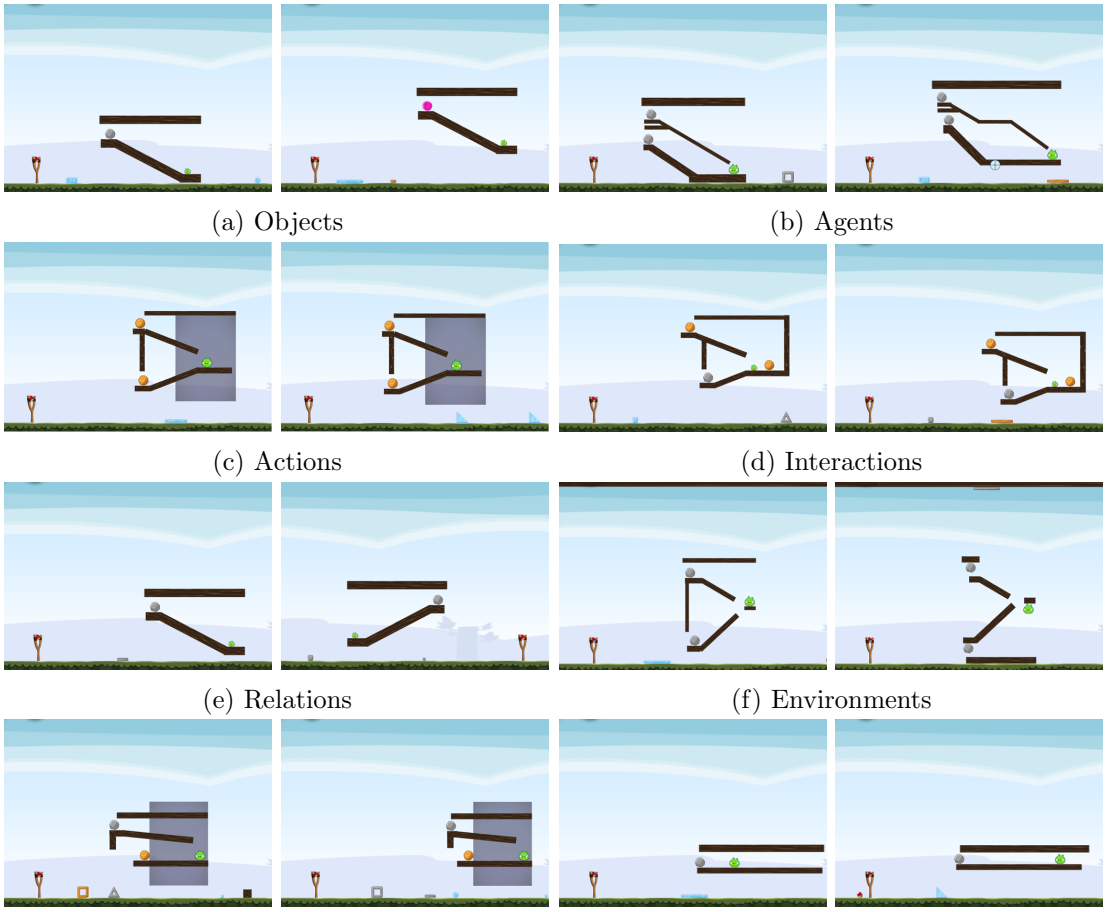
(a) Objects (b) Agents

(c) Actions (d) Interactions

(e) Relations (f) Environments

(g) Goals (h) Events

Figure 5.13: Task templates of the multiple forces scenario with eight novelties applied to them. In each subfigure, the left figure is the normal task and the right figure is the corresponding novel task with the novelty applied.



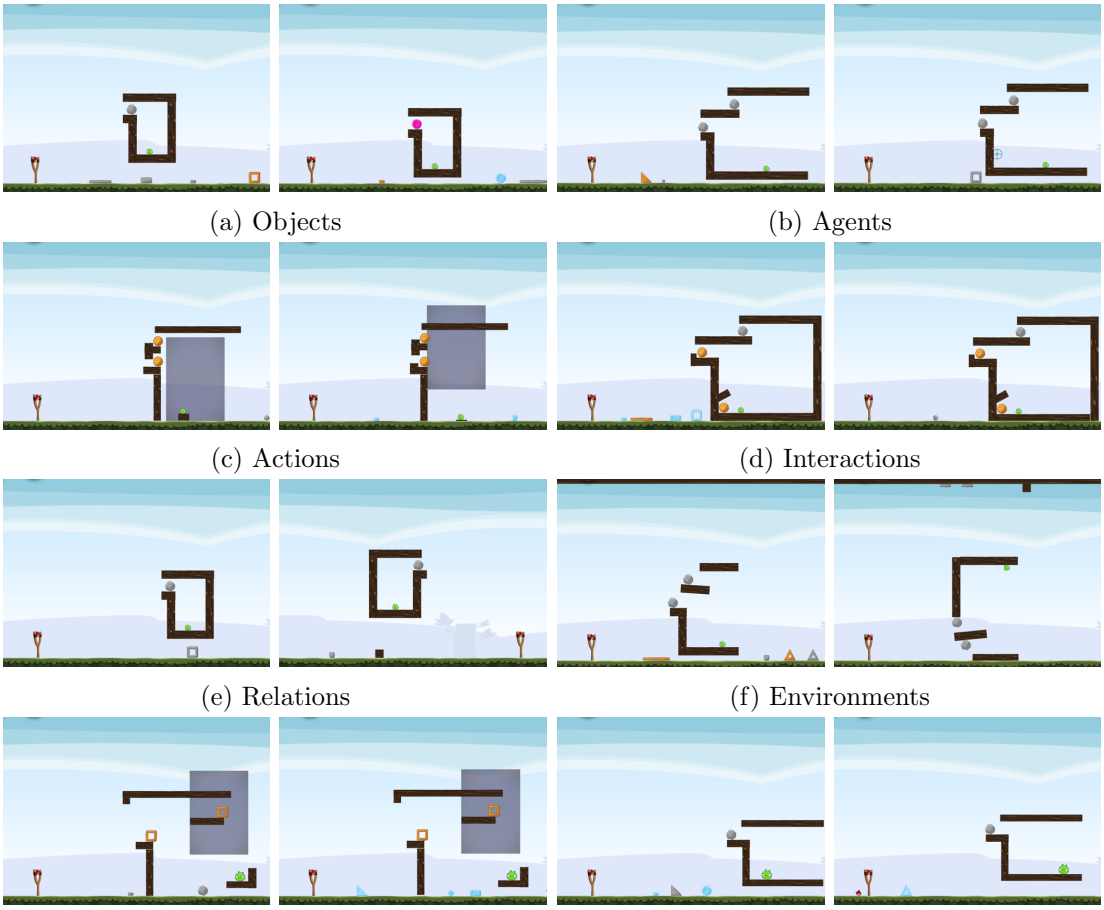
(a) Objects (b) Agents

(c) Actions (d) Interactions

(e) Relations (f) Environments

(g) Goals (h) Events

Figure 5.14: Task templates of the rolling scenario with eight novelties applied to them. In each subfigure, the left figure is the normal task and the right figure is the corresponding novel task with the novelty applied.



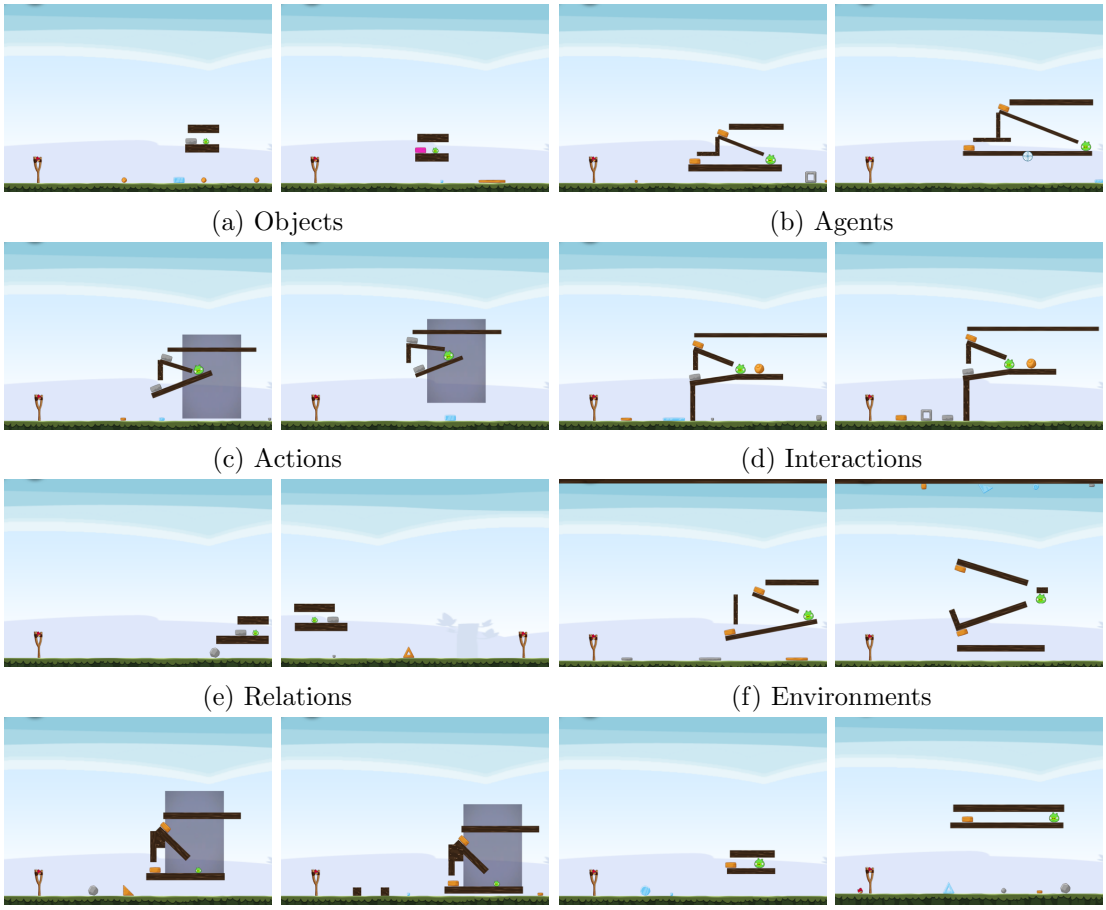
(a) Objects (b) Agents

(c) Actions (d) Interactions

(e) Relations (f) Environments

(g) Goals (h) Events

Figure 5.15: Task templates of the falling scenario with eight novelties applied to them. In each subfigure, the left figure is the normal task and the right figure is the corresponding novel task with the novelty applied.



(a) Objects (b) Agents

(c) Actions (d) Interactions

(e) Relations (f) Environments

(g) Goals (h) Events

Figure 5.16: Task templates of the sliding scenario with eight novelties applied to them. In each subfigure, the left figure is the normal task and the right figure is the corresponding novel task with the novelty applied.

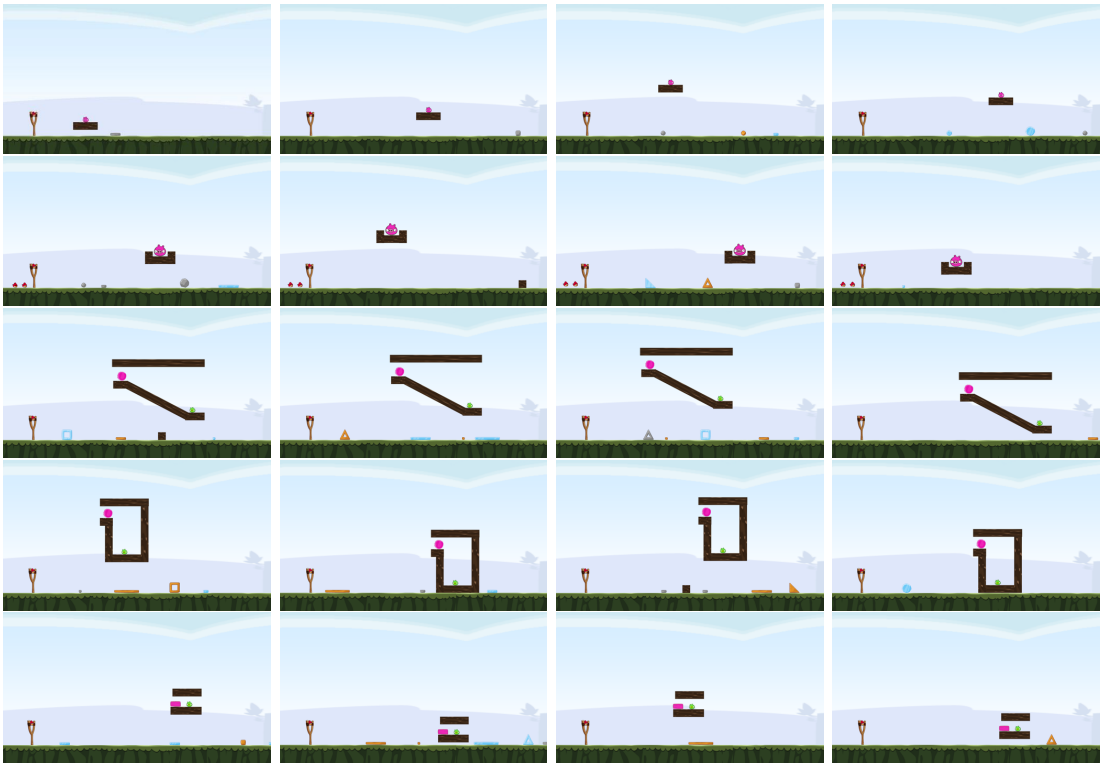


Figure 5.17: Each row shows four example tasks generated from the same task template. The positions of the objects and the distraction objects vary within the tasks of the same template, therefore each task has its own solution; however, all tasks of the same template can be solved by the same physical rule of the physical scenario that task template belongs to.

5.7.3 Designing Novel Tasks in NovPhy

As discussed in Section 3.1 of the main paper, when designing novel tasks, we followed the desideratum - the agent has to work under the effects of the novelty to solve the task. To satisfy this desideratum, when designing novel tasks we ensure that the novelty is introduced to the task such that the novelty affects at least one of the physical interaction phases (initial, middle, or final) in the solution of the task. Table 5.4 shows the physical interaction phases that were affected when designing the 40 novelty-scenarios in NovPhy.

Table 5.4: table

Physical Scenario	Novelty	Physical Interaction Phases Affected
Single force	1	final
	2	middle
	3	middle
	4	middle
	5	initial, middle
	6	initial, middle
	7	middle
	8	middle
Multiple forces	1	final
	2	middle
	3	middle, final
	4	middle, final
	5	initial, middle
	6	initial, middle
	7	middle, final
	8	middle
Rolling	1	initial
	2	middle
	3	middle
	4	middle
	5	initial, middle
	6	initial, middle
	7	middle
	8	middle
Falling	1	initial
	2	middle
	3	middle
	4	middle
	5	initial, middle
	6	initial, middle
	7	middle
	8	middle
Sliding	1	initial
	2	middle
	3	middle
	4	middle
	5	initial, middle
	6	initial, middle
	7	middle
	8	middle

The physical interaction phases of the solution that were affected by the novelty. Novelties 1 to 8 represents: 1. objects, 2. agents, 3. actions, 4. interactions, 5. relations, 6. environments, 7. goals, and 8. events.

5.7.4 Novelty Detection Performance

The detection performance is only measured in the agents that performed under the uninformed condition: Datalab, Eagle's Wing, Pig Shooter, and Random. The agent's detection module is based on the pass rate deviation. We have incorporated two algorithms, 1) simple moving average pass rate (sma) and 2) pre-assumed moving average pass rate (pma) that can be used as detection modules.

The simple moving average method compares two consecutive moving average pass rates of a given window size and checks if the second pass rate deviates from the first pass rate more than the selected threshold. We have studied the variations of window sizes 5 and 10, and thresholds 0.2, 0.4, 0.6, and 0.8.

In the pre-assumed moving average method, we assume that a trained agent knows the pass rate of the normal tasks of a given task template as those tasks are provided to the agent for training beforehand. Therefore, in this method, we check if the pass rate within a given window size is deviated by a number of standard deviations (the detection thresholds). We have studied variations of window sizes 5 and 10, and thresholds 1.5 and 2.

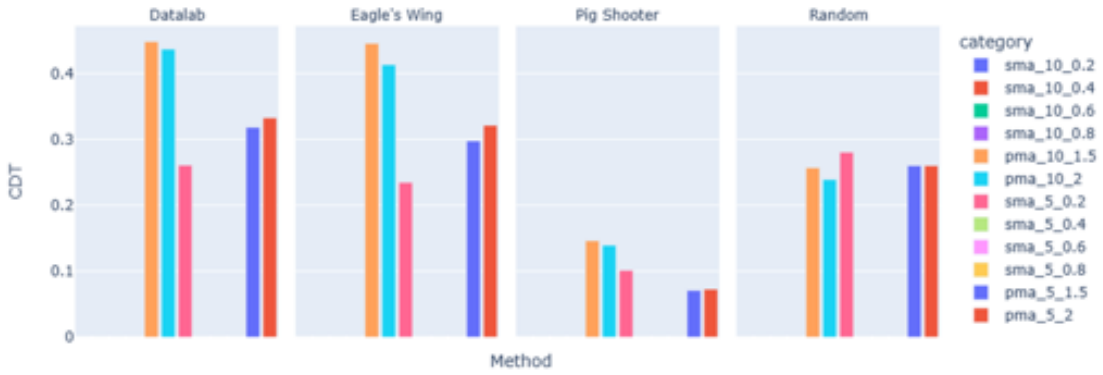


Figure 5.18: The CDT of the agents under the two algorithms sma and pma under variation of window size (the first value following the algorithm name) and detection thresholds (the last value).

Figure 5.18 shows the CDT results of the agents based on the variations of the window size and detection threshold. For Datalab, Eagle's Wing, and Pig Shooter, the pma method with window size 10 and 1.5 standard deviation detection threshold produced the highest CDT while for Random agent, the sma method with window size 5 and detection threshold 0.2 provided the highest CDT. The method that produces the highest CDT is used as the novelty detection method of the respective agent and the results achieved using that method are presented in the main paper and used in the rest of the appendix.

Novelty detection performance is measured using the measures CDT and DD. Figures 5.19-5.22 present the heat maps of CDT (higher the better) and DD (lower the better)

for each agent. Figures 5.23 and 5.24 present the summary of detection performance per novelty (NPS_{CDT}) and Figures 5.25 and 5.26 shows the summary per scenario (SPN_{CDT}). As it can be seen from the four summary figures and in comparison of agent heatmaps with humans, it can be seen that agents have a large room for improvement in novelty detection.

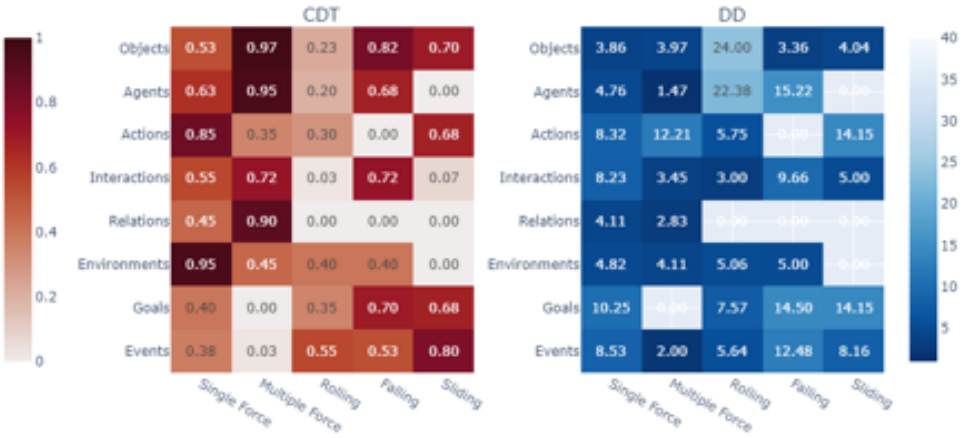


Figure 5.19: The CDT and DD results of the Datalab.

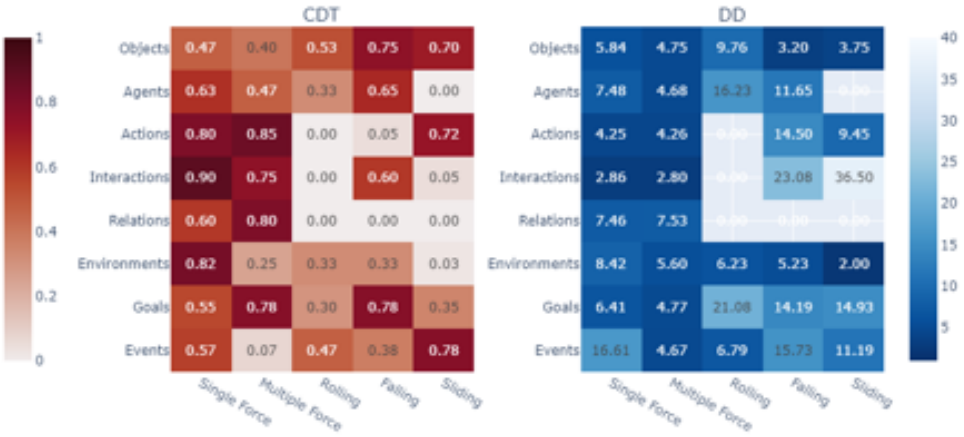


Figure 5.20: The CDT and DD results of the Eagle's Wing.

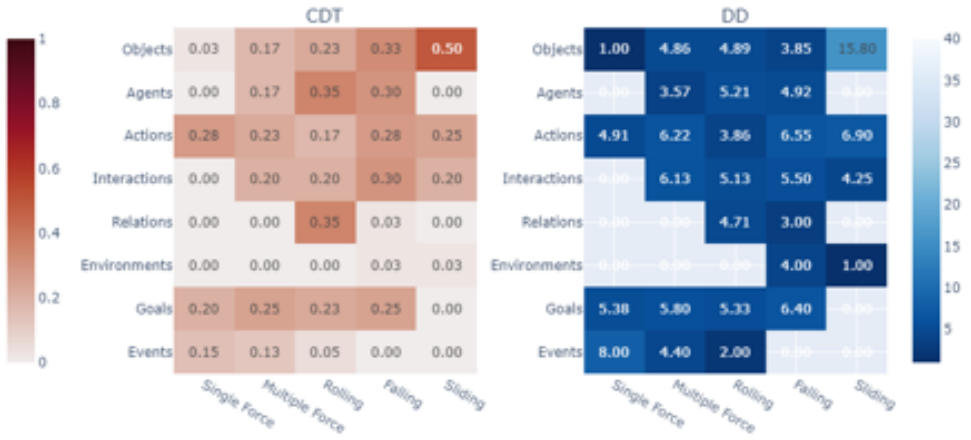


Figure 5.21: The CDT and DD results of the Pig Shooter.

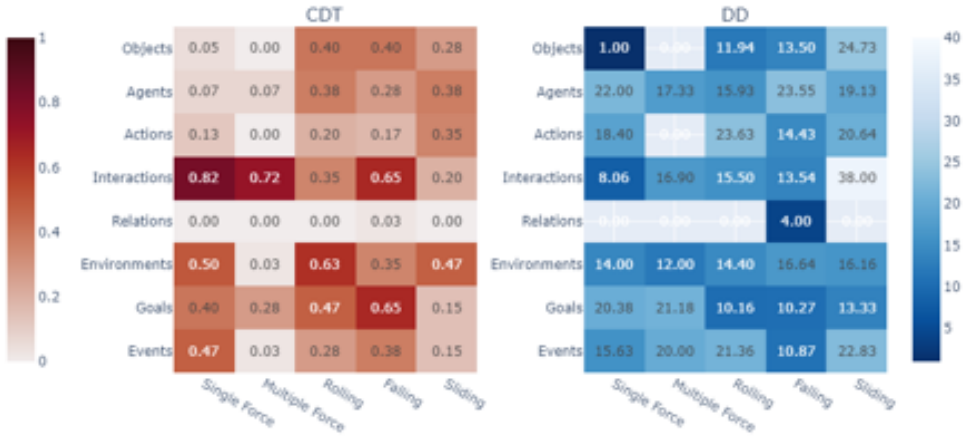


Figure 5.22: The CDT and DD results of the Random agent.

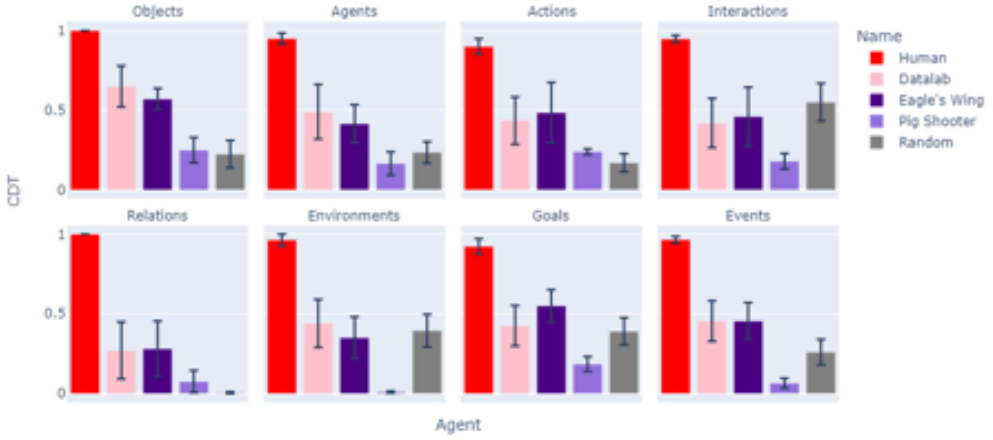


Figure 5.23: NPS_{CDT} of the agents for the eight novelties.

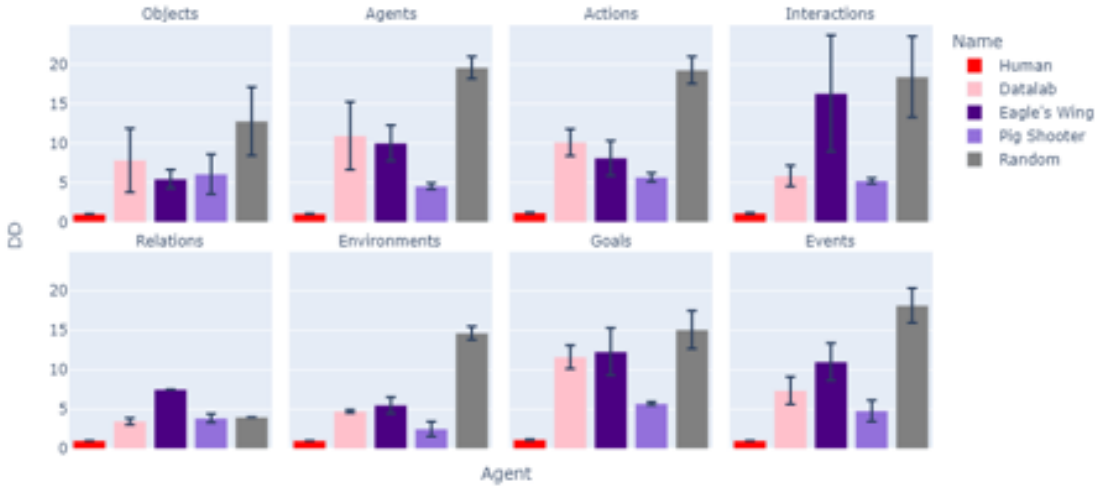


Figure 5.24: NPS_{DD} of the agents for the eight novelties.

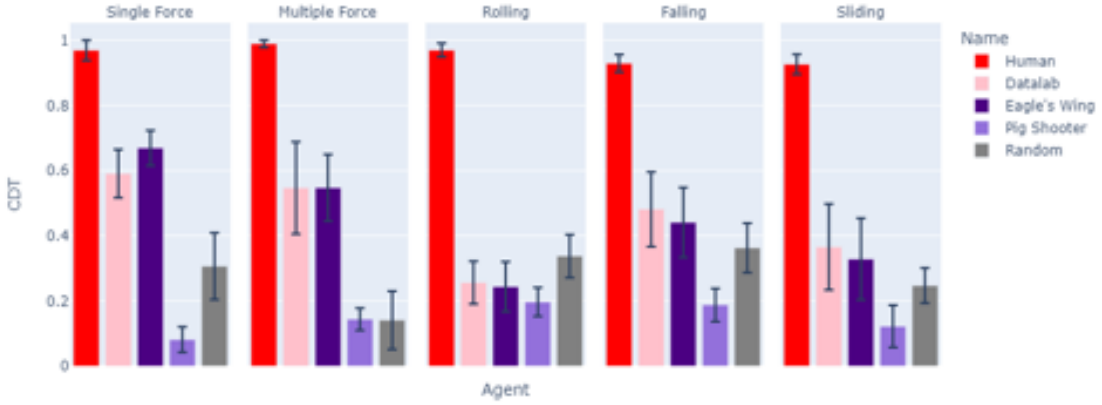


Figure 5.25: SPN_{CDT} of the agents for the five scenarios.

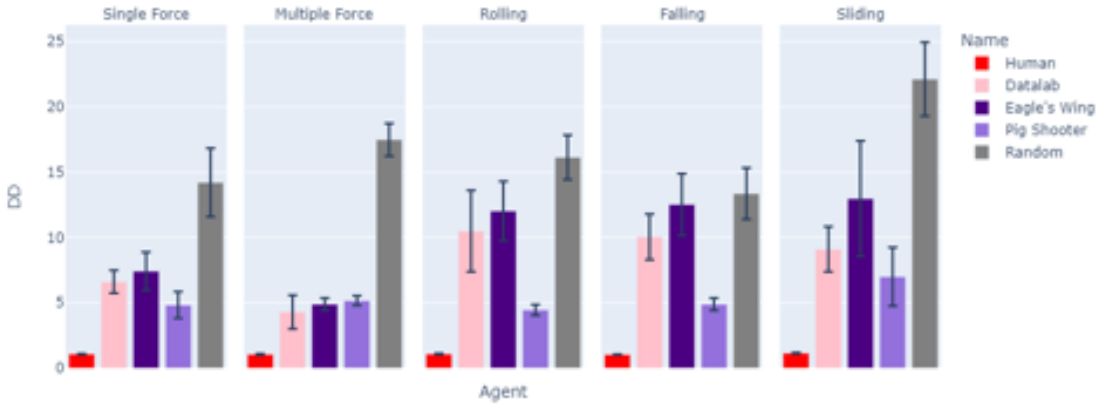


Figure 5.26: SPN_{DD} of the agents for the five scenarios.

5.7.5 Novelty Adaptation Performance

Figure 5.27: The pass rate of the agents in all novelty-scenarios. The x-axis represents the index of the task in trials. Indexes -40 to -1 are for normal tasks and 0 to 40 are for novel tasks. The y-axis represents the pass rate. **DQN Adapt and Relational Adapt are the agents with NAPPING, which is discussed in Chapter 6.**

The adaptation curves of each novelty-scenario are presented in Figure 5.27. The AP and AUS values derived from the adaptation curves for each agent are shown in the heat maps in Figures 5.28 - to 5.38. The AUS values are calculated from the performance of the agent in the last 50% of the novel tasks (m=20). The adaptation results per novelty are shown in Figures 5.39 and 5.40 while Figures 5.41 and 5.42 show the adaptation results per scenario performance. It can be seen that agents DQN Adapt and DQN Relational could reach human AP and AUS results in some of the novelty-scenarios.

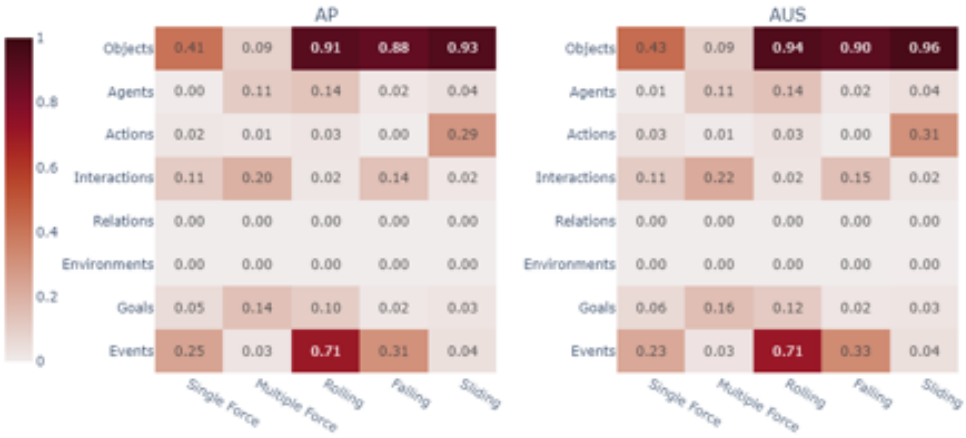


Figure 5.28: The AP and AUS results of the DQN Offline.

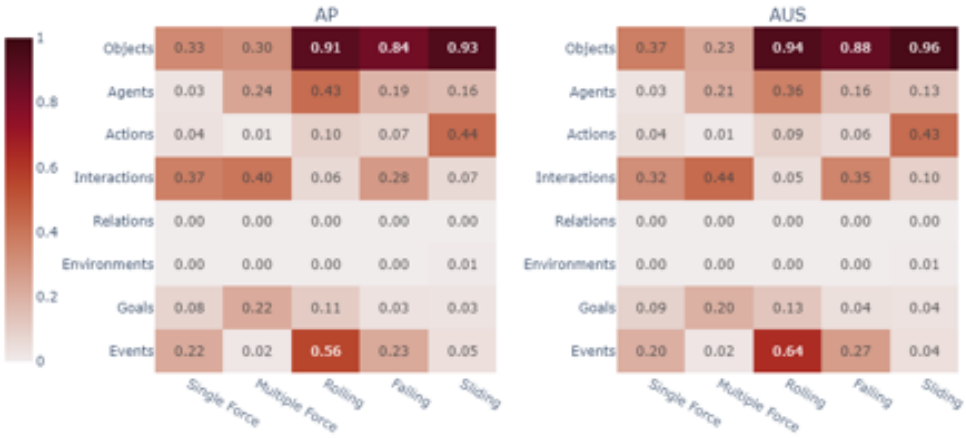


Figure 5.29: The AP and AUS results of the DQN Online.

However, looking at the adaptation plots, within the same number of tasks that were given to humans, those two agents have not reached the performance the humans could achieve, showing that there is room for improvement in terms of adaptation efficiency.

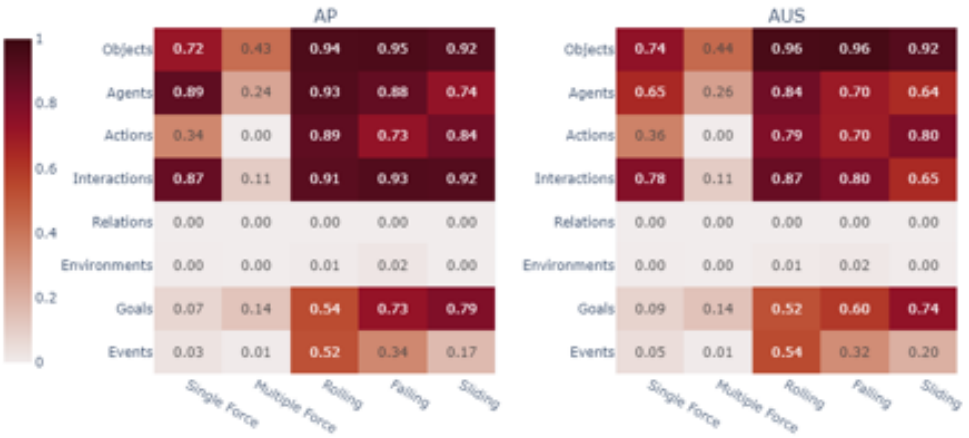


Figure 5.30: The AP and AUS results of the DQN Adapt.

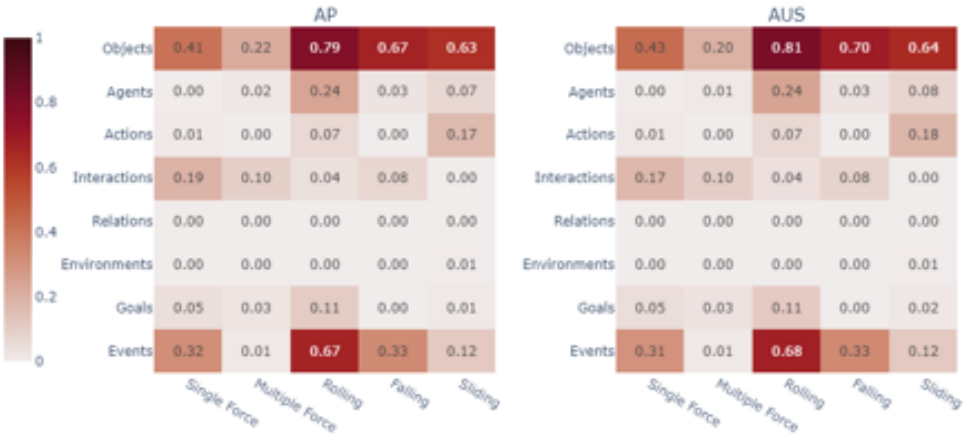


Figure 5.31: The AP and AUS results of the Relational Offline.

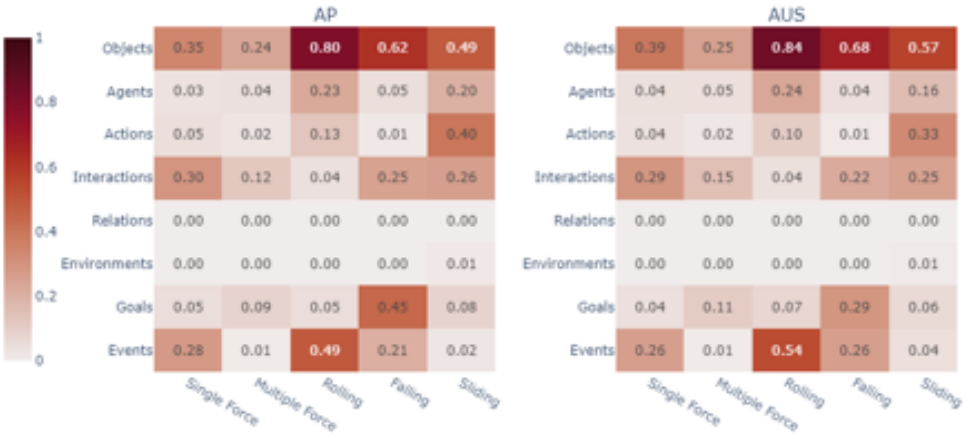


Figure 5.32: The AP and AUS results of the Relational Online.

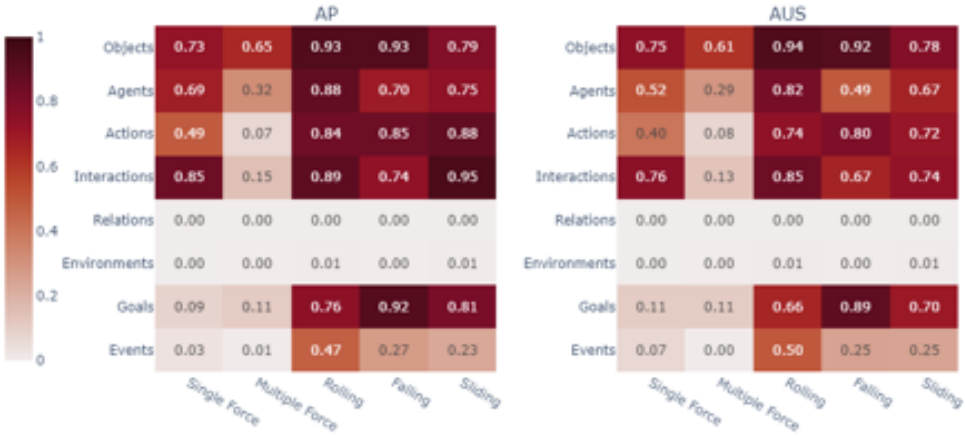


Figure 5.33: The AP and AUS results of the Relational Adapt.

Figure 5.34: The AP and AUS results of the Naive Adapt.

Figure 5.35: The AP and AUS results of the Datalab.

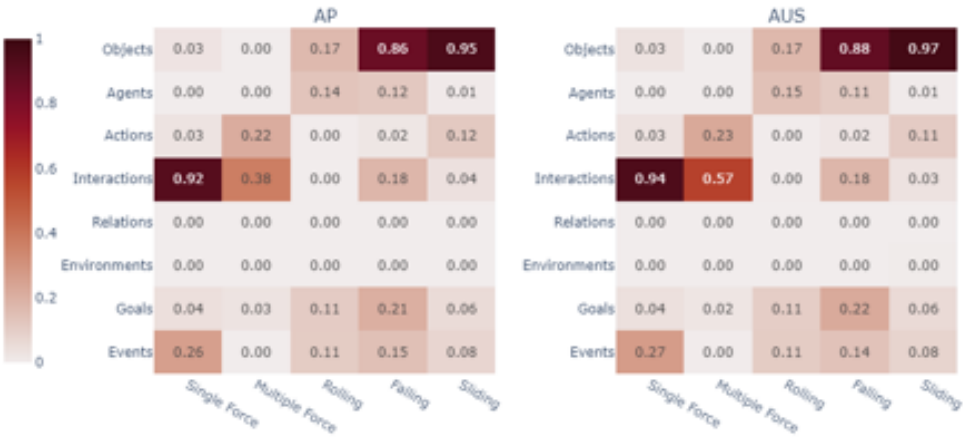


Figure 5.36: The AP and AUS results of the Eagle's Wing.

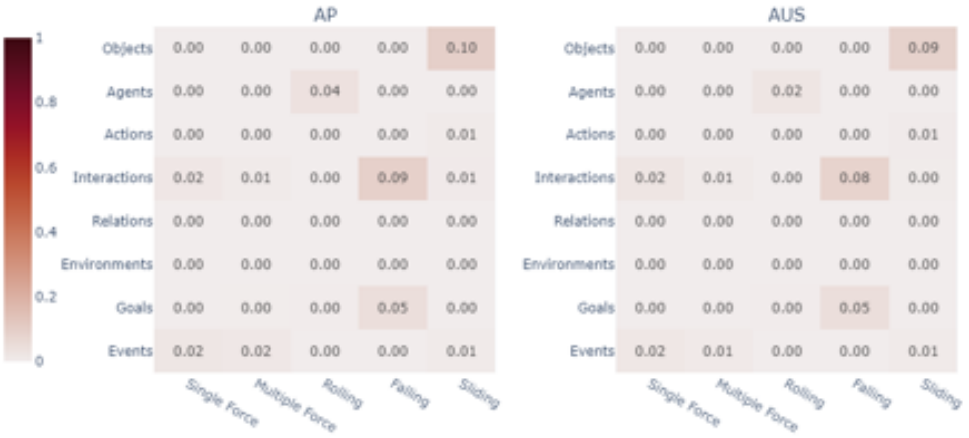


Figure 5.37: The AP and AUS results of the Pig shooter.

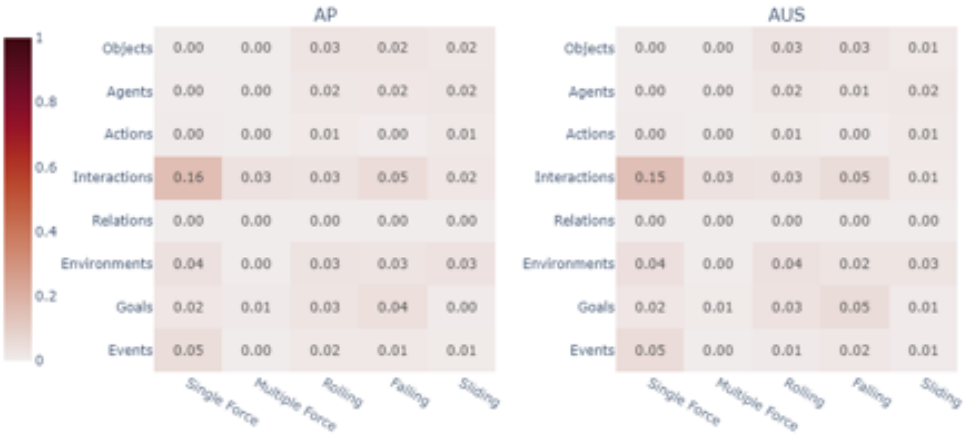


Figure 5.38: The AP and AUS results of the Random agent.

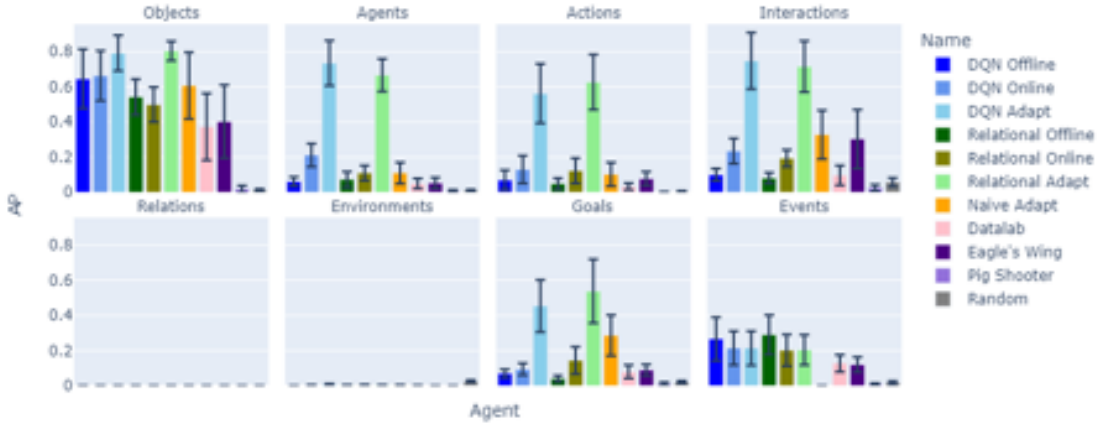


Figure 5.39: NPS_{AP} of the agents for the eight novelties.

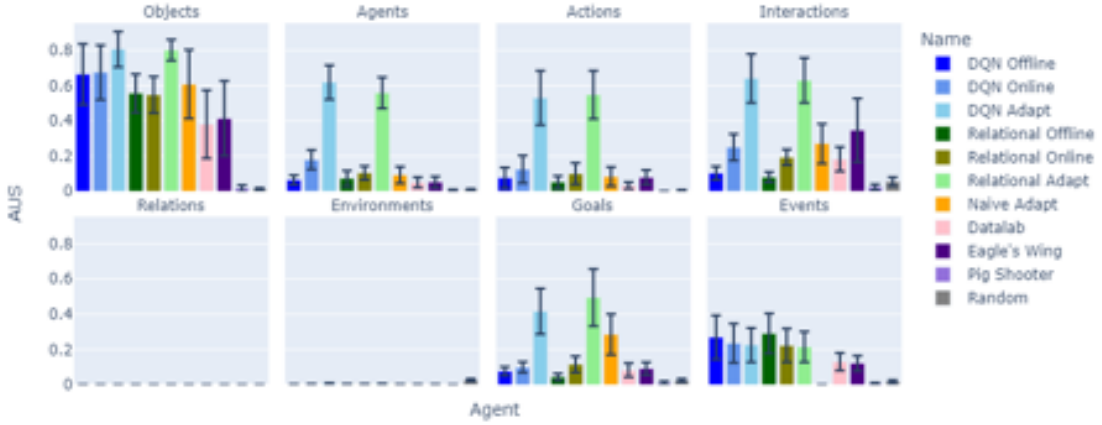


Figure 5.40: NPS_{AUS} of the agents for the eight novelties.

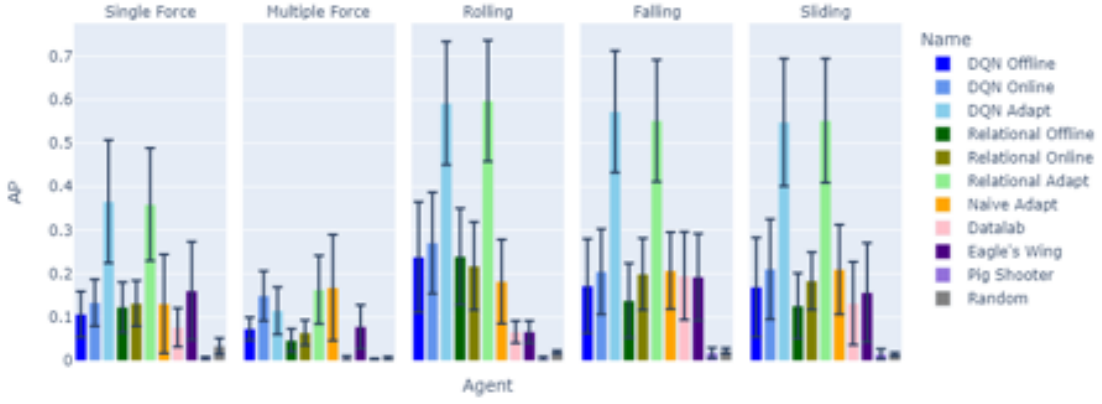


Figure 5.41: SPN_{AP} of the agents for the five scenarios.

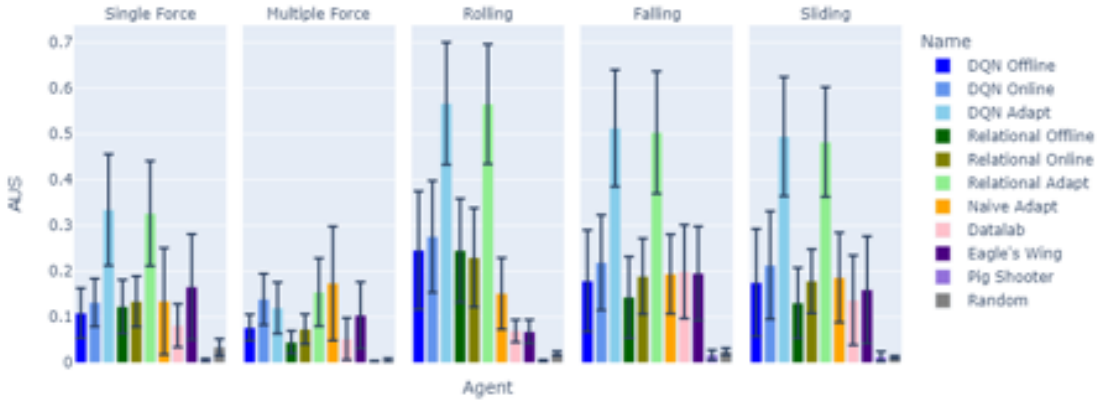


Figure 5.42: SPN_{AUS} of the agents for the five scenarios.

5.7.6 Human Detection vs Adaptation

In the previous sections, we have discussed humans' detection and adaptation performance and shown that humans' performance is far better than the agents' performance. This section analyses the relationship between novelty detection and adaptation performance of humans.

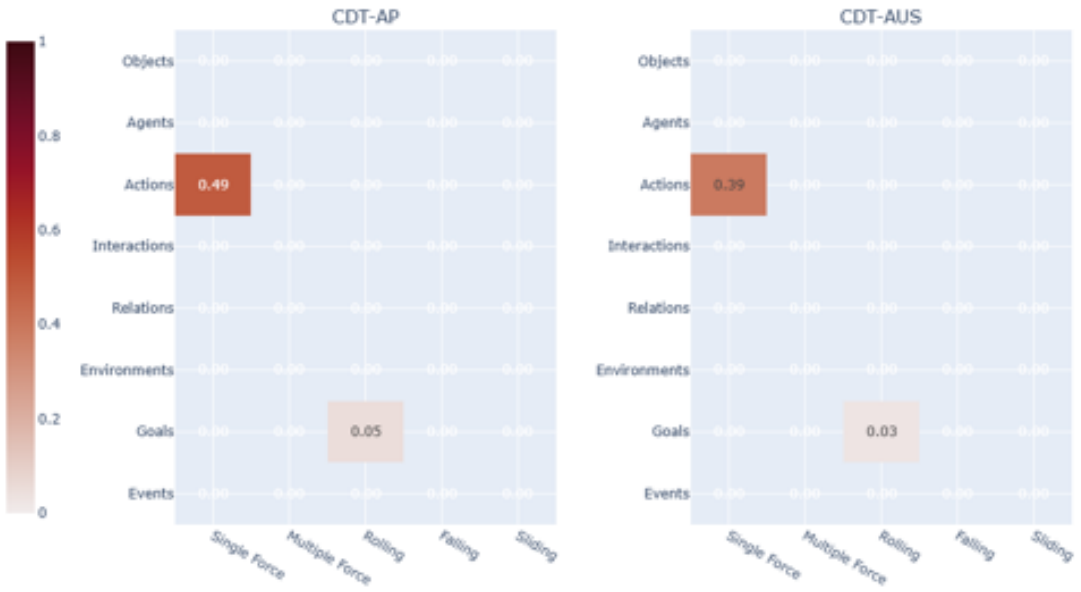


Figure 5.43: The p-value of the Mann-Whitney test for adaptation performance on the CDT=1 group and CDT=0 group

Figure 5.43 shows the p-value of the Mann-Whitney test conducted to test if there is a difference in adaptation performance based on whether the person correctly detected the novelty or not (CDT = 1 or CDT = 0). For CDT = 0 group, we have only used the data

from the players who did not detect novelty (excluded the data of the players who did wrong defections). The null hypothesis is, the adaptation performance of the two groups is equal. As Figure 5.43 indicates, almost all the entries are not defined as all human players correctly detected novelty and hence there are no players who did not detect the novelty in those novelty-scenarios. For the actions-single force novelty-scenario, the p-values in both AP and AUS exceed the 5% level of significance we consider. i.e., we do not have enough evidence to reject the null hypothesis, meaning that irrespective of the novelty detection, players performed similarly. The novelty here is the increased upward force of Air Turbulence, which is not visually detectable unless interacted. Some of the human players did not detect this novelty but they managed to solve the task as there is only a subtle change in the action from normal tasks to novel tasks. On the other hand, in the novelty-scenario goals-rolling, there is a significant difference in adaptation performance based on the CDT. In this novelty-scenario, the action has to be considerably adjusted when moving from normal to novel tasks, thus if the novelty is not detected, it would be difficult to adapt.

Figure 5.44: The Spearman's rank correlation for adaptation performance and detection delay

Figure 5.44 shows the Spearman's rank correlation coefficient between detection delay and adaptation performance (AP and AUS). For this calculation, if the person did not detect novelty in the trial, we have assumed the detection delay to be five, which is the maximum number of novel instances in a trial + 1. As can be seen from the figures, the correlation coefficient is undefined for some novelty-scenarios. Those are the cases where all the human players detected the novelty in the first novel task (DD = 1). As shown in the DD-AP heatmap, there are five moderately strong negative correlations (between -0.6 and -0.7) and they are all statistically significant at 5% level of significance.

This indicates that, for those novelty-scenarios, the more tasks it takes to detect, the lesser the asymptotic adaptation performance. According to DD-AUS heatmap, two novelty-scenarios have moderate positive correlations (0.41 and 0.52). However, they are not statistically significant at 5% level of significance. Only the moderately strong negative correlation value at novelty-scenario interactions-sliding is significant at 5% level of significance.

Chapter 6

NAPPING: Rapid Open-World Adaptation by Adaptation Principles Learning

This chapter is curated from "Rapid Open-World Adaptation by Adaptation Principles Learning" [183].

6.1 Introduction

Human cognition excels in adapting to diverse and dynamic environments, a capability still beyond the reach of current deep reinforcement learning (DRL) algorithms. Despite significant achievements across various domains [10, 169, 150, 94], DRL's performance often suffers when environments deviate slightly from training conditions [181].

For instance, consider a scenario where household robots are responsible for assisting with daily chores in a dynamic home environment. Imagine a situation where a new microwave or washing machine is introduced into the household. These additions present sudden and persistent changes to the robot's operating environment, requiring it to quickly adapt its routines and interactions to accommodate the new appliances. Humans can readily adapt to the presence of these new items, quickly integrating them into their daily routines. This adaptability is a crucial difference between human cognition and current deep reinforcement learning methodologies.

To address this challenge, we propose **N**ovelty **A**daptation **P**rinci**p**les Learn**ing** (**NAPPING**), an innovative method aimed at enhancing the adaptability of Deep Reinforcement Learning (DRL) agents in environments with novelty introduced (Figure 6.1).

In essence, NAPPING works together with an underperforming deep reinforcement learning baseline agent, utilizing its actions as a basis for improvement. By analyzing the

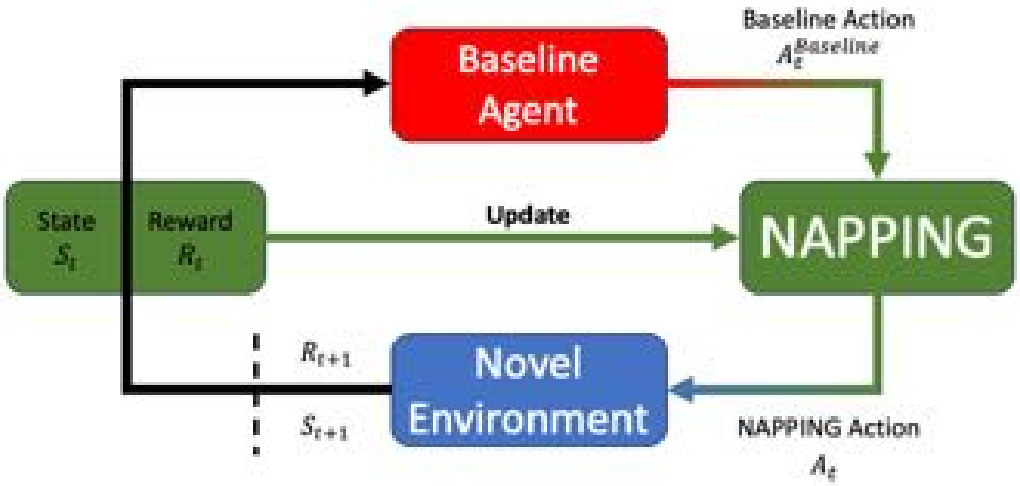


Figure 6.1: High-level NAPPING Framework

shortcomings of the baseline agent, NAPPING identifies problematic regions within the state space embedded in the baseline agent. Subsequently, it systematically refines actions specifically within these identified problematic regions. Leveraging the representations of the embedded state, NAPPING formulates adaptation principles tailored to address the challenges posed by specific environmental novelties. This approach facilitates the seamless integration of adaptive behaviours, ensuring that the baseline policy remains effective in unaffected regions.

We demonstrate the efficiency and effectiveness of our proposed method in action domains. We evaluate NAPPING in four different environments (Section 6.5). There are two popular control domains CartPole and MountainCar, one path-finding domain CrossRoad, and one physical reasoning domain AngryBirds, using the Science Birds Novelty test-bed [185]. The performance of our method is compared to the online learning version and the fine-tuning version of the baseline agents in CartPole, MountainCar and CrossRoad and is compared to two state-of-the-art novelty adaptation agents in AngryBirds. For each domain, we create novel situations by varying extensively the physical and spatial parameters of each environment except for the AngryBirds domain, where we evaluate our method on the provided novelties in the 2022 AIBirds Competition Novelty Track [2]. The evaluation results suggest that while the baselines model fails on almost all novelties, our method rapidly and effectively reacts to novel situations, accommodating many novelties in a few episodes.

Moreover, we present a case study and discuss why NAPPING outperforms other approaches, such as retaining the network, using the domain of cart-pole in Section 6.6. We

explain empirically the superior performance of NAPPING over standard online learning methods by visualizing the evolution of embedding spaces for both a DQN agent with online learning and with NAPPING. Our results demonstrate that while most of the regions in the embedding spaces of the pre-trained agent remain unchanged even after the agent solves the novel environment, NAPPING can discern and adjust actions accordingly, facilitating rapid adaptation. In contrast, standard online learning exhibits drastic oscillations in the updates of the embedding space. We also discuss the limitations of NAPPING in Section 6.6 as well.

6.2 Related Work

Traditional Deep Reinforcement Learning (DRL) approaches typically handle novelties as gradual changes in environments [83, 127, 40]. However, our focus in this work is on addressing sudden, long-term changes in the environment. While meta-reinforcement learning enables AI agents to rapidly acquire new skills by learning from prior experiences, conventional meta-reinforcement learning algorithms require training on multiple distinct tasks to achieve optimal weight initialization or hyperparameter settings before tackling new tasks [187]. In contrast, our approach centres on scenarios where agents are trained on a single environment specification, with existing baseline agents showing acceptable performance in pre-novelty conditions [90].

Transfer reinforcement learning involves selecting appropriate source tasks, understanding their relationship to the target task, and transferring knowledge accordingly [162]. However, we do not assume knowledge of the target environment or task, making standard transfer learning techniques challenging to apply. A conceptually similar approach to our proposed method is transfer learning via policy reuse, where policies learned from source tasks are repurposed to construct policies for target tasks [191]. Policy Reuse aims at speeding up the learning process of a new task by using past policies that solve different similar tasks to bias the exploration process. In contrast, we partition the trained policy into multiple regions, reusing effective regions while relearning ineffective ones for the same tasks with novelties.

Concept drift, as described in Lu et al. [104], refers to the unpredictable shifts in the distribution of streaming data over time, including research of methodologies for drift detection, understanding, and adaptation. One common reaction to concept drift involves retraining models with the most recent data, facilitated by explicit drift detectors. Paired Learners [11], following this approach, employ stable and reactive learners, swapping them when a new concept is detected based on their classification performances. While straightforward, adopting a window-based strategy requires a balance between window size, impacting data representation and model training. ADWIN [28], an algorithm addressing this issue, dynamically adjusts window sizes based on change rates within sub-windows. Nonetheless, it's worth noting, as highlighted by Langley [90], that while concept drift literature often focuses on classification tasks, this paper diverges by considering action domains within a Markov decision process framework.

More comparative analyses between open-world learning and other related paradigms, including out-of-distribution detection, anomaly detection, novelty detection, novelty discovery, meta-learning, open-set recognition, incremental learning, zero-shot incremental learning, representation incremental learning, class and representation incremental learning, and zero-shot learning open world learning, and many other have been conducted [90, 72, 64, 120, 20].

6.2.1 Open-world Learning

Open-world learning has emerged as a significant area of research recently, with novelty adaptation being a key task. Initially proposed by Bendale [26], open-world learning has been studied from various theoretical perspectives, including formal problem definitions [90, 91, 72], typologies of novelties [90, 31, 116], and agent frameworks and architectures [90, 91, 116, 101, 120, 34], enabling agents to operate effectively in environments with unexpected novelties.

Various approaches have emerged to tackle open-world learning challenges, with some emphasizing novelty detection [110, 32, 99] and others focusing on novelty adaptation [64, 88, 160, 86, 122]. For instance, RMA (Rapid Motor Adaptation) [88] enables legged robots to adapt to varying terrains, payloads, and wear and tear. While akin to our approach, RMA requires training the base policy in a simulator with diverse environmental parameters, addressing known unknowns. Conversely, our NAPPING framework handles unknown unknowns, which can be captured by the sensors of the agent, without presuming the types of novel situations it may encounter.

HYDRA [86, 160] and OpenMIND [122] are planning-centric agents capable of detecting and adapting to novelties using a range of AI techniques, including automated planning and anomaly detection. Similarly, RAPid-Learn [64] employs a hybrid planning and learning approach grounded in domain knowledge, albeit requiring an explicit understanding of environmental structures, which our method does not assume.

6.2.2 Adaptive Partitioning for Reinforcement Learning

Adaptive partitioning in reinforcement learning represents a groundbreaking solution to the memory constraints faced in real-time control problems [154, 155, 153]. The driving force behind this research is the exploration of reinforcement learning (RL) techniques for resolving critical computing system challenges like memory management, resource allocation, and load balancing. In these domains, optimal control policies must be learned directly on the chip to meet strict latency requirements, imposing significant memory limitations. To address this hurdle, adaptive partitioning offers a pathway to more efficiently discretize problem spaces, thereby mitigating memory demands. This innovative approach shows promise in tackling various control problems [7]. However, our method differs from these existing approaches in two key aspects. Firstly, instead of learning a new policy for each region, NAPPING focuses on adjusting actions only in regions where the baseline policy fails. Secondly, rather than splitting regions based

on the number of visits, we use regret of the chosen action by the agent for region partitioning.

Another line of work, leveraging Partially Observable Markov Decision Process (POMDP) frameworks, has employed methodologies like Voronoi Tessellation to split continuous action spaces. These approaches often assume the existence of a generator - a model of the environment that can approximate the environment's transition kernel, providing the next state, observation, and reward given the current state and action [70, 84, 100]. Our method does not assume such a generator. It is essential to distinguish between the environment's generator, which is an approximation of the environment and the generators, which we refer to in Section 6.4 used to construct Voronoi regions.

6.3 Problem Definition

We adopt the standard framework used in reinforcement learning (RL), where agents interact with an environment modelled as a Markov Decision Process (MDP), denoted by $E = \langle S, A, Pr, R \rangle$. Here, S represents the set of environment states, A is the set of available actions for the agent, $Pr(s_{t+1}|a_t, s_t)$ is the transition probability function, which determines the distribution of the next state given the current state and action, and R is the reward function providing immediate rewards for state-action transitions. At each time step t within an episode, the agent observes a state $s_t \in S$, selects an action $a_t \in A$, and receives a reward r_{t+1} upon transitioning to state s_{t+1}, continuing this process until the episode terminates. In episodic tasks, the goal of the agent is to find a policy π that maximizes the expected total discounted reward $G_t = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \middle| S_t = s\right]$, where the discount factor $\gamma \in (0, 1)$ and T is the final time step in an episode.

Motivated by the *Theory of Environment Change* [90], we define novelties as transformations of the underlying environment E. Formally, a novelty is represented as a transformation function ϕ, resulting in a post-novelty environment denoted as $\phi(E) = \langle S_\phi, A_\phi, P_\phi, R_\phi \rangle$. This transformation may alter various aspects of the environment, including the state space, action space, transition probabilities, and/or reward function.

In this work, we focus on only the unanimous novelties—those that both are detectable by the agent and impact the agent's performance. We specifically focus on novelties where the action space of the agent remains unchanged, and novelties can be managed by modifying the policy $\pi(s_t)$ of the DRL agent. The reason for this choice is that the actions available to an autonomous agent are typically fixed in real-world scenarios. For instance, in self-driving cars, the actions available to the vehicle are typically limited to maneuvers such as accelerating, braking, and turning. It is unlikely that the vehicle can suddenly fly. Therefore, adjusting novelties would require the agent to modify the sequence or parameters of the action to regain its performance in pre-novelty environments.

Following the same setup for the open-world learning problem defined in [90], we also assume there exists an autonomous agent that can reach acceptable performance in the pre-novelty MDP – the environment before the transformation.

The problem addressed in our work stands distinct from both few-shot learning (FSL) and incremental learning (IL) paradigms. Few-shot learning involves training a model on a small data set with limited examples per class, enabling it to generalize to new classes with only a few examples each. In contrast, the problem we tackle does not involve constantly learning new classes, as novelty does not consistently appear. Our objective is to ensure the system's functionality even in situations unaffected by novelty. Therefore, our approach aims to adapt network performance with minimal intervention by adjusting labels for select samples while preserving knowledge for unmodified ones. Importantly, this adaptation is achieved without the need to retrain the model for new classes. Incremental learning, also referred to as continuous learning, lifelong learning, or never-ending learning, has gained substantial attention in the field of machine learning. It is primarily employed to address the issue of catastrophic forgetting, where performance on previously learned tasks sharply deteriorates after learning new tasks. However, unlike class-incremental learning (Class-IL), where ample samples are available for each incremental category [163], our problem requires methods to adapt to novelties as fast as possible. Furthermore, while incremental learning contains three major scenarios [166] —Task-IL, Domain-IL, and Class-IL—we concentrate on adjusting the input-output mapping within the same domain, using the same task and class.

More comparative analyses between open-world learning and related paradigms have been conducted [90, 72, 64, 120, 20].

6.4 NAPPING - Novelty Adaptation Principles Learning

In this paper, we address the challenge of adapting deep reinforcement learning (DRL) agents to novel environments by introducing the NAPPING framework. We consider a deterministic Markov Decision Process (MDP) with discrete action space A and either discrete or continuous state space. In scenarios involving continuous action spaces, we discretize the spaces. Our approach utilises a pre-trained DRL agent operating that has a state-embedded space MS residing in a metric space defined by distance function d. Extensions to continuous action spaces and non-deterministic environments are discussed in the Future Work Section 6.7.

We begin by discussing our motivation regarding the effectiveness of the NAPPING algorithm in adapting baseline agents to novel environments. We believe that, instead of retraining a trained network to adapt to novelties—essentially using gradient descent and making global changes—dynamically adjusting the agent's behaviour directly in only the regions of the embedding space where the agent fails would be more effective for dealing with novel situations, which are not entirely new tasks or domains.

The embedded space, also known as the latent feature space, is a space where state vectors are arranged such that similar states requiring the same actions are grouped together spatially. Consequently, when the agent detects a novel state and adaptation is required, its policy is likely to adapt not only to the novel state but also to similar states

in the surrounding areas of the embedded space. For example, in scenarios where the agent's input consists of images, two conceptually similar states, which might exhibit distant representations in pixel space, are close to each other in the embedded space.

To capture the underlying structure effectively, we propose partitioning the embedding space using Voronoi tessellation, which divides the space into regions based on proximity to a set of points. , where each point is referred to as a **generator** and is a point in the embedded space. Therefore, these partitions implicitly contains conceptually similar states. . The partitioning process undergoes iterative updates and refinement as the agent interacts with the environment, gathering more information about the partitions in the embedding space to determine where the baseline agent still functions effectively and where it does not. In instances where the baseline fails to perform satisfactorily, NAPPING replaces the action in the corresponding region with an adaptation action.

We refer to the mapping from the partition in the embedding space P where the baseline agent selects action a^i to the adaptation action a^j as an **adaptation principle**. This concept captures the idea that in situations requiring adaptation, where humans need to adapt, humans can abstract the situation and adjust their previous actions to new ones. For example, if we realize that driving too fast (a^i) over a speed bump (P) is not a good idea, we will then slow down (a^j) next time when we encounter a speed bump.

To validate our algorithm we evaluate the NAPPING algorithm across action domains with different baseline agents to showcase its practical performance in Section 6.5. Moreover, we present a case study demonstrating why NAPPING outperforms standard deep learning approaches in the domain of cart-pole in Section 6.6.

It is also worth noting that NAPPING does not only handle novelties persisting in the environment once introduced. The adaptation action associated with a partition in the embedded space can be mapped back to the baseline agent's action if a novelty detection component informs NAPPING that the novelty is no longer present.

In short, our algorithm addresses novelties by identifying and adjusting actions in partitions where the DRL agent exhibits unacceptable performance, termed **adaptation principles**. These principles guide the agent to choose more effective actions in specific contexts.

6.4.1 NAPPING Algorithm

Overall, NAPPING consists of two conceptual processes: a) partitioning the latent space into Voronoi regions defined by a set of generator points and b) assigning an action to each region, which defines the updated policy. The partitioning process begins with the initialization of two sets: $\mathcal{G} = \{G_{a^1}, ..., G_{a^{\text{num}(A)}}\}$ to contain the generator points of each partition in $\mathcal{P}$, and $\mathcal{P} = \{P_{a^1}, ..., P_{a^{\text{num}(A)}}\}$ to store the partitions for num(A) different actions select by the baseline agent. Each P_{a^i} comprises non-overlapping partition denoted as $\{P^k_{a^i}\}^j_{k=1}$, representing distinct regions in the embedded space MS_{a^i}. Here, MS_{a^i} denotes the embedded space where the baseline agent selects action a^i. The union

of all subsets $\bigcup_{k=1}^{j} P_{a^i}^k$ covers the entirety of space $MS_{a^i} = P_{a^i}$. Likewise, G_{a^i} may contain multiple sets, $G_{a^i}^k$, where each $G_{a^i}^k$ corresponds to a partition $P_{a^i}^k$ formed by the Voronoi regions generated by $G_{a^i}^k$. These generator points are utilised to construct the Voronoi tessellation, with points in the same $G_{a^i}^k$ set indicating that partition $P_{a^i}^k$ is a union of all Voronoi regions generated by $G_{a^i}^k$.

For the assignment of the action to the partitions, , we introduce the set $\mathcal{AP} = \{AP_{a^1}, ..., AP_{a^{num(A)}}\}$ to hold the adaptation principles. Each AP_{a^i} is initialized as $\{AP_{a^i}^1\}$, where each $AP_{a^i}^k$ is defined over the kth partition when action a^i is selected by the baseline as $AP_{a^i}^k : P_{a^i}^k \rightarrow A(s)$. Therefore, each adaptation principles is a function defined over a partition and outputs an action. More specifically, it map states within the partition to adaptation actions in the available action space $A(s)$.

The NAPPING algorithm starts by receiving the baseline policy π^{baseline}, the number of episodes K, and the regret threshold thre as input parameters. It then initializes sets $\mathcal{G}$, $\mathcal{P}$, $\mathcal{AP}$, and a set H to store the triplets of $(ms_t, a_t, \text{Regret})$ history. The $P_{a^i}^1$ and $AP_{a^i}^1$ cover the entire space and return the baseline action. Also, $AP_{a^i}^1$ is initialised to be a **Closed** principle (discussed in Sec 6.4.3). The algorithm iterates over each episode and time step, determining the action a_t using the corresponding adaptation principles based on the embedded state ms_t through the **SelectAction** procedure.

The key questions for NAPPING are: (a) how to select the generator points, and (b) how to assign the actions. From lines 2 to 18 in the Update procedure, we discuss how generator points are selected and how new partitions are formed. From lines 19 to 26, we discuss how actions are assigned to the partitions. In short, after receiving the reward r_t and transitioning to a new state s_{t+1}, NAPPING evaluates the action regret. If the regret exceeds a certain threshold and the s_{t+1} falls in a **Closed** adaptation principle, it creates new partitions. Otherwise, it continues to try other actions. The algorithm terminates after the K episode. If there are still **Open** adaptation principles in AP, they are converted to **Closed** ones with the corresponding action to be the baseline action, indicating regions where adaptation is deemed unnecessary or impossible.

Algorithm 1 NAPPING

procedure NAPPING($\pi^{baseline}$, num(A), K, thre)
  Initialise the generators set $\mathcal{G} \leftarrow \{G_{a^1} \leftarrow \{\}, ..., G_{a^{\text{num}(A)}} \leftarrow \{\}\}$
  Initialise the partitions set $\mathcal{P} \leftarrow \{P_{a^1}, ..., P_{a^{\text{num}(A)}}\}$, where $P_{a^i} \leftarrow \{P^1_{a^i}\}$. The $P^1_{a^i}$ containing a single ball covering the whole embedding space where the baseline agent chooses action a^i
  Initialise the adaptation principles set $\mathcal{AP} \leftarrow \{AP_{a^1}, ..., AP_{a^{\text{num}(A)}}\}$, where $AP_{a^i} \leftarrow \{AP^1_{a^i}\}$ and $AP^1_{a^i}(ms) = a^i$ given $ms \in P^1_{a^i}$
  Initialise a set H to store the triplets of $(ms_t, a_t, \text{Regret})$ history
  for each episode $k \leftarrow 1, ..., K$ **do**
    for $t \leftarrow 1, ..., T$ **do**
      Receive state s_t
      ms_t = model-embedded state of state s_t
      $a_t^{\text{baseline}} = \pi^{\text{baseline}}(s_t)$
      $a_t \leftarrow$ **SelectAction**$(a_t^{\text{baseline}}, ms_t, H)$ ▷ Alg 2
      Receive reward r_t and transition to new state s_{t+1}
      Regret $\leftarrow$ **EvaluateAction**(s_t, a_t, r_t, s_{t+1}) ▷ Sec 6.4.2
      Update$(ms_t, a_t, \text{Regret}, \text{thre})$ ▷ Alg 3
      Add $(ms_t, a_t, \text{Regret})$ to H
    end for
  end for
  for $AP_a \in \mathcal{AP}$ **do**
    for $AP^k_a \in AP_a$ **do**
      if AP^k_a is **Open then**
        Set AP^k_a to **Closed** with $AP^k_a(ms) = a$
      end if
    end for
  end for
end procedure

6.4.2 Evaluate Action

In this section, we propose assessing action quality through regrets. Regrets can be represented by the difference between the cumulative reward obtained from a learned policy or the potential cumulative reward achievable by an optimal policy. For example, if we use the optimal policy action regret can be defined as:

$$Regret(s, a) = Q^*(s, a) - Q^\pi(s, a)$$

Here, $Q^*(s, a)$ represents the value of the optimal Q-value for action a in state s, and $Q^\pi(s, a)$ represents the Q-value of the action selected by the learned policy in state s.

Directly computing regret using optimal policy is often infeasible. Nonetheless, several practical approaches exist for approximating regret. These include the utilization of

heuristic methods [39], leveraging expert knowledge [137], employing function estimators [175, 102], and utilizing techniques to estimate an agent's performance deviation from the optimal action in hindsight [138]. While these approximations may not produce perfect regret values, they offer valuable insights into the performance of the learned policy.

In our study, we opt to employ heuristics for evaluating action regret. Heuristics provide a practical and accessible means of assessing action quality, especially in complex environments where obtaining an optimal policy is challenging. These heuristics are designed based on domain-specific knowledge and are tailored to capture the essential aspects of the task. The heuristic measures of regret can be formalized as regret with respect to an optimal policy in a simplified MDP. For example, in a navigation task, a simple heuristic could prioritize actions that lead the agent closer to the goal state.

Importantly, these heuristics focus solely on the task at hand, without any consideration for novelty. While this simplification may overlook certain aspects of the environment, it allows us to evaluate action regret in a consistent and interpretable manner across different domains.

In the corresponding evaluation section, we provide details of the specific heuristics used in our study.

6.4.3 Adaptation Principles and Partitions

In this section, we discuss the process of selecting actions from adaptation principles, the creation and refinement of partitions, and the update of adaptation principles.

We begin by introducing two fundamental concepts related to adaptation principles: **Closed** Adaptation Principles and **Open** Adaptation Principles. A **Closed** adaptation principle is identified when the ideal action is determined for the corresponding adaptation principle. Conversely, **Open** adaptation principles are those where NAPPING continues to seek a better action for adjustment. Intuitively, when the regret associated with an action surpasses a predefined threshold and the model-embedded state ms_t falls within a closed region, NAPPING infers that the existing partition is too coarse and refinement is necessary. Conversely, if ms_t falls within an open region, NAPPING records it to H as an unsuccessful action attempt.

Selecting Actions from Adaptation Principles: The process of selecting actions from adaptation principles involves two scenarios: closed and open adaptation principles. For closed adaptation principles, $AP^k_{a_t^{\text{baseline}}}$ is directly used to return the corresponding action to the principles. In the case of open adaptation principles, when receiving the model-embedded state ms_t, NAPPING examines the history H to identify the k nearest ms's and their corresponding actions, weighting them to output the most probable action. The intuition is to select the action most likely to work. If the action attempted in the proximity resulted in a high regret value, another action is considered.

Algorithm 2 Select Adaptation Principle Action

procedure SELECTACTION(a_t^{baseline}, ms_t, H)
  Find $AP^k_{a_t^{\text{baseline}}}$ such that $ms_t \in P^k_{a_t^{\text{baseline}}}$
  if $AP^k_{a_t^{baseline}}$ is **Closed then**
    return $AP^k_{a_t^{baseline}}(ms_t)$
  end if
  Initialize $WeightedRegret$ dictionary with all actions and an initial value of 0
  for each triplet $(ms, a, Regret)$ in k nearest states from H **do**
    $distance \leftarrow d(ms, ms_t)$
    $WeightedRegret[a] \leftarrow WeightedRegret[a] + \frac{Regret}{distance}$
  end for
  $a_t \leftarrow a$ with the lowest $WeightedRegret$ (Randomly select actions to break ties)
  return a_t
end procedure

Updating Partitions and Adaptation Principles: We employ the **Update** (Alg 3) process to refine partitions and improve adaptation principles iteratively. Initially, this process identifies the corresponding adaptation principle $AP^k_{a_t^{\text{baseline}}}$ and evaluates whether the regret exceeds a predefined threshold *thre*. If the regret surpasses this threshold, it signals an issue with the action selected at ms_t. In such case, if the associated adaptation principle is **Closed**, indicating adaptation was completed, a regret exceeding the threshold necessitates partition refinement. To achieve this, we initialise a new generator set $G^{k+1}_{a^{\text{baseline}}}$ containing ms_t, adding it to G^{baseline}_a. Consequently, a new partition $P^{k+1}_{a^{\text{baseline}}}$ is generated around ms_t and appended to $P_{a^{\text{baseline}}}$. Subsequently, an open adaptation principle is defined on $P^{k+1}_{a^{\text{baseline}}}$ (Fig 6.2.b). Conversely, if the regret surpasses the threshold and the adaptation principle $AP^K_{a^{\text{baseline}}}$ is **Open**, indicating ongoing learning, partition adjustment is unnecessary (Fig 6.2.c).

In cases where the regret is below the threshold, indicating a potentially effective action and the adaptation principle is **Closed**, ms_t is added to $G^k_{a^{\text{baseline}}}$ to expand the partition $P^k_{a^{\text{baseline}}}$. This expansion is guided by the assumption that the current adaptation principle remains effective for the surroundings of ms_t (Fig 6.2.a).

However, if the underlying adaptation principle is previously **Open**, partition adjustment is required. This adjustment involves adding the closest point where the action a_t yielded a regret exceeding the threshold ms_c to the generator point for the partition. Subsequently, a new closed adaptation principle is created with $G^{k+1}_{a^{\text{baseline}}} = \{ms_t\}$, and $P_{a^{\text{baseline}}}$ is updated using Voronoi Tessellation on $G_{a^{\text{baseline}}}$ (Fig 6.2.d). This ensures that the new adaptation principle $AP^{k+1}_{a^{\text{baseline}}}$ is defined on $P^{k+1}_{a^{\text{baseline}}}$ and outputs the action a_t. Adding the ms_c to the generators for $P^k_{a^{\text{baseline}}}$ ensures that only the region most likely to work with a_t is split from the partition $P^k_{a^{\text{baseline}}}$.

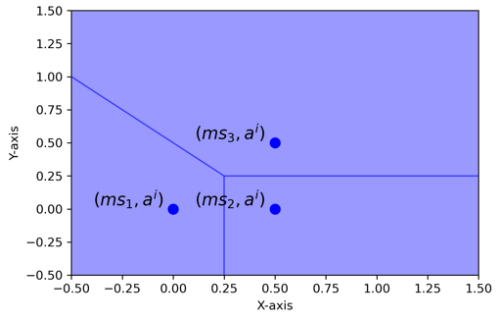
Algorithm 3 Update Adaptation Principles and Paritions

procedure UPDATE(ms_t, a_t, Regret, thre)
  Find $AP^k_{a^{\text{baseline}}_t}$ such that $ms_t \in P^k_{a^{\text{baseline}}_t}$
  if Regret $>$ thre **then**
    if $AP^k_{a^{\text{baseline}}}$ is **Closed then**
      $G^{k+1}_{a^{\text{baseline}}} \leftarrow \{ms_t\}$
      add $G^{k+1}_{a^{\text{baseline}}}$ to $G_{a^{\text{baseline}}}$
    end if
  else
    if $AP^k_{a^{\text{baseline}}}$ is **Closed then**
      Add ms_t to $G^k_{a^{\text{baseline}}}$
    else
      $ms_c \leftarrow$ the closest point in $P^k_{a^{\text{baseline}}}$ that a_t has a higher than threshold Regret
      Add ms_c to $G^k_{a^{\text{baseline}}}$
      $G^{k+1}_{a^{\text{baseline}}} \leftarrow \{ms_t\}$
      Add $G^{k+1}_{a^{\text{baseline}}}$ to $G_{a^{\text{baseline}}}$
    end if
  end if
  $P_{a^{\text{baseline}}} \leftarrow$ Update $P_{a^{\text{baseline}}}$ by using Vornoroi Tesslation on $G_{a^{\text{baseline}}}$
  if $P^{k+1}_{a^{\text{baseline}}}$ is created **then**
    if Regret $>$ thre **then**
      $AP^{k+1}_{a^{\text{baseline}}}$ is **Open**
    else
      $AP^{k+1}_{a^{\text{baseline}}}$ is **Closed** with $AP^{k+1}_{a^{\text{baseline}}}(ms) = a_t$
    end if
    Add $AP^{k+1}_{a^{\text{baseline}}}$ to $AP_{a^{\text{baseline}}}$
  end if
end procedure

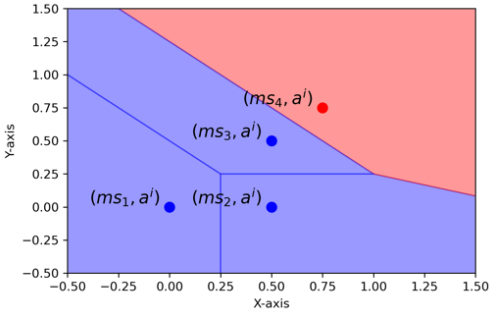
6.5 Evaluation Results in Action Domains

In this section, we proceed to evaluate the performance of our proposed method across four distinct action domains (refer to Fig. 6.3): CartPole [23], MountainCar [33, 119]), CrossRoad [125], and Angry Birds using Science Birds Novelty [185]. CartPole and MountainCar are two popular classic control domains, CrossRoad is a path-finding domain, and Angry Birds is a physics puzzle game.

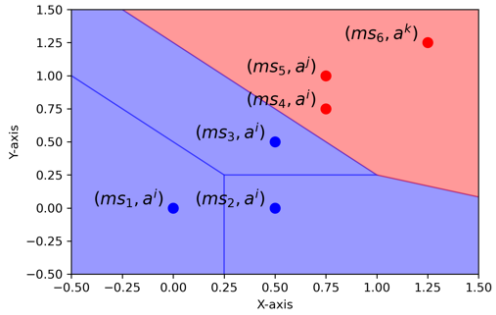
Evaluation Protocol: To evaluate our method, we adopt a standard open-world learning trial setting as discussed in Chapter 5 as well as [131] and employed in [86, 160, 64, 185]. Each trial comprises a sequence of episodes starting from the pre-novelty environment



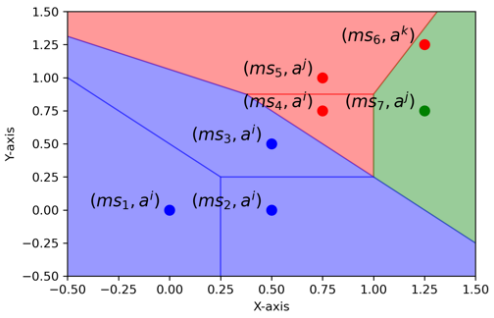
(a) The initial three ms all have lower than threshold regret. Therefore, generators are $G_{a^i} = \{G^1_{a^i}\}$, $G^1_{a^i} = \{ms_1, ms_2, ms_3\}$. $P_{a^i} = \{P^1_{a^i}\}$, $P^1_{a^i} = MS_{a^i}$ covers the blue region. $AP = \{AP^1_{a^i}\}$ where $AP^1_{a^i}$ is closed and returns action a^i. this scenario illustrates that the baseline agent performs well in the first three states ms_1, ms_2 and ms_3 when it uses action a_i. This corresponds to lines 9 to 10 in the procedure **Update**.



(b) The incoming (ms_4, a^i) has higher than threshold regret. Therefore, $G_{a^i} = \{G^1_{a^i}, G^2_{a^i}\}$ where $G^2_{a^i} = \{ms_4\}$. $P_{a^i} = \{P^1_{a^i}, P^2_{a^i}\}$, $P^1_{a^i}$ covers the blue region, and $P^2_{a^i}$ covers the red region. $\mathcal{AP} = \{AP^1_{a^i}, AP^2_{a^i}\}$ where $AP^2_{a^i}$ is open. This indicates that a new Voronoi cell is introduced due to the above-threshold regret state, but NAPPING has yet to determine an alternative action. NAPPING will need to explore other actions.

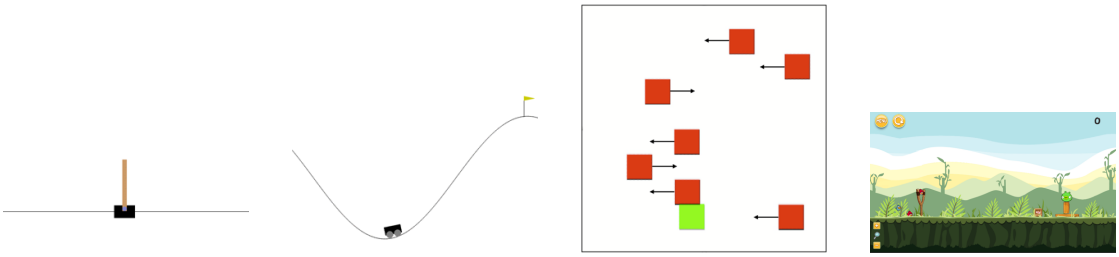


(c) Two additional states arrive, and the open adaptation principle $AP^2_{a_i}$ tries action a^j and a^k. However, both of the attempts result in above-threshold regrets. Therefore, no adjustments are necessary, and $AP^1_{a^i}$ remains open.



(d) The adaptation principle $AP^2_{a^i}$ selects the action a^j at ms_7, and receives a low than threshold regret. Therefore, we update G^i_a such that $G_{a^i} = \{G^1_{a^i}, G^2_{a^i}, G^3_{a^i}\}$, $G^2_{a^i} = \{ms_4, ms_5\}$, $G^3_{a^i} = \{ms_7\}$. The partition P_{a^i} is generated using the new set of generator points using Voronoi tessellation, resulting in the new partition $P_{a^i} = \{P^1_{a^i}, P^2_{a^i}, P^3_{a^i}\}$. where $P^3_{a^i}$ covers the green region. Also, $\mathcal{AP} = \{AP^1_{a^i}, AP^2_{a^i}, AP^3_{a^i}\}$ where $AP^2_{a^i}$ is still open and $AP^3_{a^i}$ is closed with $AP^3_a(ms) = a^j$.

Figure 6.2: An example of the Update Procedure a) adaptation principle is closed and regret is below the threshold b) adaptation principle is closed and regret is above the threshold c) adaptation principle is open and regret is above the threshold, and d) adaptation principle is open and regret is below the threshold.



(a) CartPole (b) MountainCar (c) CrossRoad (d) AngryBirds

Figure 6.3: Four different action domains are used in this work.

E. After a series of pre-novelty episodes, a novelty is introduced, transforming the environment to $\phi(E)$. Subsequently, all remaining episodes are drawn from $\phi(E)$. Each trial involves only one transformation applied to the pre-novelty environment. The novelty is assumed to be a sudden change to the environment, after which the agent must adapt its policy to recover as quickly as possible. Specifically, a trial consists of 80 episodes, with the first 40 episodes sampled from the pre-novelty environment and the subsequent 40 from the post-novelty environment. The novelty is introduced at episode 0, resulting in pre-novelty episodes indexed from -40 to -1. This trial setup is consistent across all domains, except for Angry Birds, where the number of pre-novelty episodes is randomized.

Novelty Detection: While novelty detection is crucial in open-world learning and remains an open research question, our focus in this paper primarily lies on novelty adaptation. As such, we employ a simple heuristic in each domain to detect the introduction of novelty. The use of a 5-episode rolling average reward allows us to discern significant deviations from the expected performance, indicating potential novelties that may impact the agent's adaptation process. This heuristic is particularly effective in identifying novelties that substantially affect performance, as indicated by a consistent decrease in the rolling average reward over multiple episodes. However, it may not capture nuances in cases where novelties manifest as transient fluctuations in performance or if the novelty does not significantly impact the agent's ability to learn.

In the case of Angry Birds, the novelty status is directly provided to the agent, simplifying the adaptation process by eliminating the need for the rolling average heuristic. This direct provision of novelty status ensures that the agent can promptly adapt to novel scenarios as soon as they arise.

To maintain consistency across domains, all agents initiate training immediately after the detection of novelty. As mentioned, this detection is determined by the 5-episode rolling average falling below a domain-dependent threshold, signalling a significant deviation from expected performance. Once novelty is detected, agents commence training in the novel environment.

Generation of Novelties: Post-novelty environments in CartPole, MountainCar, and

CrossRoad are created by varying the physical or spatial parameter values of the environment. For the Angry Birds domain, we use novelties from the 2022 AIBirds Competition [2] Novelty Track. Further details on novelty generation are discussed in subsequent sections. It is important to note that the agent possesses no prior knowledge of the potential novelties that may be introduced.

6.5.1 CartPole

The OpenAI Cartpole environment serves as a classical reinforcement learning benchmark, originating from the cart-pole problem described by Barto et al. [23]. In this environment, a pole is attached to a cart, and the objective is to keep the pole balanced upright by moving the cart left or right. A reward of +1 is provided for each step in which the pole remains upright. The episode terminates if the pole deviates more than 12 degrees from the vertical position or if the cart moves more than 2.4 units from the centre, representing the edge of the display. To ensure a finite reward, we cap the maximum possible reward at 200, resulting in total episode rewards ranging from 1 to 200.

To approximate regret, we use a simple heuristic based on the absolute value of the angular speed of the pole after the agent chooses an action. If the angular speed increases beyond a threshold of 0, the action is considered to require adaptation. This heuristic allows us to assess the effectiveness of the agent's actions in mitigating undesirable pole movements.

Baseline Agents

We assess our method using two pre-trained agents: Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) [136]. Both agents solve the tasks in the pre-novel environment. We evaluate each agent under four settings: no learning (-Baseline), online learning (-Online) where the entire network undergoes training, fine-tuning (-FineTune) that only the last layer of the network is trained, and ours (-NAPPING).

Novelty

To introduce novelty, we manipulate various parameters in the CartPole environment, including the length of the pole, gravity, mass of the cart, mass of the pole, and magnitude of the pushing force, each initially set to default values of 0.5, 9.8, 1, 0.1, and 10, respectively. Exploring the efficacy of our proposed method, we sample parameter values across an extensive range, spanning from one-tenth to ten times the default value for each parameter:

- Length: the length of the pole from 0.05 to 5 units
- Gravity: the gravity of the environment from 0.98 to 98 units
- MassCart: the mass of the cart from 0.1 to 10 units

- Masspole: the mass of the pole from 0.01 to 1 unit
- Force_mag: the moving force of the cart from 1 to 100 units

For each trial, parameter values in the novel environment are uniformly sampled from the specified ranges. We conduct 5000 trials per agent, employing a consistent novelty detection heuristic: a novelty is detected if the five-rolling average episodic reward falls below 150.

CartPole Results and Discussion

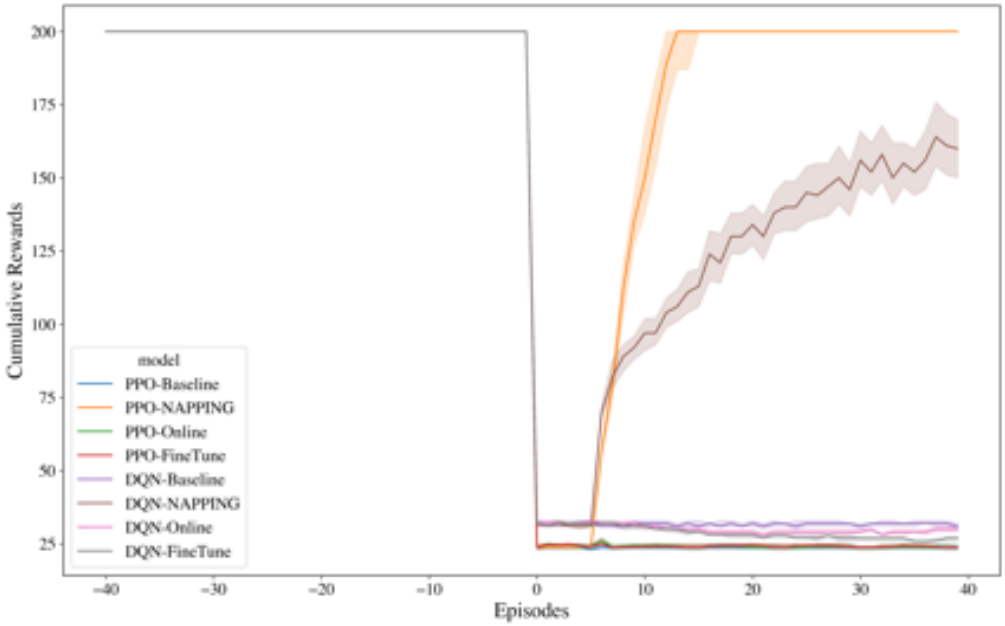


Figure 6.4: Overall Cumulative Rewards on CartPole by Agents

Figure 6.4 displays the overall cumulative rewards on CartPole for all the models investigated. Here, we utilise the median instead of the mean to mitigate the impact of results from unsolvable or challenging novel environments. Observing the figure, it is evident that none of the baseline, online learning, and fine-tuning agents exhibit any significant adaptation. In contrast, both the PPO-NAPPING agent and the DQN-NAPPING agent react promptly and effectively to the introduced novelties. Notably, the median cumulative rewards of PPO-NAPPING reach 200 in less than ten episodes. Although DQN-NAPPING also demonstrates substantial adaptation progress, it adapts less efficiently compared to PPO-NAPPING. This suggests that PPO has learned a more suitable embedding space that enables NAPPING to identify practical adaptation

principles quickly.

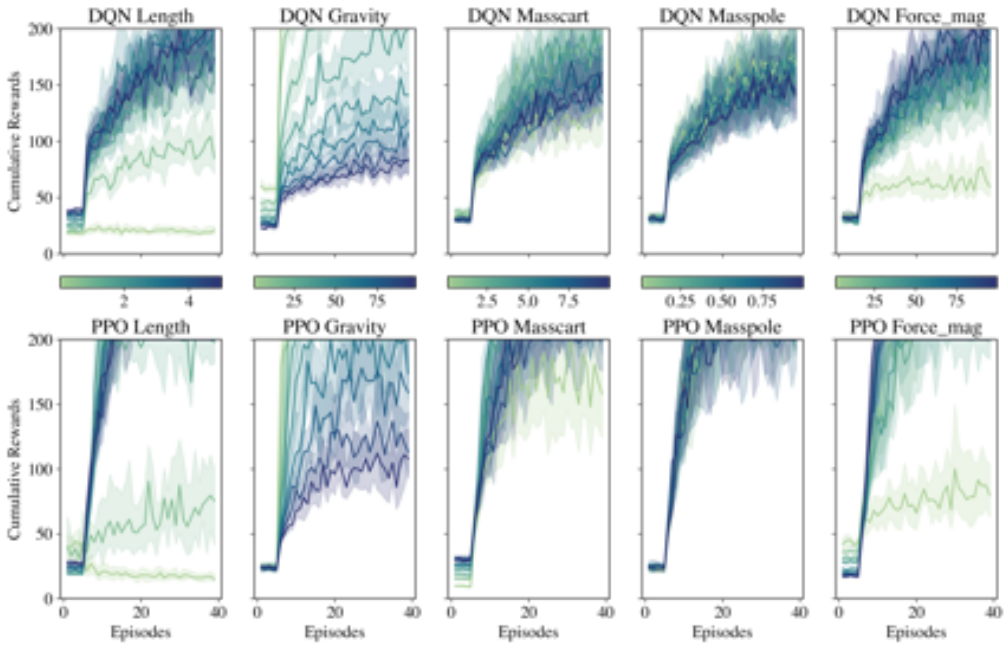


Figure 6.5: Overall Cumulative Rewards on CartPole By Environment Parameter Values. The first row illustrates the performance of the DQN-NAPPING agent, while the second row illustrates the PPO-NAPPING agent. The colour of the cumulative rewards lines in each plot represents the specific parameter values used to generate the testing environment, with darker colours indicating higher parameter values.

Figure 6.5 depicts the performance of DQN-NAPPING and PPO-NAPPING across various parameter values specifically for post-novelty episodes. The initial five episodes showcase the performance of DQN-Baseline and PPO-Baseline, as novelties are detected starting from episode 5. The visualization highlights how NAPPING facilitates adaptation for both DQN and PPO agents across a spectrum of novelties, ranging from relatively easy to more challenging scenarios. Additionally, the figure confirms previous findings [181] indicating that the performance of baseline agents significantly deteriorates even in scenarios where tasks become ostensibly easier, such as when gravity decreases. Notably, NAPPING empowers standard DQN and PPO models to adapt effectively to the majority of novelties, with exceptions for particularly demanding situations, such as when the pole length or pushing force reaches minimal values, or when gravity becomes exceptionally high.

6.5.2 MountainCar

The OpenAI MountainCar environment is a widely studied control domain based on the deterministic Markov Decision Process (MDP) problem outlined by [119]. In this environment, an underpowered car navigates along a curve, striving to reach a goal state positioned atop a "mountain." At each time step, the agent selects one of three actions: Forward, Stay, or Backward. The Forward action propels the car in the positive x direction, while the Backward action moves it in the negative x direction. The agent's state is defined by two variables: the horizontal position, x, and velocity, $\dot{x}$. A reward of -1 is received per time step, motivating the agent to reach the goal state as swiftly as possible. An episode concludes either when the agent achieves the goal state or after 500 time steps, whichever comes first. Consequently, the reward for an episode falls within the range of 0 to -500. An episode is failed if the agent receives -500.

To approximate regret in the MountainCar environment, we evaluate the action based on the direction of the car. If the car is moving left and the chosen action is to move right, this discrepancy results in a high regret value. Conversely, if the action aligns with the car's current direction, it receives a low regret value.

Baseline Agents

To assess the performance of NAPPING under varying baseline conditions, we include the Proximal Policy Optimization (PPO) algorithm: PPO agent, along with the Deep Q-Network (DQN) agent [136]. PPO and DQN demonstrate proficiency in solving the pre-novelty environment, achieving average episodic rewards of approximately -100 and -120, respectively. PPO also exhibits superior generalisation capabilities, successfully solving even many of the novel situations. Including this agent allows us to establish an empirical upper bound on performance, facilitating a more intuitive assessment of the extent to which NAPPING aids DQN in adapting.

Novelty

To introduce novelty into the MountainCar environment, we manipulate two key parameters: the pushing force (default: 0.001) and gravity (default: 0.0025). Given the potential for extreme parameter values to render the game unsolvable, such as minimal pushing force and substantial gravity, we limit the range of pushing force to $[0.0001, 0.02]$ and gravity to $[0.0001, 0.005]$ to create novelties.

Similar to the CartPole domain, each trial in MountainCar comprises 80 episodes, with the initial 40 episodes drawn from the pre-novelty environment and the subsequent 40 from the novel environment. Parameter values for the novel environment are uniformly sampled from the specified range for each trial. We conduct 2000 trials per agent, with novelty detection activated from episode 5 (the 45th episode) using a consistent heuristic across all agents: a novelty is detected if the five-episode rolling average episodic reward falls below -120 or exceeds -80.

6.5.3 MountainCar Results and Discussion

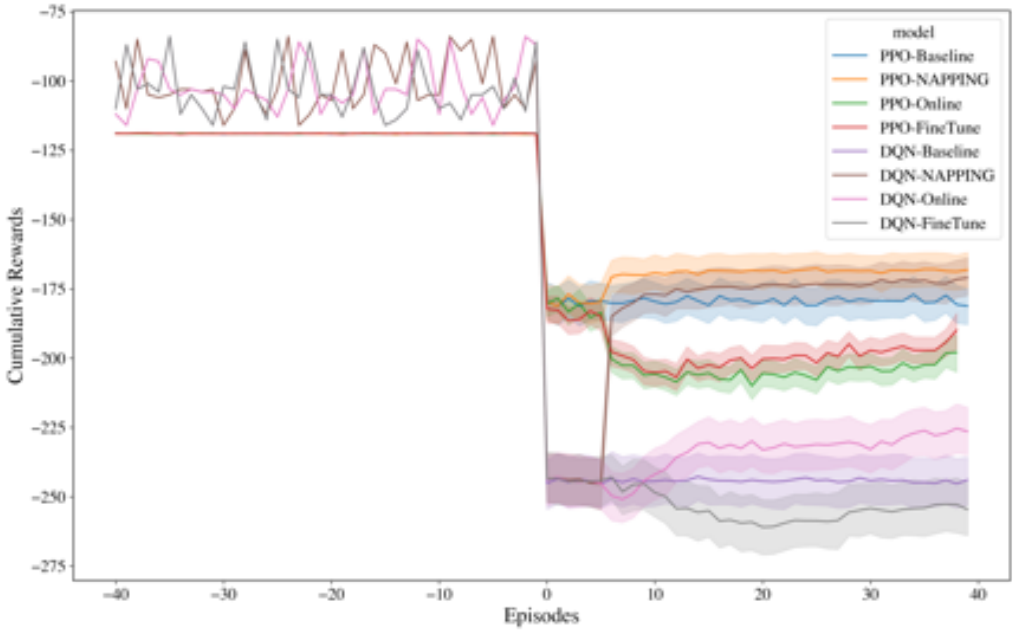


Figure 6.6: Overall Cumulative Rewards on MountainCar by Agents.

Figure 6.6 illustrates the overall cumulative rewards in the MountainCar domain. Unlike CartPole, where cumulative rewards reflect successful pole balancing, in MountainCar, they indicate the speed at which an agent reaches the goal state. Consequently, returning to the pre-novelty performance level is not always feasible as the environment may become more challenging. All "-NAPPING" agents demonstrate rapid adaptation to novel scenarios, with a notable increase in performance observed in episode 5 when adaptation principles are learned. Particularly, NAPPING achieves significant adaptation within the first episode in the post-novelty environment. Despite starting lower than PPO-NAPPING, DQN-NAPPING quickly catches up and converges to a similar performance level.

Figure 6.7 presents the average performance of -NAPPING agents on MountainCar across different novel parameter values. Notably, while PPO-NAPPING's performance aligns more consistently with task difficulty, DQN-NAPPING's performance declines across all gravity values. Additionally, PPO-NAPPING achieves better performance as the environment becomes less challenging (e.g., larger pushing force or lower gravity values). Interestingly, NAPPING aids DRL agents in adapting not only to novelties that increase task difficulty but also to those that simplify the environment.

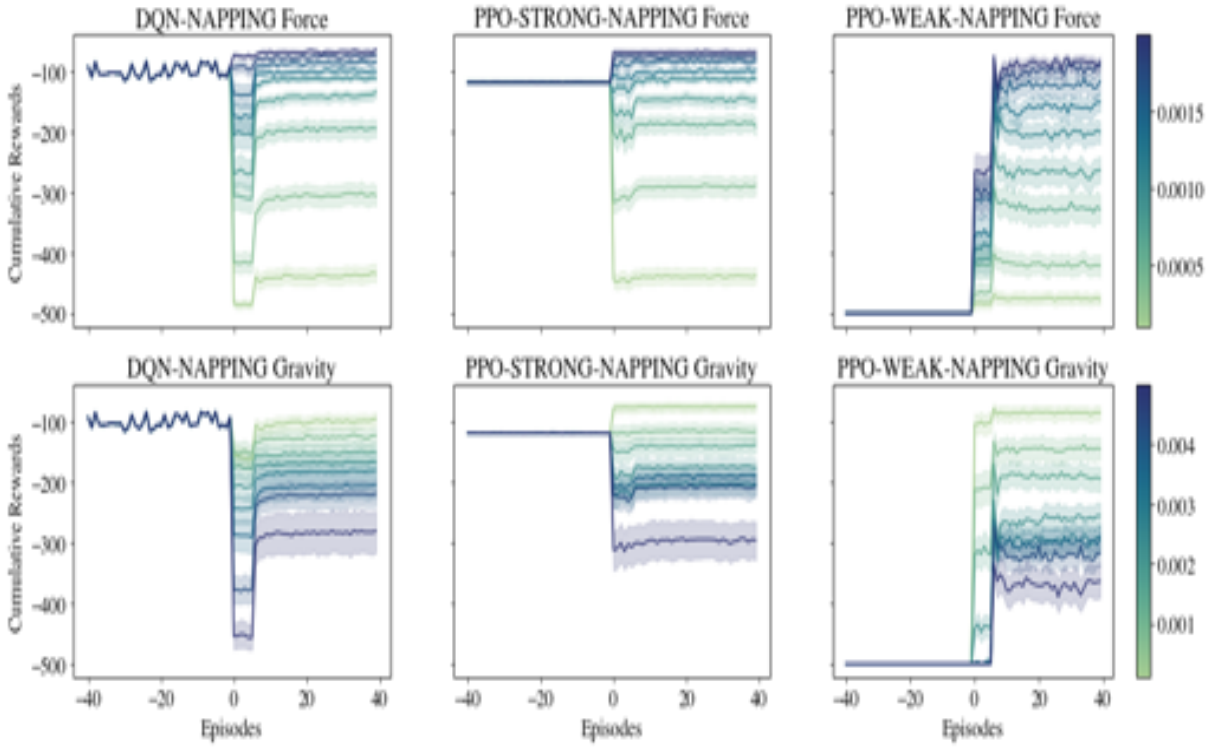


Figure 6.7: Overall Cumulative Rewards on MountainCar by Novelty. This figure showcases the average performance of -NAPPING agents on MountainCar across various novel parameter values. Following the evaluation methodology used in CartPole, the colour gradient of the average cumulative reward lines in each plot reflects the specific parameter values used to generate the environment, with darker shades indicating higher values.

Figure 6.8 provides a comparative analysis of performance across various gravity and force settings for all -NAPPING agents. Each dot on the graph represents an episode conducted with novel parameter values, with force represented on the x-axis and gravity on the y-axis. Dot colour signifies the cumulative reward attained, with darker green hues indicating higher rewards and red denoting failed tasks. Generally, tasks become more manageable with increased force and reduced gravity, though environments situated near the top-left corner may pose unsolvable challenges.

Interestingly, the performance of the PPO-Baseline agent exhibits variability, as the failed tasks are interspersed with solved tasks, highlighting its instability. Conversely, while the DQN-Baseline fails more episodes, its failures appear more consistent, making it easier to discern the boundary of the agent's ability.

Notably, with NAPPING, PPO-NAPPING was able to handle parameters that were previously unstable. While DQN-Baseline performs significantly poorer than PPO-Baseline,

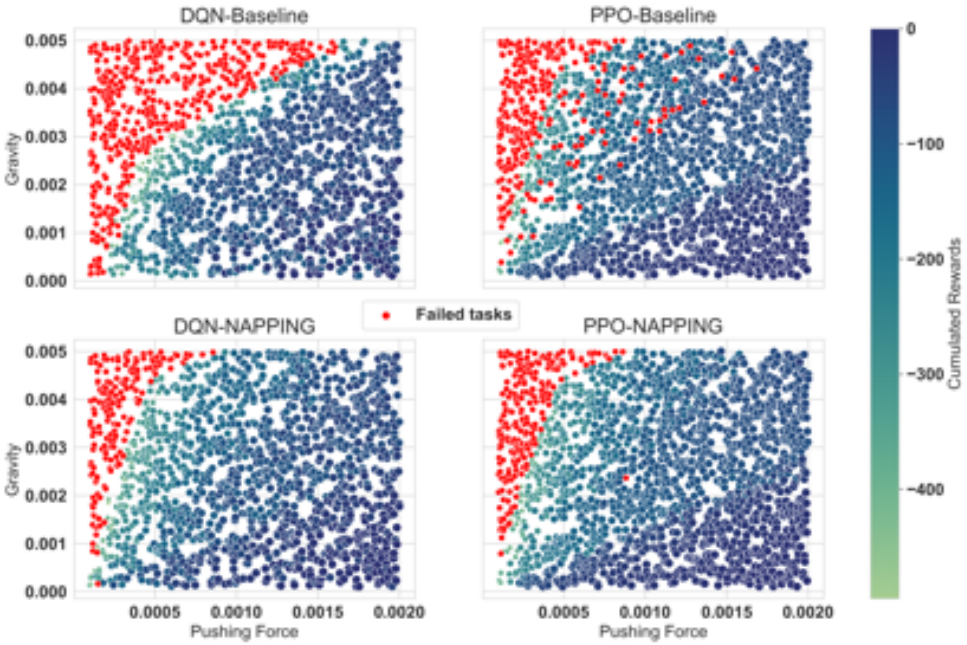


Figure 6.8: Overall Cumulative Rewards on MountainCar by Novelties. Each dot represents an episode with novel environment parameter values, where the x-axis denotes force and the y-axis represents gravity. The dot colour indicates the average cumulative reward achieved, with darker green colours indicating higher rewards and red indicating failed tasks (tasks can not be solved within 500 steps). The pre-novelty environment is marked as a yellow cross in the middle.

DQN-NAPPING achieves performance levels comparable to PPO-NAPPING, underscoring the effectiveness of NAPPING.

6.5.4 CrossRoad

CrossRoad is a domain inspired by OpenAI Freeway. Unlike OpenAI Freeway, CrossRoad offers full control over the speeds and locations of moving objects and allows the agent to move in multiple directions. It features 8 "cars" (red boxes) moving horizontally at various speeds and directions. The objective is to cross the road while navigating the "player" (green box) and avoiding collisions with cars. The agent can move left, right, up, down, or stay in place. Episodes begin with the player at the bottom line and end if the player collides with a car, crosses the road, or reaches the episode length of 100 steps. Rewards of +1 are given for crossing the road, -1 for colliding with a car, or

100 steps have passed. The agent operates with a partial observable state space, seeing only its location and adjacent eight cells, each with x and y coordinates, resulting in an 18-dimensional vector state.

As a pathfinding problem, we employ a commonly used heuristic to approximate regret: the distance to the finish line. Therefore, when evaluating actions, if the agent moves closer to the finishing line, the action receives a low regret score; conversely, if the action moves the agent away from the goal, it receives a higher regret score.

Baseline Agents

Due to the absence of pre-trained agents, we train DQN and PPO agents using default settings [136]. These agents solve all the tasks in the pre-novelty environment. Following the protocol established in previous domains, we compare offline (-Baseline), online learning (-Online), and fine-tuning (-FineTune) versions with NAPPING (-NAPPING) for both DQN and PPO agents.

Novelty

Eight novelty bases in CrossRoad represent different layouts of the game environment:

- Super Slow Speeds: Cars move much slower.
- Super Fast Speeds: Cars move much faster.
- New Speeds: Cars move at different speeds.
- Opposite Direction: Cars move in opposite directions.
- All to the Left: All cars move left.
- All to the Right: All cars move right.
- Shift Speeds: Car speeds are shifted.
- Reverse Cars: Car order is reversed.

We further vary initial positions and car speeds for each novelty base by adding noise sampled from $[-1, 1]$ for position and $[-10, 10]$ for speed. This noise is also applied to the pre-novelty environment, resulting in nine different novelties. Novelty detection occurs if the five-rolling average episodic reward falls below 1. Each DRL agent undergoes 100 trials per novelty, totalling 1800 trials.

CrossRoad Results and Discussion

Figure 6.9 illustrates the overall performance of the evaluated agents. Consistent with results from previous domains, only agents with NAPPING exhibited adaptation behaviour, with DQN-NAPPING and PPO-NAPPING achieving around 0 cumulative rewards, indicating a 50% task-solving rate for novel tasks. Conversely, all other agents

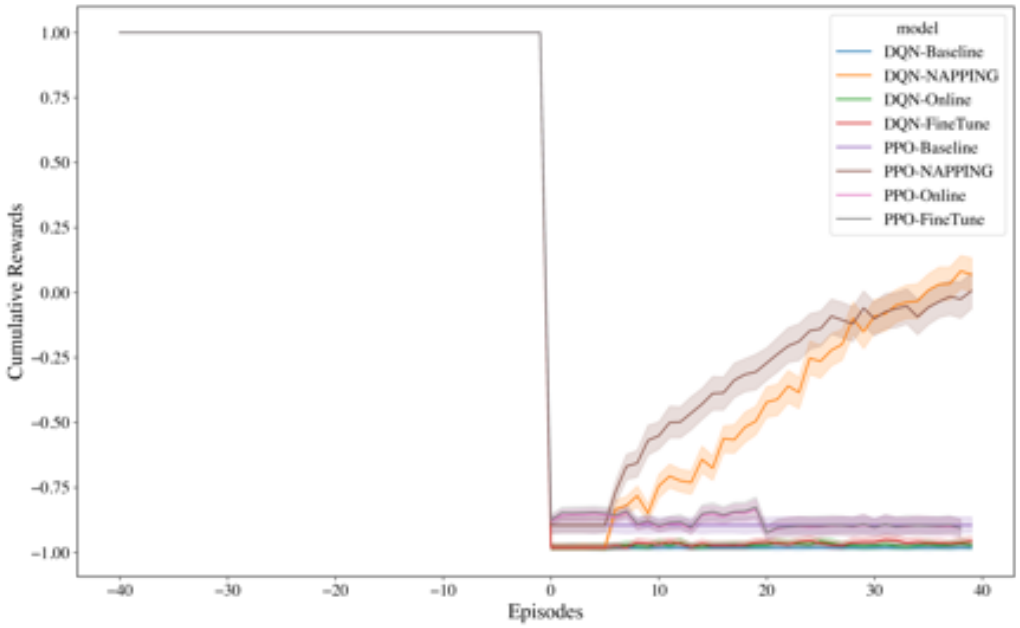


Figure 6.9: Overall Cumulative Rewards on CrossRoad

failed to learn effective policies to respond to the novelties. Comparing PPO-NAPPING and DQN-NAPPING, PPO-NAPPING demonstrated slightly faster adaptation due to a more robust baseline model but exhibited slightly lower asymptotic performance than DQN-NAPPING.

Figure 6.10 presents the performance of DRL agents with NAPPING. Interestingly, agents with different backbones exhibited varying adaptation performances. For instance, DQN-NAPPING responded more quickly to the New Speed novelty and achieved a higher asymptotic performance compared to PPO-NAPPING. Conversely, PPO-NAPPING outperformed DQN-NAPPING in All to the Right and Super Fast Speeds novelties and showed superior performance in Shifted Speeds and Super Slow Speeds novelties.

6.5.5 Angry Birds

Angry Birds, a widely popular game known for its realistic physical simulation, has become a prominent domain for testing physical reasoning among AI practitioners [140, 184]. In this game, the objective is to eliminate all pigs at each level by launching birds from a slingshot. Pigs are typically shielded by structures made of blocks with varying sizes, shapes, and materials. Some birds possess special abilities activated upon release from the slingshot. Players are limited to selecting a bird trajectory by pulling it back

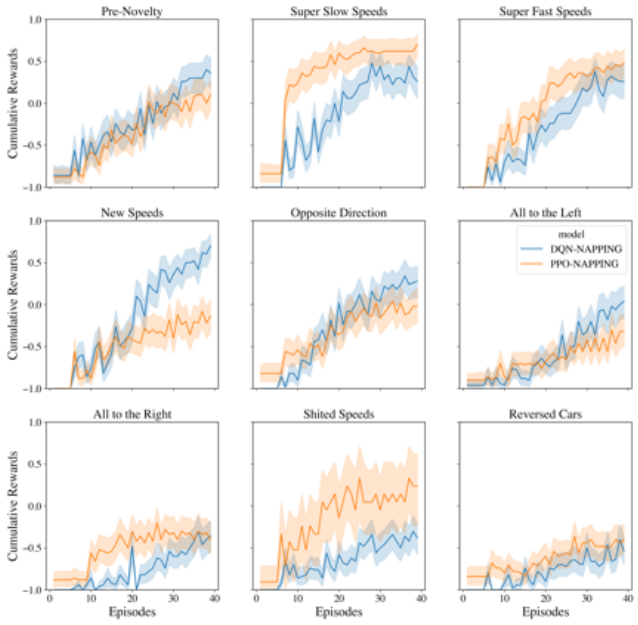


Figure 6.10: Overall Cumulative Rewards on CrossRoad by Novelty. Each plot depicts the performance of both DQN-NAPPING and PPO-NAPPING on the post-novelty environment, totalling 9 distinct novelties.

in the slingshot to predefined release coordinates (x, y) and then tapping the screen to activate special powers at the desired time t after release.

The AIBIRDS competition [2], organized annually since 2012 alongside the International Joint Conference on Artificial Intelligence, challenges AI researchers to develop agents capable of handling continuous action spaces and environments with incomplete knowledge of object physics. Since 2021, the AIBIRDS Novelty Track [185] has aimed to spur the development of agents equipped to handle unseen novelties using physical reasoning skills. In this study, we assess our method using novelties introduced in the latest AIBIRDS Novelty Track competition.

In the AIBIRDS Novelty Track, agents can request two types of state representations: screenshots and ground truth. A screenshot state is a $480x640$ coloured image, while ground truth is provided in JSON format, detailing foreground objects in a screenshot. Each object in the ground truth is represented as a polygon with vertices and colour maps quantizing colours appearing in the object. Agents can request these representations at any time during gameplay.

Given that NAPPING currently supports agents with discrete action spaces, we dis-

cretize Angry Birds' action space into three actions: selecting objects to shoot, choosing between high or low trajectories, and determining target offsets. To enhance agent strategy flexibility and address trajectory planner noise, we allow the agent to select from seven different target offset locations, ranging from no changes to 15/25/35 pixels vertical variations. Additionally, we assume the agent always uses full slingshot strength, pulling the bird to the furthest point before release.

In the context of Angry Birds, we employ a straightforward heuristic to approximate regret: the number of pigs destroyed. Essentially, the more pigs that are eliminated, the lower the regret associated with the action. We identify underperforming regions when the action fails to eliminate any pigs.

Baseline Agents

In the Angry Birds domain, we trained two types of agents: a DQN agent [173, 167] and a DQN-Rel agent equipped with a relational module [188]. These agents were modelled after those utilized in previous research [184]. Our training regimen followed the framework established by the competition organizers, leveraging pre-novelty tasks provided by them [184]. These tasks encompassed 2450 game levels generated from seven distinct templates.

During training, both agents completed a total of 20,000 episodes, maintaining a pass rate exceeding 95%. The pass rate reflects the proportion of game levels in which the agent successfully destroyed all pigs, a key performance metric in Angry Birds.

Our evaluation involved comparing our approach with several others. These included a heuristic adaptation agent (*Naive Adaptation*), three state-of-the-art open-world learning agents: *CIMARRON 2022* [2], *HYDRA* and *OpenMIND*) [160, 86, 122], the latter of which was the official champion of the 2022 AIBIRDS Competition Novelty Track. Additionally, we also included a non-adaptive agent (*DataLab*) [141] for comprehensive comparison purposes.

Novelty

The 2022 AIBIRDS competition Novelty Track introduces several novelties:

- Green Egg: A green rectangular object resembling pigs but with a different shape
- Bone: An object requiring destruction to progress in the game level
- Taller Slingshot: The slingshot is twice as tall, requiring adjusted release points
- Pig Color: All objects are coloured like pigs
- Circumcircle: Objects represented by circumcircles in ground truth instead of shapes
- Floating Wood: Wooden objects can float on ice

- Bird Likes Pig: Red Bird cannot damage pigs

Evaluation of all agents follows the trial setting, with the number of pre-novelty episodes varying randomly between each trial, as per competition settings. Each trial consists of 40 novel episodes, with 40 trials conducted for each novelty.

Angry Birds Results and Discussion

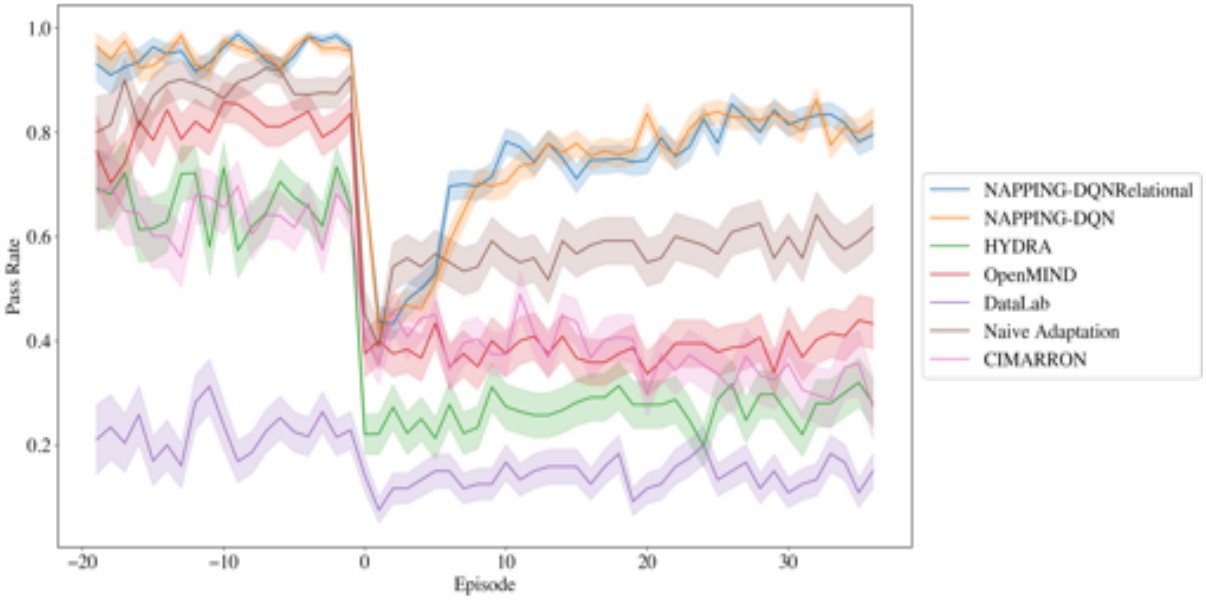


Figure 6.11: Average pass rate of Agents on AIBIRDS Competition Novelty Track by Agent

Figure 6.11 illustrates the overall pass rates of the seven agents evaluated in the Angry Birds domain. Unlike previous domains, the number of pre-novelty episodes varies randomly in Angry Birds. Therefore, we designate the episode where novelty is introduced as 0, with pre-novelty episodes represented as negative numbers. *DQN-NAPPING* and *DQNRel-NAPPING* exhibit the strongest novelty adaptation performance among all agents. While averaging just above 0.9 in pre-novelty instances, the pass rates of -NAPPING agents drop to below 0.4 immediately after novelty introduction, recovering to around 0.8 before trial completion. The *Naive Adaptation* agent, though quick to respond to some novel tasks within 1 episode, shows lower asymptotic performance compared to *DQN-NAPPING* and *DQNRel-NAPPING*. *OpenMIND*, a state-of-the-art open-world learning agent, achieves pass rates above 0.8 pre-novelty and above 0.4 post-novelty introduction. Notably, *HYDRA* and *CIMARRON* agents exhibit similar pre-novelty performance, with *CIMARRON* showing greater novelty resilience and less pronounced performance reduction post-novelty. However, *CIMARRON* appears to struggle in adapting to novelties, with pass rates gradually decreasing after novelty introduction. Despite its previous championship in the standard AIBIRDS Competition track, the

non-adaptive *DataLab* agent displays the lowest overall performance, with pass rates ranging from around 0.3 to below 0.2 across agents.

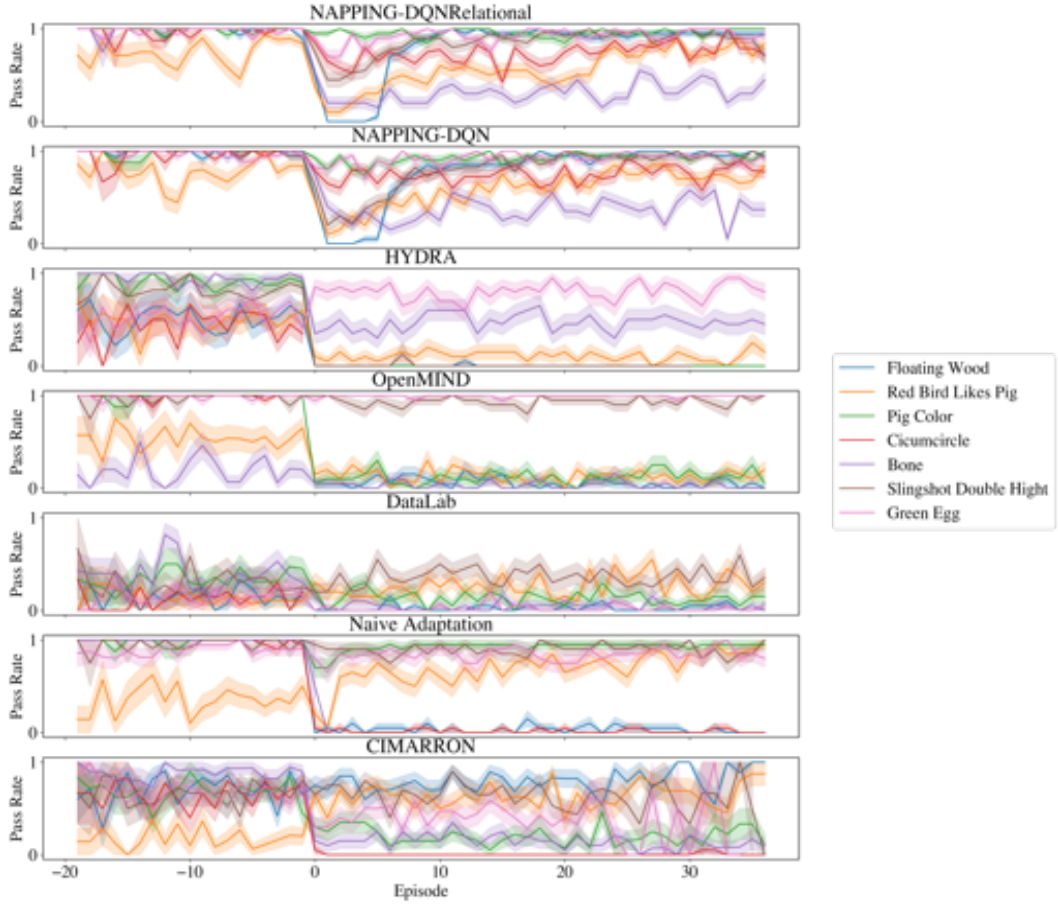


Figure 6.12: Average pass rate of Agents on AIBIRDS Competition Novelty Track by Novelty

Figure 6.12 presents pass rates of agents by different novelties. -NAPPING agents exhibit responsiveness to all novelties, returning to pre-novelty performance levels except for the Bone novelty. Notably, -NAPPING agents also demonstrate resilience to the Pig Color novelty, as agents (including *Naive Adaptation*) use object labels directly instead of colour maps. However, *Naive Adaptation* fails to adapt to the Circumcircle and Floating Wood novelties. For the Circumcircle novelty, this failure stems from a break in the *Naive Adaptation*'s input state assumption regarding object vertex representation, leading to crashes. Regarding the Floating Wood novelty, *Naive Adaptation* struggles to adjust trajectory strategies across different game levels, as it can only remember one solution triplet. Conversely, -NAPPING agents lack specific input state assumptions, working directly over DRL model embedding spaces. Additionally, -NAPPING agents can identify diverse solution actions for tasks with distinct spatial arrangements and

timing requirements, assumed to have varied representations in the embedded space of DRL agents. Pass rates of agents on each novelty are included in Appendix 6.8.

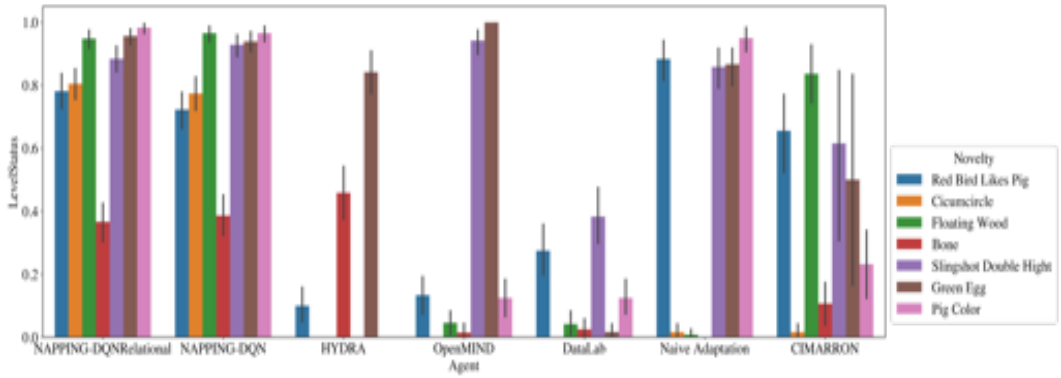


Figure 6.13: Average asymptotic (last 10 episodes) pass rate on AIBIRDS Competition Novelty Track

Figure 6.13 displays the asymptotic (last ten episodes) pass rates of different agents across different novelties. -NAPPING agents demonstrate similar asymptotic performance levels, with DQNRel slightly outperforming DQN in Bird Likes Pig and Circumcircle novelties but slightly underperforming in others. *Naive Adaptation* shows relatively high pass rates if its heuristic can solve novelties but fails otherwise. *OpenMIND* excels particularly in adapting to Green Egg and Taller Slingshot novelties, achieving the highest asymptotic pass rates among all agents but fares poorly in other novelties. *CIMARRON* appears highly robust, with pass rates consistently higher than the non-adaptive agent *DataLab*.

6.6 Discussion

In this section, we conduct a comprehensive examination of the underlying mechanisms of NAPPING. We present visual representations that illustrate the dynamic evolution of embedding spaces, comparing NAPPING with the conventional deep reinforcement learning algorithm, DQN in the domain of CartPole. Our objective is to provide an in-depth understanding of the inner workings of NAPPING, shedding light on why it surpasses traditional learning methods.

We begin by introducing a novel scenario in the CartPole domain, where we increase the length of the pole. This novelty is depicted in Figure 6.14. To evaluate the effectiveness of NAPPING in comparison to a standard DQN agent (referred to as DRL in the figures), similarly to the previous sections, we incorporate NAPPING into the DQN framework (represented as NAPPING in the figures). The standard DQN agent achieves optimal performance in solving the pre-novelty CartPole domain, consistently attaining an average cumulative reward of 200 per episode.

Figure 6.14: Example Novelty in the CartPole Domain: Increased Length of the Pole

To elucidate the distinctions between NAPPING and standard DRL methods, we analyzed the evolution of their embedding spaces across episodes during the learning process. Our investigation highlights that the introduction of novelty doesn't necessarily demand a complete overhaul of the agent's policy. Rather, NAPPING strategically modifies only the regions in the embedding space where the previous policy proves ineffective, resulting in a more efficient learning process.

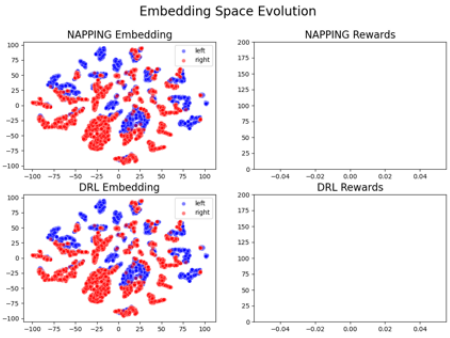
Figure 6.15 provides insight into the changes occurring from episode 0 to episode 24. A notable observation is the contrasting dynamics between the embedding spaces of the NAPPING and DQN agents. While both agents start from the same embedding spaces, with the selection of "left" and "right" actions being nearly equal, the embedding space of the DQN agent undergoes more pronounced changes over time. Specifically, by episode 16, the DQN agent demonstrates a tendency to favour the "right" action in a larger portion of the embedding space, with a further shift towards "left" action dominance by episode 24.

Conversely, the changes in the embedding space of the NAPPING agent are comparatively smaller. This discrepancy arises from the nature of NAPPING, which adjusts the mapping only when the selected action within a partition of the embedding space proves ineffective. Consequently, NAPPING achieves a maximum reward of 200 within 20 training episodes, highlighting its efficiency in adapting to the environment.

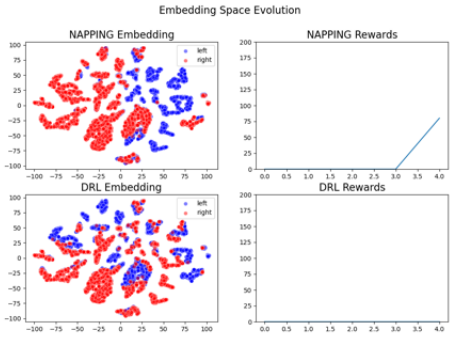
Figure 6.16 illustrates the performance from episode 100 until the end of the training, which is episode 200. Notably, while NAPPING has already completed its training in episode 20, the DRL agent requires 165 episodes to achieve an average reward of 200.

Furthermore, in Figure 6.16, subfigure (f) illustrates that the embedding space between the DQN and NAPPING closely aligns. Notably, while it took the DQN agent 200 episodes to reach this embedding space, NAPPING achieved the same outcome in just 20 episodes.

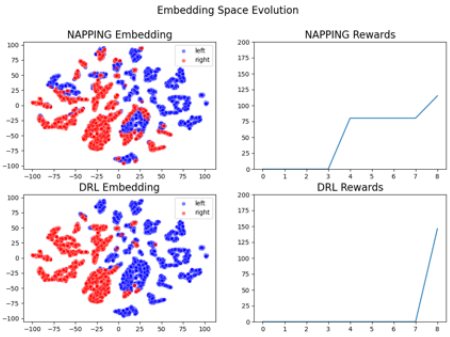
Moreover, the comparison between the initial embedding space in Figure 6.15, subplot (a), and the final embedding space further validates the rationale behind NAPPING. It underscores that the introduction of novelty doesn't necessarily mandate a global adjustment of the agent's policy. Instead, NAPPING strategically adjusts only the



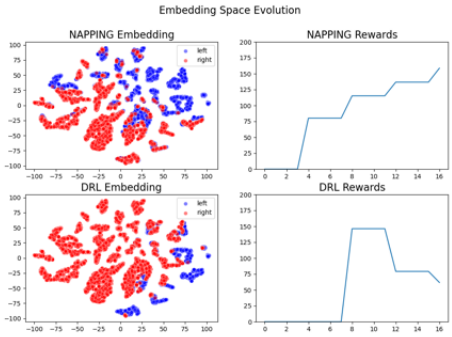
(a) Episode 0



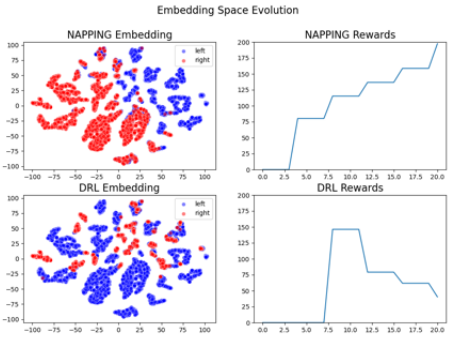
(b) Episode 4



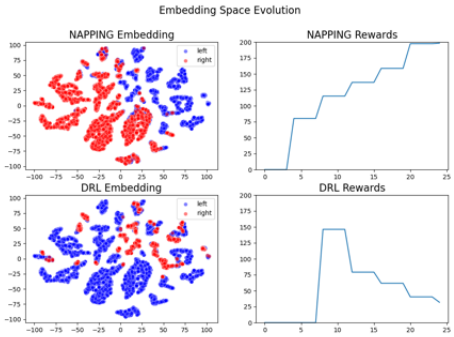
(c) Episode 8



(d) Episode 16

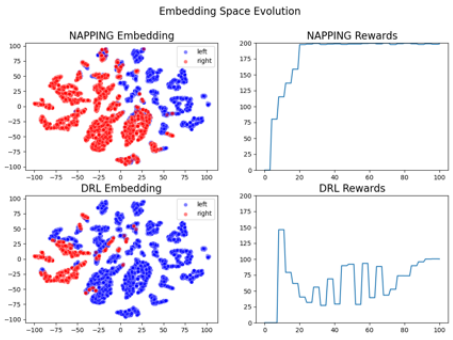


(e) Episode 20

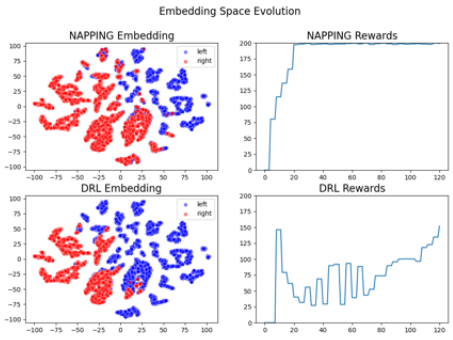


(f) Episode 24

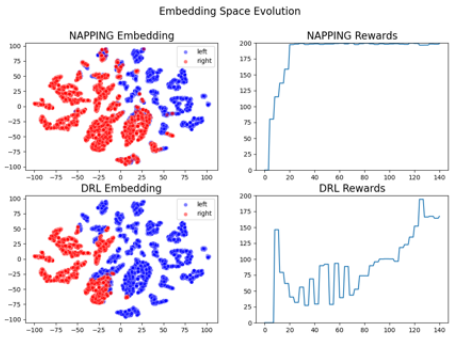
Figure 6.15: Evolution of Embedding Space from Episodes 0 to 24. Each subplot in the visualization depicts the embedding space of DQN with NAPPING (NAPPING) alongside the DQN (DRL) agent up to a specific episode. The embedding space was generated using t-SNE, with red dots representing the model-embedded states where the agent selects the action "right," while blue dots denote the agent choosing "left." Additionally, the subplots in the second column present the average reward achieved by the method up to the current episode, calculated by averaging over 200 testing episodes.



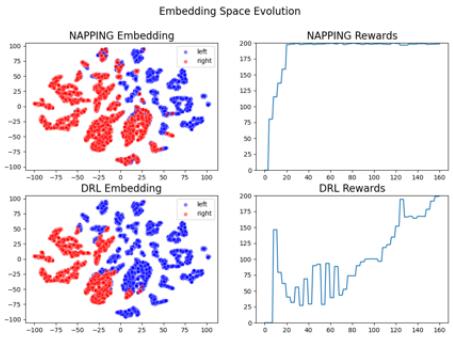
(a) Episode 100



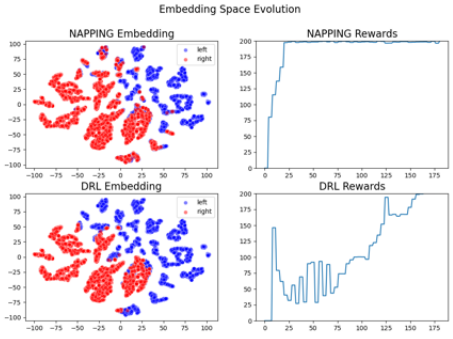
(b) Episode 120



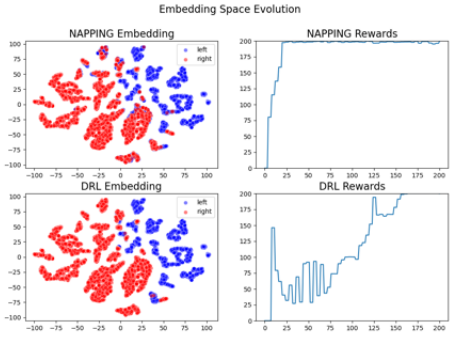
(c) Episode 140



(d) Episode 160



(e) Episode 180



(f) Episode 200

Figure 6.16: Evolution of Embedding Space from Episodes 100 to 200

regions in the embedding space where the previous policy is ineffective, leading to a more efficient learning process.

Limitations: One potential limitation arises when the novelty does not alter the proximity of states in the embedded space. In such cases, states that were previously close may become irrelevant, leading to ineffective adaptation principles. This can occur when the sensors of the agent cannot observe the novelty directly.

For instance, imagine a scenario in the cart-pole domain where a randomly moving block enters the scene and poses a potential collision risk with either the cart or the pole. Despite the dynamic nature of the environment, the agent's sensors are unable to directly perceive the presence of the moving block from the state space representation. Consequently, NAPPING may fail to detect and adapt to this novel event effectively.

In this scenario, the adaptation principles generated by NAPPING may not accurately account for the presence of the moving block, as the novelty is not observable to the agent's sensors. As a result, the underperforming regions identified by NAPPING may be random or misaligned with the actual areas requiring adaptation. This limitation highlights the importance of ensuring that the agent's sensory capabilities are appropriately aligned with the types of novelty present in the environment for effective adaptation.

Another limitation arises from the reliance on heuristics for evaluating action regret. While heuristics provide a practical means of assessing action quality, they may not always capture the full complexity of the environment. In certain scenarios, heuristic-based evaluation may lead to suboptimal adaptation decisions, particularly if the heuristics fail to accurately capture the nuances of the task. We discuss the future work to address this issue in Section 6.7.1.

Additionally, NAPPING's effectiveness may be influenced by the quality and representativeness of the training data. If the training data does not adequately cover the diversity of potential environments or lacks sufficient examples of novel scenarios, the learned adaptation principles may not generalize well to unseen environments. This limitation underscores the importance of robust and diverse training data in ensuring the effectiveness of NAPPING across a wide range of scenarios. The future work to mitigate this problem is also discussed in Section 6.7.1.

6.7 Conclusion and Future Work

In this paper, we introduced NAPPING (**N**ovelty **A**daptation **P**rinci**p**les Learn**ing**), a novel approach that addresses the challenge of efficiently adapting reinforcement learning agents to novel scenarios. Through comprehensive evaluations across four distinct action domains, we empirically demonstrate the efficacy of NAPPING in outperforming traditional deep reinforcement learning approaches in adapting to novel environments.

Our investigation into the inner workings of NAPPING revealed its ability to strategically modify the agent's policy only in regions of the embedding space where the

previous policy proves ineffective. This targeted adaptation approach allows NAPPING to achieve superior performance more efficiently compared to conventional methods.

Visual representations of the dynamic evolution of embedding spaces provided valuable insights into the learning process of NAPPING. By analyzing the changes in the embedding space across episodes, we observed that NAPPING adapts to novel scenarios with minimal disruption to previously learned policies, resulting in faster convergence and higher overall performance.

Despite its effectiveness, NAPPING is not without limitations. The method may struggle to adapt when novelty is not directly observable to the agent's sensors, highlighting the importance of aligning sensory capabilities with the types of novelty present in the environment. Additionally, reliance on heuristic-based evaluation and the quality of training data may influence NAPPING's effectiveness and generalization capabilities. We discuss also the potential future works in Section 6.7.1.

In conclusion, NAPPING represents a promising advancement in the field of reinforcement learning, offering a principled approach to efficiently adapting agents to novel scenarios. By leveraging targeted policy modifications and adaptive learning mechanisms, NAPPING holds the potential to significantly improve the performance and adaptability of reinforcement learning agents in real-world applications.

6.7.1 Future Work

Several potential improvements could make NAPPING more generalisable and effective. One avenue for future work could involve mitigating the need for predefined heuristic-based regrets and thresholds. One approach to addressing this limitation could involve dynamically updating the evaluation function upon entering novel environments using model states and episodic rewards. Additionally, extending NAPPING to work with agents possessing continuous action spaces would broaden its applicability to a wider range of environments. Finally, rather than sampling actions from the action space, learning which actions are more likely to achieve the task could enhance NAPPING's efficiency.

Furthermore, addressing the challenge of generating high-quality embedding spaces for NAPPING could benefit from various specific techniques and methodologies. One potential avenue is to explore advanced neural network architectures tailored for learning representations that effectively capture semantic similarities. Techniques such as Siamese networks [42] can train models to embed states into a latent space where similar states are closer together and dissimilar states are farther apart. By leveraging these techniques, we can ensure that the embedding space preserves essential semantic relationships among states, thereby enhancing NAPPING's ability to identify and adapt to novel situations. Furthermore, leveraging techniques from unsupervised or self-supervised learning can be beneficial for learning robust embeddings. Methods such as contrastive learning or predictive coding aim to learn representations that are invariant to irrelevant variations in the input data while preserving relevant information. By training NAPPING with

embeddings learned using these techniques, we can ensure that it focuses on capturing the essential characteristics of the environment, leading to more reliable adaptation to novel situations.

6.8 Appendix: Performance of all agents per novelty in the AIBIRDS competition novelty track

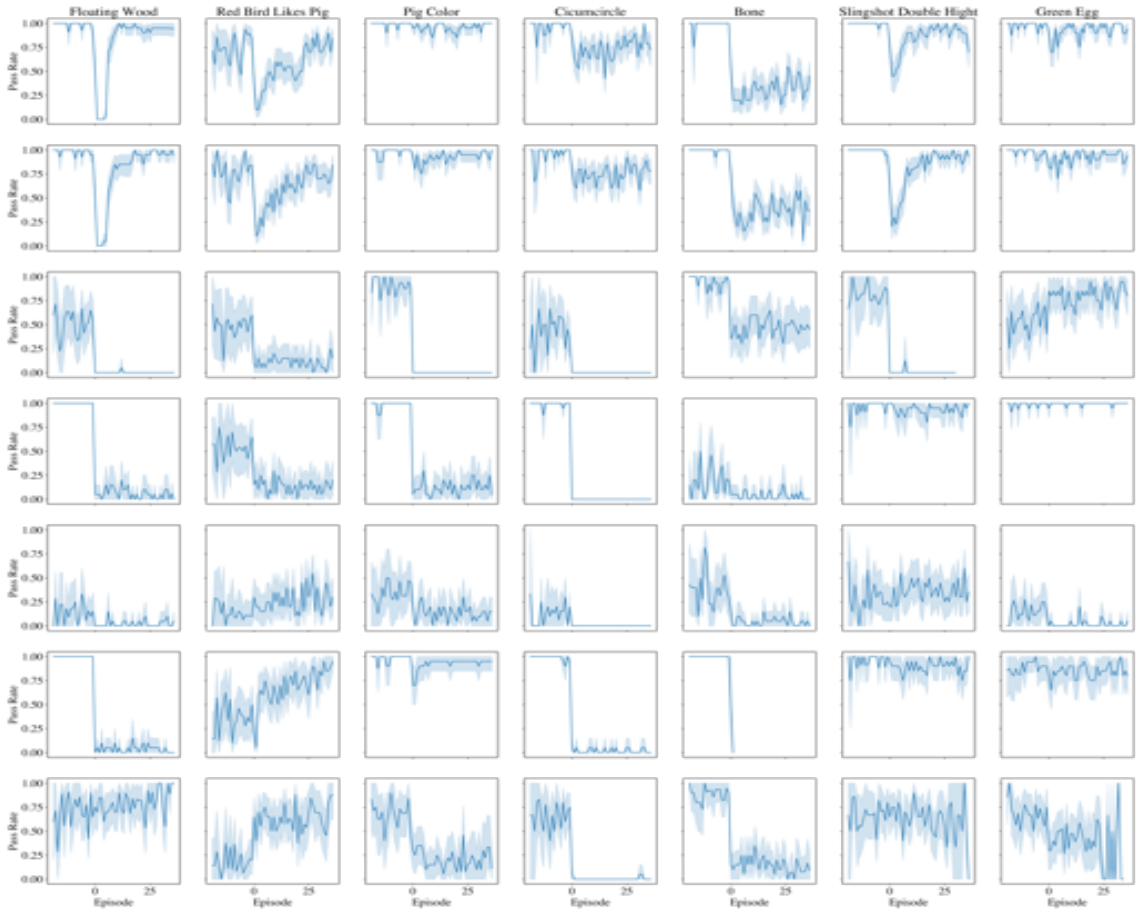


Figure 6.17: Performance of all agents per novelty in the AIBIRDS competition novelty track

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we focused on the challenges of developing AI systems capable of operating in open and unpredictable environments. Our research progressed through three interconnected phases:

Addressing Evaluation Challenges in Physical Domains: Determining whether an AI agent's success in handling novel scenarios in the real physical world arises from a genuine understanding of adaptation or mere luck is not straightforward. To address this problem, we developed a new evaluation methodology and a benchmarking framework, named Phy-Q, to measure an agent's physical reasoning intelligence. This framework provides a foundation for an accurate and comprehensive assessment of the physical reasoning capabilities of AI agents.

Creating a Versatile Benchmark for Open-World Learning We then introduced the Science Birds Novelty framework and the NovPhy benchmark, which offer controlled yet versatile environments for AI agents to encounter and adapt to a wide range of novelties. These tools facilitate the systematic introduction of unforeseen elements into an AI agent's operational landscape, aiding in a comprehensive evaluation of AI agents' open-world learning capabilities.

Developing Adaptive AI Agents with NAPPING We finally presented the NAPPING (Novelty Adaptation Principles Learning) algorithm, which enables AI agents to rapidly adapt to novel situations. NAPPING operates on top of trained deep reinforcement learning agents and helps them to rapidly adapt to different novel situations. Empirical demonstrations in the four simulated environments show that NAPPING outperforms traditional deep reinforcement learning and state-of-the-art open-world learning

approaches in adapting to novel environments.

The implications of our work extend to various industries, including autonomous vehicles, robotics, healthcare, and disaster response. Enhancing the adaptability of AI systems can lead to more effective collaboration between AI and human counterparts in dynamic and ever-changing conditions.

In conclusion, our research has taken important steps towards developing AI agents that can be deployed in the real physical world. We hope our work can contribute to a future where AI systems assist us in our daily activities in real-world environments.

7.2 Future Work

The current implementation of NAPPING employs a somewhat 'greedy' adaptation strategy, prioritizing the maximization of the *thre* value for the current action using the *Eval* function. However, the *Eval* function's limitation lies in its evaluation of only the immediate state s_t and the subsequent state s_{t+1}, failing to capture the long-term value of actions. Future enhancements will expand the algorithm's scope from the state space S to the state-action space SA and redefine NAPPING's high-level algorithm using Q-value functions.

Future work can focus on enhancing the NAPPING algorithm by eliminating the dependency on predefined components, specifically the *Eval* function and the threshold parameter, *thre*. This enhancement aims to enable NAPPING to achieve the following tasks:

1. **Dynamic Ineffectiveness Detection:** Allow NAPPING to autonomously recognize when the trained policy loses its effectiveness within the target domain.

2. **Apply to Continuous Action Space**: Extend the algorithm's applicability to domains with continuous action spaces.

Moreover, a more comprehensive comparison of the NAPPING approach with transfer learning, particularly weight transfer and recent methods for network restructuring, could provide deeper insights into the method's effectiveness, especially in more complex scenarios where outdated knowledge may hinder performance. Further exploration of hybrid planning and reinforcement learning techniques would also be valuable, building on the promising results in simpler problems. Expanding the range of test domains and including more complex novelty types could offer a broader evaluation of the NAPPING framework. Lastly, while the dissertation outlines the challenges of open-world novelty, a more detailed exploration of the distinctions between broad generalization and open-world novelty would strengthen the work and provide a clearer framework for future research.

Bibliography

[1] Ahmed*, O.; Träuble*, F.; Goyal, A.; Neitz, A.; Bengio, Y.; Schölkopf, B.; Wüthrich, M.; and Bauer, S., 2021. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. In *9th International Conference on Learning Representations (ICLR)*. https://openreview.net/pdf?id=SK7A5pdrgov. *equal contribution.

[2] AIBIRDS, 2021. Angry birds AI competition. http://aibirds.org/. http://aibirds.org/. [Accessed: August. 22, 2021].

[3] Allen, K. R.; Bakhtin, A.; Smith, K.; Tenenbaum, J. B.; and van der Maaten, L., 2020. Ogre: An object-based generalization for reasoning environment. In *NeurIPS Workshop on Object Representations for Learning and Reasoning*.

[4] Allen, K. R.; Smith, K. A.; and Tenenbaum, J. B., 2020. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117, 47 (2020), 29302–29310. doi:10.1073/pnas.1912341117. https://www.pnas.org/content/117/47/29302.

[5] Araújo, J. P.; Figueiredo, M.; and Botto, M. A., 2020. Single-partition adaptive q-learning. *arXiv preprint arXiv:2007.06741*, (2020).

[6] Araujo, J. P.; Figueiredo, M. A.; and Botto, M. A., 2022. Control with adaptive q-learning: A comparison for two classical control problems. *Engineering Applications of Artificial Intelligence*, 112 (2022), 104797.

[7] Araújo, J. P.; Figueiredo, M. A.; and Ayala Botto, M., 2022. Control with adaptive q-learning: A comparison for two classical control problems. *Engineering Applications of Artificial Intelligence*, 112 (2022), 104797. doi:https://doi.org/10.1016/j.engappai.2022.104797. https://www.sciencedirect.com/science/article/pii/S0952197622000732.

[8] Archibald, C.; Altman, A.; Greenspan, M.; and Shoham, Y., 2010. Computational Pool: A new challenge for game theory pragmatics. *AI Magazine*, 31 (Dec. 2010), 33–41. doi:10.1609/aimag.v31i4.2312.

[9] Arndt, K.; Hazara, M.; Ghadirzadeh, A.; and Kyrki, V., 2020. Meta reinforcement learning for sim-to-real domain adaptation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2725–2731. IEEE.

[10] Arulkumaran, K.; Deisenroth, M. P.; Brundage, M.; and Bharath, A. A., 2017. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, (2017).

[11] Bach, S. H. and Maloof, M. A., 2008. Paired learners for concept drift. In *2008 Eighth IEEE International Conference on Data Mining*, 23–32. IEEE.

[12] Baillargeon, R., 1995. Physical reasoning in infancy. *The cognitive neurosciences*, (1995), 181–204.

[13] Baillargeon, R. and DeJong, G., 2017. Explanation-based learning in infancy. *Psychonomic Bulletin and Review*, 24 (07 2017). doi:10.3758/s13423-017-1334-4.

[14] Baillargeon, R. and DeVos, J., 1991. Object permanence in young infants: Further evidence. *Child Development*, 62, 6 (1991), 1227–1246. http://www.jstor.org/stable/1130803.

[15] Baillargeon, R. and Devos, J., 1991. Object permanence in young infants: Further evidence. *Child Development*, 62, 6 (1991), 1227. doi:10.2307/1130803.

[16] Baillargeon, R. and Hanko-Summers, S., 1990. Is the top object adequately supported by the bottom object? young infants' understanding of support relations. *Cognitive Development*, 5, 1 (1990), 29–53. doi:https://doi.org/10.1016/0885-2014(90)90011-H. https://www.sciencedirect.com/science/article/pii/088520149090011H.

[17] Baillargeon, R.; Li, J.; Ng, W.; and Yuan, S., 2008. An account of infants' physical reasoning. *Learning and the Infant Mind*, (2008), 66–116. doi:10.1093/acprof:oso/9780195301151.003.0004.

[18] Baillargeon, R.; Needham, A.; and Devos, J., 1992. The development of young infants' intuitions about support. *Early Development and Parenting*, 1, 2 (1992), 69–78. doi:https://doi.org/10.1002/edp.2430010203. https://onlinelibrary.wiley.com/doi/abs/10.1002/edp.2430010203.

[19] Bakhtin, A.; van der Maaten, L.; Johnson, J.; Gustafson, L.; and Girshick, R., 2019. Phyre: A new benchmark for physical reasoning. In *NeurIPS*.

[20] Balloch, J.; Lin, Z.; Hussain, M.; Srinivas, A.; Wright, R.; Peng, X.; Kim, J.; and Riedl, M., 2022. Novgrid: A flexible grid world for evaluating agent response to novelty. *arXiv preprint arXiv:2203.12117*, (2022).

[21] Baradel, F.; Neverova, N.; Mille, J.; Mori, G.; and Wolf, C., 2020. Cophy: Counterfactual learning of physical dynamics. In *ICLR*. doi:10.21227/ps5q-8m55. https://dx.doi.org/10.21227/ps5q-8m55.

[22] Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D., 2017. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30 (2017).

[23] Barto, A. G.; Sutton, R. S.; and Anderson, C. W., 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13 (1983), 834–846.

[24] Battaglia, P. W.; Hamrick, J. B.; and Tenenbaum, J. B., 2013. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110, 45 (2013), 18327–18332.

[25] Bear, D.; Wang, E.; Mrowca, D.; Binder, F. J.; Tung, H.-Y.; Pramod, R.; Holdaway, C.; Tao, S.; Smith, K. A.; Sun, F.-Y.; et al., 2021. Physion: Evaluating physical prediction from vision in humans and machines. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

[26] Bendale, A. and Boult, T., 2015. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1893–1902.

[27] Bengio, Y.; Deleu, T.; Rahaman, N.; Ke, N. R.; Lachapelle, S.; Bilaniuk, O.; Goyal, A.; and Pal, C., 2020. A meta-transfer objective for learning to disentangle causal mechanisms. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

[28] Bifet, A. and Gavalda, R., 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, 443–448. SIAM.

[29] Bliss, J. and Ogborn, J., 1994. Force and motion from the beginning. *Learning and Instruction*, 4, 1 (1994), 7–25. doi:https://doi.org/10.1016/0959-4752(94)90016-7. https://www.sciencedirect.com/science/article/pii/0959475294900167.

[30] Borovička, T.; Špetlík, R.; and Rymeš, K., 2014. Datalab angry birds ai. http://aibirds.org/2014-papers/datalab-birds.pdf. http://aibirds.org/2014-papers/datalab-birds.pdf. [Accessed: July. 31, 2021].

[31] Boult, T.; Grabowicz, P. A.; Prijatelj, D.; Stern, R.; Holder, L.; Alspector, J.; Jafarzadeh, M.; Ahmad, T.; Dhamija, A. R.; Cli; Cruz, S.; Shrivastava, A.; Vondrick, C.; and Scheirer, W., 2021. Towards a unifying framework for formal theories of novelty. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35 (05 2021), 15047–15052.

[32] Boult, T. E.; Windesheim, N. M.; Zhou, S.; Pereyda, C.; and Holder, L. B., 2022. Weibull-open-world (wow) multi-type novelty detection in cartpole3d. *Algorithms*, 15, 10 (2022), 381.

[33] Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W., 2016. Openai gym. *ArXiv*, abs/1606.01540 (2016).

[34] Burachas, G. T.; Grigsby, S.; Ferguson, W.; Krichmar, J.; and Rao, R. Metacognitive mechanisms for novelty processing: Lessons for ai.

[35] CartPole, 2022. CartPole 3D domain. https://github.com/holderlb/WSU-SAILON-NG/tree/master/domains/cartpole.

[36] Chapman, D. and Kaelbling, L. P., 1991. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Ijcai*, vol. 91, 726–731.

[37] Cheke, L. G.; Loissel, E.; and Clayton, N. S., 2012. How do children solve aesop's fable? *PLOS ONE*, 7, 7 (07 2012), 1–12. doi:10.1371/journal.pone.0040574. https://doi.org/10.1371/journal.pone.0040574.

[38] Chen, Z.; Mao, J.; Wu, J.; Wong, K.; Tenenbaum, J.; and Gan, C., 2021. Grounding physical concepts of objects and events through dynamic visual reasoning. In *International Conference on Learning Representations (ICLR) 2021*. Vienna, Austria.

[39] Cheng, C.-A.; Kolobov, A.; and Swaminathan, A., 2021. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34 (2021), 13550–13563.

[40] Cheung, W. C.; Simchi-Levi, D.; and Zhu, R., 2020. Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism. In *International Conference on Machine Learning*, 1843–1854. PMLR.

[41] Chevalier-Boisvert, M.; Willems, L.; and Pal, S., 2018. Minimalistic gridworld environment for gymnasium. https://github.com/Farama-Foundation/Minigrid.

[42] Chicco, D., 2021. Siamese neural networks: An overview. *Artificial neural networks*, (2021), 73–94.

[43] Choi, D.; Morgan, M.; Park, C.; and Langley, P., 2007. A testbed for evaluation of architectures for physical agents. (01 2007).

[44] Chollet, F., 2019. On the measure of intelligence.

[45] Crosby, M.; Beyret, B.; Shanahan, M.; Hernández-Orallo, J.; Cheke, L.; and Halina, M., 2020. The animal-ai testbed and competition. In *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, vol. 123 of *Proceedings of Machine Learning Research*, 164–176. PMLR. https://proceedings.mlr.press/v123/crosby20a.html.

[46] Dao, J.; Green, K.; Duan, H.; Fern, A.; and Hurst, J., 2022. Sim-to-real learning for bipedal locomotion under unsensed dynamic loads. In *2022 International Conference on Robotics and Automation (ICRA)*, 10449–10455. IEEE.

[47] Davis, E., 2006. Physical reasoning. https://cs.nyu.edu/~davise/papers/handbookKR.pdf. https://cs.nyu.edu/~davise/papers/handbookKR.pdf.

[48] Day, R. H. and McKenzie, B. E., 1973. Perceptual shape constancy in early infancy. *Perception*, 2, 3 (1973), 315–320. doi:10.1068/p020315. https://doi.org/10.1068/p020315. PMID: 4794127.

[49] Diezmann, C. M. and Watters, J. J., 2000. Identifying and supporting spatial intelligence in young children. *Contemporary Issues in Early Childhood*, 1, 3 (2000), 299–313. doi:10.2304/ciec.2000.1.3.6. https://doi.org/10.2304/ciec.2000.1.3.6.

[50] Doctor, K.; Task, C.; Kildebeck, E.; Kejriwal, M.; Holder, L.; and Leong, R., 2022. Toward defining a domain complexity measure across domains. *Proceedings of the AAAI Conference on Artificial Intelligence, Designing Artificial Intelligence for Open Worlds*, (2022).

[51] Emery, N. J. and Clayton, N. S., 2009. Tool use and physical cognition in birds and mammals. *Current Opinion in Neurobiology*, 19, 1 (2009), 27–33. doi:https://doi.org/10.1016/j.conb.2009.02.003. https://www.sciencedirect.com/science/article/pii/S0959438809000063. Cognitive neuroscience.

[52] Enternainment, R., 2021. Angry birds game. https://www.rovio.com/games/angry-birds. https://www.rovio.com/games/angry-birds. [Accessed: December. 13, 2021].

[53] Farahani, A.; Voghoei, S.; Rasheed, K.; and Arabnia, H. R., 2021. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, (2021), 877–894.

[54] Feeney, P.; Schneider, S.; Lymperopoulos, P.; Liu, L.; Scheutz, M.; and Hughes, M. C., 2022. NovelCraft: A dataset for novelty detection and discovery in open worlds. *arXiv preprint arXiv:2206.11736*, (2022).

[55] Felix Haase, D. W., 2021. Bambird 2020. https://github.com/dwolter/BamBirds. https://github.com/dwolter/BamBirds. [Accessed: August. 25, 2021].

[56] Fernández, F. and Veloso, M., 2006. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 720–727.

[57] Ferreira, L. and Toledo, C., 2014. A search-based approach for generating angry birds levels. In *Proceedings of the 9th IEEE International Conference on Computational Intelligence in Games*, CIG'14 (Dortmund, Germany, 2014).

[58] Firestone, C., 2020. Performance vs. competence in human–machine comparisons. *Proceedings of the National Academy of Sciences*, 117, 43 (2020), 26562–26571.

[59] Gamage, C.; Pinto, V.; Xue, C.; Stephenson, M.; Zhang, P.; and Renz, J., 2021. Novelty Generation Framework for AI Agents in Angry Birds Style Physics Games. In *2021 IEEE Conference of Games, COG 2021*.

[60] Gamage, C.; Pinto, V.; Xue, C.; Zhang, P.; Nikonova, E.; Stephenson, M.; and Renz, J., 2023. Novphy: A testbed for physical reasoning in open-world environments. *Artificial Intelligence (under review)*, (2023).

[61] Geirhos, R.; Jacobsen, J.-H.; Michaelis, C.; Zemel, R.; Brendel, W.; Bethge, M.; and Wichmann, F. A., 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2, 11 (2020), 665–673.

[62] Gilden, D. L. and Proffitt, D. R., 1994. Heuristic judgment of mass ratio in two-body collisions. *Perception & Psychophysics*, 56, 6 (Nov. 1994), 708–720. doi:10.3758/BF03208364. http://link.springer.com/10.3758/BF03208364.

[63] Girdhar, R. and Ramanan, D., 2020. Cater: A diagnostic dataset for compositional actions and temporal reasoning. In *ICLR*.

[64] Goel, S.; Shukla, Y.; Sarathy, V.; Scheutz, M.; and Sinapov, J., 2022. Rapid-learn: A framework for learning to recover for handling novelties in open-world environments. *arXiv preprint arXiv:2206.12493*, (2022).

[65] Goel, S.; Tatiya, G.; Scheutz, M.; and Sinapov, J., 2021. NovelGridworlds: A benchmark environment for detecting and adapting to novelties in open worlds. *International Foundation for Autonomous Agents and Multiagent Systems, AAMAS*, (2021).

[66] Goertzel, B. and Pennachin, C., 2007. *Artificial general intelligence*, vol. 2. Springer.

[67] Grace, K.; Salvatier, J.; Dafoe, A.; Zhang, B.; and Evans, O., 2018. Viewpoint: When will ai exceed human performance? evidence from ai experts. *Journal of Artificial Intelligence Research*, 62 (07 2018), 729–754. doi:10.1613/jair.1.11222.

[68] He, K.; Zhang, X.; Ren, S.; and Sun, J., 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385.

[69] Hespos, S. J. and Baillargeon, R., 2001. Infants knowledge about occlusion and containment events: A surprising discrepancy. *Psychological Science*, 12, 2 (2001), 141–147. doi:10.1111/1467-9280.00324.

[70] Hoerger, M.; Kurniawati, H.; Kroese, D.; and Ye, N., 2023. Adaptive discretization using voronoi trees for continuous pomdps. *The International Journal of Robotics Research*, (2023).

[71] Isaksen, A.; Wallace, D.; Finkelstein, A.; and Nealen, A., 2017. Simulating strategy and dexterity for puzzle games. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 142–149. doi:10.1109/CIG.2017.8080427.

[72] Jafarzadeh, M.; Dhamija, A. R.; Cruz, S.; Li, C.; Ahmad, T.; and Boult, T. E., 2020. A review of open-world learning and steps toward open-world learning without labels. doi:10.48550/ARXIV.2011.12906. https://arxiv.org/abs/2011.12906.

[73] Jafarzadeh, M.; Dhamija, A. R.; Cruz, S.; Li, C.; Ahmad, T.; and Boult, T. E., 2020. A review of open-world learning and steps toward open-world learning without labels. *arXiv e-prints*, (2020), arXiv–2011.

[74] James, S.; Ma, Z.; Arrojo, D. R.; and Davison, A. J., 2020. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5, 2 (2020), 3019–3026.

[75] Jelbert, S. A.; Taylor, A. H.; Cheke, L. G.; Clayton, N. S.; and Gray, R. D., 2014. Using the aesop's fable paradigm to investigate causal understanding of water displacement by new caledonian crows. *PLOS ONE*, 9, 3 (03 2014), 1–9. doi:10.1371/journal.pone.0092895. https://doi.org/10.1371/journal.pone.0092895.

[76] Jo, J. and Bengio, Y., 2017. Measuring the tendency of cnns to learn surface statistical regularities.

[77] Justesen, N.; Torrado, R. R.; Bontrager, P.; Khalifa, A.; Togelius, J.; and Risi, S., 2018. Illuminating generalization in deep reinforcement learning through procedural level generation.

[78] Kaiser, M.; Proffitt, D.; and McCloskey, M., 1985. The development of beliefs about falling objects. *Attention, Perception, and Psychophysics*, 38, 6 (Nov. 1985), 533–539. doi:10.3758/BF03207062.

[79] Keil, F. C., 2003. Folkscience: coarse interpretations of a complex reality. *Trends in Cognitive Sciences*, 7, 8 (Aug. 2003), 368–373. doi:10.1016/S1364-6613(03)00158-X. https://linkinghub.elsevier.com/retrieve/pii/S136466130300158X.

[80] Kejriwal, M. and Thomas, S., 2021. A multi-agent simulator for generating novelty in monopoly. *Simulation Modelling Practice and Theory*, 112 (2021), 102364. doi:https://doi.org/10.1016/j.simpat.2021.102364. https://www.sciencedirect.com/science/article/pii/S1569190X21000770.

[81] Keller, P. W.; Mannor, S.; and Precup, D., 2006. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, 449–456.

[82] Kemp, C. C.; Edsinger, A.; and Torres-Jara, E., 2007. Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, 14, 1 (2007), 20–29. doi:10.1109/MRA.2007.339604.

[83] Khetarpal, K.; Riemer, M.; Rish, I.; and Precup, D., 2020. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, (2020).

[84] Kim, B.; Lee, K.; Lim, S.; Kaelbling, L.; and Lozano-Pérez, T., 2020. Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 9916–9924.

[85] Kingma, D. P. and Ba, J., 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980 (2014). http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14.

[86] Klenk, M.; Piotrowski, W.; Stern, R.; Mohan, S.; and de Kleer, J., 2020. Model-based novelty adaptation for open-world ai. In *International Workshop on Principles of Diagnosis (DX)*.

[87] Knox, W.; Glass, B.; Love, B.; Maddox, W.; and Stone, P., 2012. How humans teach agents: A new experimental perspective. *International Journal of Social Robotics*, 4 (07 2012). doi:10.1007/s12369-012-0163-x.

[88] Kumar, A.; Fu, Z.; Pathak, D.; and Malik, J., 2021. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, (2021).

[89] Lake, B. M.; Ullman, T. D.; Tenenbaum, J. B.; and Gershman, S. J., 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40 (2017), e253. doi:10.1017/S0140525X16001837.

[90] Langley, P., 2020. Open-world learning for radically autonomous agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 13539–13543.

[91] Langley, P., 2022. Constraints on theories of open-world learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, Designing Artificial Intelligence for Open Worlds*.

[92] Langley, P.; Nicholas, D.; Klahr, D.; and Hood, G., 1981. A Simulated World for Modeling Learning and Development. 274–276. Berkeley, CA.

[93] Lawson, R., 2012. Mirrors, mirrors on the wall... the ubiquitous multiple reflection error. *Cognition*, 122, 1 (Jan. 2012), 1–11. doi:10.1016/j.cognition.2011.07.001. https://linkinghub.elsevier.com/retrieve/pii/S0010027711001776.

[94] Lazaridis, A.; Fachantidis, A.; and Vlahavas, I., 2020. Deep reinforcement learning: A state-of-the-art walkthrough. *Journal of Artificial Intelligence Research*, 69 (2020), 1421–1471.

[95] Leslie, A., 1984. Spatiotemporal continuity and the perception of causality in infants. *Perception*, 13 (02 1984), 287–305. doi:10.1068/p130287.

[96] Leslie, A. M., 1994. Tomm, toby, and agency: Core architecture and domain specificity. *Mapping the Mind*, (1994), 119–148. doi:10.1017/cbo9780511752902.006.

[97] Levillain, F. and Bonatti, L. L., 2011. A Dissociation Between Judged Causality and Imagined Locations in Simple Dynamic Scenes. *Psychological Science*, 22, 5 (May 2011), 674–681. doi:10.1177/0956797611404086. http://journals.sagepub.com/doi/10.1177/0956797611404086.

[98] Li, D.; Meng, L.; Li, J.; Lu, K.; and Yang, Y., 2022. Domain adaptive state representation alignment for reinforcement learning. *Information Sciences*, 609 (2022), 1353–1368.

[99] Li, R.; Hua, H.; Haslum, P.; and Renz, J., 2021. Unsupervised novelty characterization in physical environments using qualitative spatial relations. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, vol. 18, 454–464.

[100] Lim, M. H.; Tomlin, C. J.; and Sunberg, Z. N., 2021. Voronoi progressive widening: efficient online solvers for continuous state, action, and observation pomdps. In *2021 60th IEEE conference on decision and control (CDC)*, 4493–4500. IEEE.

[101] Liu, B.; Mazumder, S.; Robertson, E.; and Grigsby, S., 2022. Ai autonomy: Self-initiation, adaptation and continual learning. *ArXiv*, abs/2203.08994 (2022).

[102] Liu, X.-H.; Xue, Z.; Pang, J.-C.; Jiang, S.; Xu, F.; and Yu, Y., 2021. Regret minimization experience replay in off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*. https://openreview.net/forum?id=5AixAJweEyC.

[103] Lolos, K.; Konstantinou, I.; Kantere, V.; and Koziris, N., 2017. Adaptive state space partitioning of markov decision processes for elastic resource management. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 191–194. IEEE.

[104] Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; and Zhang, G., 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31, 12 (2018), 2346–2363.

[105] Mao, J.; Gan, C.; Kohli, P.; Tenenbaum, J. B.; and Wu, J., 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *CoRR*, abs/1904.12584 (2019). http://arxiv.org/abs/1904.12584.

[106] Marcus, G., 2018. Deep learning: A critical appraisal.

[107] Marcus, G., 2020. The next decade in ai: Four steps towards robust artificial intelligence.

[108] McCallum, A. K., 1996. *Reinforcement learning with selective perception and hidden state*. University of Rochester.

[109] McCloskey, M., 1983. Naive theories of motion. *Mental models,*, 14 (2) (1983), 299–324.

[110] McLure, M. D. and Musliner, D. J., 2022. A changepoint method for open-world novelty detection. In *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*, 5329–5332. IEEE.

[111] Menache, I.; Mannor, S.; and Shimkin, N., 2005. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134, 1 (2005), 215–238.

[112] Minecraft, 2022. Minecraft official game. https://www.minecraft.net/en-us.

[113] Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning. In *ICML*.

[114] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, (2013).

[115] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D., 2015. Human-level control through deep reinforcement learning. *Nature*, 518, 7540 (Feb. 2015), 529–533. http://dx.doi.org/10.1038/nature14236.

[116] Molineaux, M. and Dannenhauer, D., 2022. An environment transformation-based framework for comparison of open-world learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence, Designing Artificial Intelligence for Open Worlds*.

[117] Moore, A., 1991. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of (ICML) International Conference on Machine Learning*, 333 – 337. Morgan Kaufmann.

[118] Moore, A., 1993. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Advances in neural information processing systems*, 6 (1993).

[119] Moore, A. W., 1990. Efficient memory-based learning for robot control. Technical report, University of Cambridge.

[120] Muhammad, F.; Sarathy, V.; Tatiya, G.; Goel, S.; Gyawali, S.; Guaman, M.; Sinapov, J.; and Scheutz, M., 2021. A novelty-centric agent architecture for changing worlds. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 925–933.

[121] Munos, R. and Moore, A., 1999. Influence and variance of a markov chain: Application to adaptive discretization in optimal control. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, vol. 2, 1464–1469. IEEE.

[122] Musliner, D. J.; Pelican, M. J.; McLure, M.; Johnston, S.; Freedman, R. G.; and Knutson, C., 2021. Openmind: Planning and adapting in domains with novelty. *Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems*, (2021).

[123] Newcombe, N.; Huttenlocher, J.; and Learmonth, A., 1999. Infants' coding of location in continuous space. *Infant Behavior and Development*, 22, 4 (1999), 483–510. doi:10.1016/s0163-6383(00)00011-4.

[124] Nie, Y.; Williams, A.; Dinan, E.; Bansal, M.; Weston, J.; and Kiela, D., 2020. Adversarial nli: A new benchmark for natural language understanding. In *ACL*.

[125] nikonkate, 2022. Novelty domains. https://github.com/nikonkate/novelty-domains.

[126] Nikonova, E. and Gemrot, J., 2019. Deep q-network for angry birds. *CoRR*, abs/1910.01806 (2019). http://arxiv.org/abs/1910.01806.

[127] Padakandla, S.; KJ, P.; and Bhatnagar, S., 2020. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50, 11 (2020), 3590–3606.

[128] Pan, S. J. and Yang, Q., 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22, 10 (2009), 1345–1359.

[129] Peng, X.; Balloch, J. C.; and Riedl, M. O., 2021. Detecting and adapting to novelty in games. *arXiv preprint arXiv:2106.02204*, (2021).

[130] Perdue, B. M.; Evans, T. A.; and Beran, M. J., 2018. Chimpanzees show some evidence of selectively acquiring information by using tools, making inferences, and evaluating possible outcomes. *PLOS ONE*, 13, 4 (04 2018), 1–20. doi:10.1371/journal.pone.0193229. https://doi.org/10.1371/journal.pone.0193229.

[131] Pinto, V.; Renz, J.; Xue, C.; Zhang, P.; Doctor, K.; and Aha, D. W., 2022. Measuring the performance of open-world AI systems. *Proceedings of the AAAI Conference on Artificial Intelligence, Designing Artificial Intelligence for Open Worlds*, (2022).

[132] Pinto, V.; Xue, C.; Gamage, C. N.; and Renz, J., 2021. The difficulty of novelty detection in open-world physical domains: An application to angry birds. *arXiv preprint arXiv:2106.08670*, (2021).

[133] Prada, R.; Lopes, P.; Catarino, J.; Quitério, J.; and Melo, F. S., 2015. The geometry friends game ai competition. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 431–438. doi:10.1109/CIG.2015.7317949.

[134] Proffitt, D. R.; Kaiser, M. K.; and Whelan, S. M., 1990. Understanding wheel dynamics. *Cognitive Psychology*, 22, 3 (Jul. 1990), 342–373. doi:10.1016/0010-0285(90)90007-Q. https://linkinghub.elsevier.com/retrieve/pii/0010002859090007Q.

[135] Pyeatt, L. D.; Howe, A. E.; et al., 2001. Decision tree function approximation in reinforcement learning. In *Proceedings of the third international symposium on adaptive systems: evolutionary computation and probabilistic graphical models*, vol. 2, 70–77. Cuba.

[136] Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N., 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html.

[137] Ramírez, J.; Yu, W.; and Perrusquía, A., 2022. Model-free reinforcement learning from expert demonstrations: a survey. *Artificial Intelligence Review*, (2022), 1–29.

[138] Ramos, G. d. O. and Bazzan, A. L., 2016. On estimating action regret and learning from it in route choice. In *Proceedings of the Ninth Workshop on Agents in Traffic and Transportation (ATT-2016)*, 1–8. CEUR-WS. org New York.

[139] Renz, J.; Ge, X.; Gould, S.; and Zhang, P., 2015. The angry birds AI competition. *AI Magazine*, 36 (06 2015), 85–87. doi:10.1609/aimag.v36i2.2588.

[140] Renz, J.; Ge, X.; Stephenson, M.; and Zhang, P., 2019. AI meets angry birds. *Nature Machine Intelligence*, 1, 7 (2019), 328–328.

[141] Renz, J.; Ge, X.; Verma, R.; and Zhang, P., 2016. Angry birds as a challenge for artificial intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30.

[142] Riochet, R.; Castro, M. Y.; Bernard, M.; Lerer, A.; Fergus, R.; Izard, V.; and Dupoux, E., 2020. Intphys 2019: A benchmark for visual intuitive physics understanding. *ArXiv*, abs/1803.07616 (2020).

[143] Rovio Enternainment. Angry birds game. https://www.rovio.com/games/angry-birds. [Accessed: December. 07, 2022].

[144] Sanborn, A.; Mansinghka, V.; and Griffiths, T., 2013. Reconciling intuitive physics and newtonian mechanics for colliding objects. *Psychological review*, 120 (03 2013). doi:10.1037/a0031912.

[145] Saxe, R. and Carey, S., 2006. The perception of causality in infancy. *Acta Psychologica*, 123, 1 (2006), 144–165. doi:https://doi.org/10.1016/j.actpsy.2006.05.005. https://www.sciencedirect.com/science/article/pii/S0001691806000710. Michotte's heritage in perception and cognition research.

[146] Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D., 2015. Prioritized experience replay. http://arxiv.org/abs/1511.05952. Cite arxiv:1511.05952Comment: Published at ICLR 2016.

[147] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O., 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347 (2017).

[148] Senator, T., 2019. Science of artificial intelligence and learning for open-world novelty (SAIL-ON). https://www.darpa.mil/program/science-of-artificial-intelligence-and-learning-for-open-world-novelty.

[149] Siegler, R. S., 1976. Three aspects of cognitive development. *Cognitive Psychology*, 8, 4 (Oct. 1976), 481–520. doi:10.1016/0010-0285(76)90016-5. https://linkinghub.elsevier.com/retrieve/pii/0010028576900165.

[150] Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al., 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529, 7587 (2016), 484–489.

[151] Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al., 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362, 6419 (2018), 1140–1144.

[152] Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al., 2017. Mastering the game of go without human knowledge. *nature*, 550, 7676 (2017), 354–359.

[153] Sinclair, S.; Wang, T.; Jain, G.; Banerjee, S.; and Yu, C., 2020. Adaptive discretization for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33 (2020), 3858–3871.

[154] Sinclair, S. R.; Banerjee, S.; and Yu, C. L., 2019. Adaptive discretization for episodic reinforcement learning in metric spaces. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3, 3 (2019), 1–44.

[155] Sinclair, S. R.; Banerjee, S.; and Yu, C. L., 2023. Adaptive discretization in online reinforcement learning. *Operations Research*, 71, 5 (2023), 1636–1652.

[156] Smaldone, R. A.; Thompson, C. M.; Evans, M.; and Voit, W., 2017. Teaching science through video games. *Nature chemistry*, 9, 2 (2017), 97–102.

[157] Smith, K. A. and Vul, E., 2013. Sources of Uncertainty in Intuitive Physics. *Topics in Cognitive Science*, 5, 1 (Jan. 2013), 185–199. doi:10.1111/tops.12009. https://onlinelibrary.wiley.com/doi/10.1111/tops.12009.

[158] Stephenson, M. and Renz, J., 2017. Generating varied, stable and solvable levels for angry birds style physics games. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 288–295. IEEE.

[159] Stephenson, M.; Renz, J.; Ge, X.; and Zhang, P., 2018. The 2017 AIBIRDS competition. *ArXiv*, abs/1803.05156 (2018).

[160] Stern, R.; Piotrowski, W.; Klenk, M.; de Kleer, J.; Perez, A.; Le, J.; and Mohan, S., 2022. Model-based adaptation to novelty in open-world ai. *Proceedings of the 32nd International Conference on Automated Planning and Scheduling, Bridging the Gap Between AI Planning and Reinforcement Learning*, (2022).

[161] Taylor, A.; Hunt, G.; Medina, F.; and Gray, R., 2009. Do new caledonian crows solve physical problems through causal reasoning? *Proceedings of the Royal Society B: Biological Sciences*, 276, 1655 (2009), 247–254. doi:10.1098/rspb.2008.1107. https://royalsocietypublishing.org/doi/abs/10.1098/rspb.2008.1107.

[162] Taylor, M. E. and Stone, P., 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10, 7 (2009).

[163] Tian, S.; Li, L.; Li, W.; Ran, H.; Ning, X.; and Tiwari, P., 2024. A survey on few-shot class-incremental learning. *Neural Networks*, 169 (2024), 307–324.

[164] Uther, W. T. and Veloso, M. M., 1998. Tree based discretization for continuous state space reinforcement learning. *Aaai/iaai*, 98 (1998), 769–774.

[165] Valenza, E.; Leo, I.; Gava, L.; and Simion, F., 2006. Perceptual completion in newborn human infants. *Child Development*, 77, 6 (2006), 1810–1821. http://www.jstor.org/stable/4139276.

[166] Van de Ven, G. M. and Tolias, A. S., 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, (2019).

[167] Van Hasselt, H.; Guez, A.; and Silver, D., 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 30.

[168] Velasquez, A., 2023. Transfer from imprecise and abstract models to autonomous technologies (tiamat). *Defense Advanced Research Projects Agency (DARPA) Program Solicitation*, (2023).

[169] Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al., 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 7782 (2019), 350–354.

[170] Wang, S., 2004. Young infants reasoning about hidden objects: evidence from violation-of-expectation tasks with test trials only. *Cognition*, 93, 3 (2004), 167–198. doi:10.1016/j.cognition.2003.09.012.

[171] Wang, T. J., 2017. Ai angry birds eagle wing. https://github.com/heartyguy/AI-AngryBird-Eagle-Wing. https://github.com/heartyguy/AI-AngryBird-Eagle-Wing. [Accessed: July. 31, 2021].

[172] Wang, Z.; de Freitas, N.; and Lanctot, M., 2015. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581 (2015). http://arxiv.org/abs/1511.06581.

[173] Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N., 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

[174] Wang, Z.; Williamson, R. A.; and Meltzoff, A. N., 2018. Preschool physics: Using the invisible property of weight in causal reasoning tasks. *Plos One*, 13, 3 (2018). doi:10.1371/journal.pone.0192054.

[175] Waugh, K.; Morrill, D.; Bagnell, J.; and Bowling, M., 2015. Solving games with functional regret estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29.

[176] Wechsler, D., 1958. The measurement and appraisal of adult intelligence. *Williams & Wilkins Co*, (1958).

[177] Whiteson, S., 2006. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7 (2006).

[178] Wikimedia Foundation, . Multiple-vehicle collision. https://en.wikipedia.org/wiki/Multiple-vehicle_collision. [Accessed: January. 04, 2023].

[179] Wikimedia Foundation, . Self-checkout. https://en.wikipedia.org/wiki/Self-checkout. [Accessed: January. 04, 2023].

[180] Wilcox, T. and Chapa, C., 2004. Priming infants to attend to color and pattern information in an individuation task. *Cognition*, 90, 3 (2004), 265–302. doi:10.1016/s0010-0277(03)00147-1.

[181] Witty, S.; Lee, J. K.; Tosch, E.; Atrey, A.; Clary, K.; Littman, M. L.; and Jensen, D., 2021. Measuring and characterizing generalization in deep reinforcement learning. *Applied AI Letters*, 2, 4 (2021), e45.

[182] Xing, J.; Nagata, T.; Chen, K.; Zou, X.; Neftci, E.; and Krichmar, J. L., 2021. Domain adaptation in reinforcement learning via latent unified state representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 10452–10459.

[183] Xue, C.; Nikonova, E.; Zhang, P.; and Renz, J., 2023. Rapid open-world adaptation by adaptation principles learning. *Submitted to Neurocomputing*, (2023).

[184] Xue, C.; Pinto, V.; Gamage, C.; Nikonova, E.; Zhang, P.; and Renz, J., 2023. Phy-q as a measure for physical reasoning intelligence. *Nature Machine Intelligence*, 5, 1 (2023), 83–93.

[185] Xue, C.; Pinto, V.; Zhang, P.; Gamage, C.; Nikonova, E.; and Renz, J., 2022. Science birds novelty: An open-world learning test-bed for physics domains. In *Proceedings of the AAAI Spring Symposium on Designing AI for Open-World Novelty, Palo Alto, CA, USA*, 21–23.

[186] Yi*, K.; Gan*, C.; Li, Y.; Kohli, P.; Wu, J.; Torralba, A.; and Tenenbaum, J. B., 2020. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HkxYzANYDB.

[187] Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S., 2020. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, 1094–1100. PMLR.

[188] Zambaldi, V.; Raposo, D.; Santoro, A.; Bapst, V.; Li, Y.; Babuschkin, I.; Tuyls, K.; Reichert, D.; Lillicrap, T.; Lockhart, E.; et al., 2018. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, (2018).

[189] Zeng, Z. and Davis, E., 2022. Physical reasoning in an open world. *Advances in Cognitive Systems (ACS) Conference*, (2022).

[190] Zhao, W.; Queralta, J. P.; and Westerlund, T., 2020. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, 737–744. IEEE.

[191] Zhu, Z.; Lin, K.; and Zhou, J., 2020. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, (2020).