IMPERIAL

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Neurosymbolic Policies for Interpretable and Generalisable Reinforcement Learning

Author:
Luka Kotur Corliss

Supervisor:
Francesco Belardinelli

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence and Machine Learning of Imperial College London

September 2024

Abstract

The thesis explored the integration of symbolic regression into reinforcement learning, to create a neurosymbolic model, specifically the Proximal Policy Optimization (PPO) algorithm, to improve interpretability and generalizability. The novel algorithms SLMA, SPUA, and SLPU were developed inline with these goals and tested across various Gymnasium environments, which showcased enhancements in both interpretability and generalisation compared to standard PPO, albeit with some trade-offs in performance.

The research methodology involved modifying the PPO algorithm to periodically apply symbolic regression in order to extract mathematical expressions that approximate policy behaviour, using that approximation to influence the learning. This approach allows continuous influence on the learning process, giving benefit in guiding the learning process. Extensive experiments were conducted to evaluate the algorithm's performance, interpretability, and generalization capabilities compared to standard PPO.

The results indicate that incorporating symbolic regression during learning can steer reinforcement learning models towards more interpretable and generalisable solutions. SPUA showed the most substantial gains in interpretability, while SLMA provided a better balanced algorithm between interpretability, generalizability, and performance. The combined SLPU algorithm demonstrated potential in leveraging the strengths of both approaches, provided that the system is given the proper modification for asynchronous use of its SLMA and SPUA features.

However, this research also revealed several challenges, particularly in computational complexity and the need for more refined symbolic regression techniques tailored to reinforcement learning contexts. Despite these challenges, the potential benefits of safety, explainability, and adaptability make this a promising direction for future research.

As AI systems become increasingly prevalent in critical applications, the need to understand to be able to trust their decision-making processes will only grow in importance. This work represents a step towards more interpretable and generalisable reinforcement learning models and contributes to the responsible development of AI technologies. The findings pave the way for further exploration of neurosymbolic approaches in reinforcement learning, aiming to bridge the gap between high-performing but opaque neural models and interpretable symbolic systems.

Acknowledgments

First to my supervisor Francesco Belardinelli, for our weekly meetings and feedback over the course of the project helped immensely in the final product.
Second to Titus Buckworth for his early help in understanding how to leverage existing environments and discussions in general neurosymbolic topics. Then to my two brilliant people who gave me feedback on suggested improvements for clarity and grammar, Gordian Gruentuch and my father, Steven Corliss.

Contents

Chapter 1

Introduction

Recent advancements in artificial intelligence (AI) systems have presented unique opportunities to the scientific community in their ability to find patterns and solve problems that would not be possible through traditional research [122]. Examples present in the medical field [12], as it recently was in Covid-19 [105], and increasingly impactful in viable autonomous vehicle solutions [82, 69], showcase this.

From these AI-assisted advances there is great potential for the further development of systems that aid in the research and deployment of technologies that can increase the people's well-being, highlighted in both physical and economic [71, 23] aspects. This growing importance in daily life demonstrates the need of researching effective and safe AI systems, as these systems could easily misdiagnose illness or cause vehicle accidents [121] in safety critical situations. As while autonomous vehicles are found to cause ten percent fewer accidents, cases such as when an Uber self-driving car killed a pedestrian [121] show a critical lack of robustness and in turn safety, the cornerstone of this investigation.

These created systems do have advantages, as they outperform human users [121] and even with their limitations the systems are successful, with the most successful using Neural Networks (NNs) [44]. The use and adaptation of NNs has shown effectiveness and general applicability in many environments and situations [33, 109, 1], arising from their perception due to inductive reasoning [1]. NNs present with key issues, a non-exhaustive list would include high resource consumption, high data dependency, a lack of interpretability [1] and, as increasingly seen in the literature, the lack of generalisation primarily due to over-fitting [125], meaning that the model could not perform well outside of training data, causing issues in real world use cases.

The introduction of Symbolic AIs [96] solve some of these issues. Deductive reasoning, for example, addresses the lack of interpretability and partially generalisation by building on predefined rules. A more generalized system can be fundamental, as it relates to how well the artificial intelligence system applies to knowledge outside of its training data, which is an omnipresent concept in AI and is seen as a hallmark of human cognition [79]. Symbolic AIs achieve this in a few ways. One way is how they represent outcomes in propositional logic [96], as this format does not allow for the fine-tuning of behaviour. While this can cause effectiveness issues, it allows for a more generalized outcome. Another is the compositional structure, where there are clear sub-tasks that contribute to the overall goal [41] that naturally develops with symbolic systems, due to it effectively being a combination of rules. This again lends itself to the generalisation, but additionally the potential scalability of the program [68], by having a complex systems built on reusable simpler parts.

Their limitations and benefits can characterize the difference between the two systems. NNs can potentially memorise the data given [36] and do not learn the overall behaviour, with the outcome being a distributed representation [58]. However, symbolic AIs are limited by the lack of effectiveness due to no true learning and only being able to handle certain problems due to the data structuring requirements[50, 96].

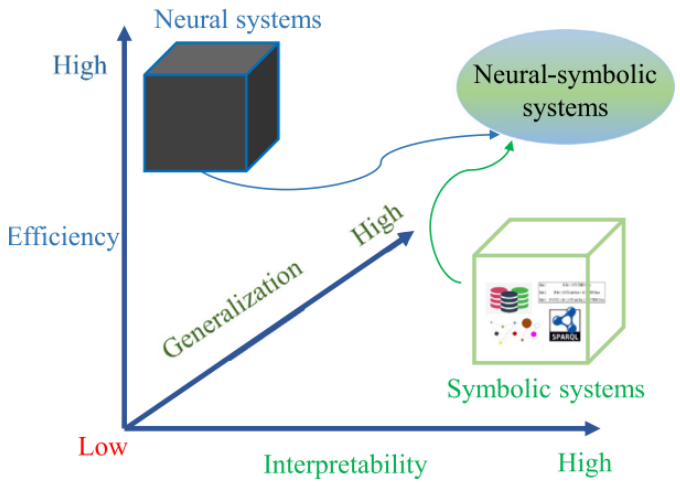

Figure 1.1: Symbolic after Learning

Neurosymbolic (NS) models [124] have the potential to address some of these key issues through the combination of both neural and symbolic networks. NS models additionally gain the benefit of generalisation as seen in Figure 1.1, again giving the system a greater degree of safety, as it can adapt to challenges outside of its training distribution more effectively. This overcomes the limitations that the individual models present, allowing the model to perform both perception and reasoning tasks [76, 30, 72, 114, 73].

Additionally, interpretability in effective systems gives the ability to audit the outcome of the model in more complex

use cases, in order have an understanding of how the system functions in specific applications. Especially important for safety-critical applications and vital when the outcome of guessing wrong is too great, such as in the aforementioned case of autonomous vehicle crashes, which could have been avoided if a greater understanding of the system was obtained. This thesis takes a step towards that outcome and solving the problem of interpretability and generalisability in AI.

1.1 Problem Statement

The research problem addressed here revolves around the issue of safety, specifically in reinforcement learning (RL) [39] due to the previously mentioned key safeguard concerns in autonomous systems and how these systems largely use RL [77, 46].

To solve these safety concerns, two key aspects were identified to be addressed: lack of interpretability in the found policies and the generalizability [77] of said policies. The research carried out here will attempt to address both of these problems by modifying RL systems as there is a clear space for improvement around existing systems in terms of safety and its derivatives of generalisation and interpretability, as defined in the background and in current literature [39, 114, 6].

Interpretability of policies refers to how well an AI solution is understood and in turn the higher degree of safety that can be guaranteed, as the architect has a greater understanding of how the model would act in different situations, allowing for them to be anticipated. Generalisability factors into safety. If a model does not successfully generalise it can not perform actions outside of training data and cannot effectively be used in real-world situations where there will almost always be actions outside of direct training distributions, such as the previous example of self-driving cars.

Neurosymbolic models allow these issues to be tackled in unison. The critical gap, outside of the ongoing challenge of greater generalisation to allow for real world agent interactions, in reinforcement learning is a lack of interpretability in the extracted symbolic policies. As the extracted policies are not always very represented of the found solution, due to the differences between how the learning's solution is represented and representation of the policy extracted.

Even with these issues the reason behind neurosymbolic AI have not been more prevalent in reinforcement learning, even though they do aid in solving core issues, is the complexity of implementation. This has led to many neurosymbolic methods that have not yet been explored, meaning that current options are likley

non-optimal due to under exploration. Again this gives additional motivation for the subsequent investigation in the combination methods to produce a neurosymbolic system.
As with any AI methodology, however, there are advantages and disadvantages to any AI system as showcased in 1.1. As laid previously out with Neural Networks and Symbolic AI and is true again with neurosymbolic AI having greater complexity in both its computational overhead and integration complexity. Both will be explored in relation to their severity in the various model architectures explored.

1.2 Research Objectives

The model architecture created in this investigation explores continuous policy extraction throughout the run to influence the learning towards generalisation and interpretability. This led to the main objective of integrating policy extraction into a reinforcement learning algorithm, and from this, the following accompanying objectives were identified.

(1) To develop an original taxonomy to categorize explored neurosymbolic models, with the systems being put into the following categories: Symbolic in Data, Symbolic during Learning, and Symbolic after Learning. (Section: 3.1)

(2) Review existing methods to gain implementational insight, additionally drawing from their testing methods and evaluation to gain further points of comparison to existing systems (Section: 3.2)

(3) Design and implement novel algorithms that use policy extraction in the learning process, for the goal of generalisation and influencing the learning program for greater interpretability , through the combination of two AI methodologies, resulting in the Symbolic Loss Modification Algorithm (SLMA) 4.1 and Symbolic Policy Update Algorithm (SPUA) 4.2. (Sections: 4, 5)

(4) Create tests and innovative metrics that validate the efficacy of the created algorithms, in terms of both effect on the program and resulting outputs for both overall effectiveness, generalisability and interpretability of the solution. (Section: 6)

(5) Analyse the results to evaluate the algorithms and draw conclusions on what situations and applications they can be successfully implemented. (Section: 7)

The organisation of the paper and its sections also follow this structure and have related headings.

1.3 Scope and Limitations

The scope in which these objectives will be applied is constrained due to the uncontrollable magnitude of situations and environments to which neurosymbolic models could be applied to, ranging from stock market prediction to video game control.
The environments investigated here are the finite number of environments contained in Gymnasium [28], where the agent has to solve an environment based on the challenges contained within it. The tests span from path finding to stabilisation. This allows for both a diverse testing set, easily reproducible results and comparable metrics to the base system in literature due to their prevalent use.
Due to prevalence of use, as well as general efficacy, the algorithm Proximal Policy Optimisation (PPO) [102] was chosen to provide the investigative foundation. The reasoning for choosing a more prevalent RL algorithm is that it allows for more literative comparisons. Additionally PPO is one of the strongest RL models, providing an effective baseline. The baseline in this case consists largely around generalisability and interpretability, striking a balance between the two, as these are the key areas of the investigation rather then just the overall performance, as is broadly the case with neurosymbolic AI[52] due to the loss of fine tuning.
Due to the novelty of the field, solutions from distillation [86] to hybrid appraoches [86], even for the same core problem, and there is not much in the form of standardization in method. These variations increase complexity of exploration, as there are more implementations to consider. To constrain the area of exploration symbolic regression (SR) [24] being chosen for the method for policy extraction, due to its effectiveness and due to its recent development is under researched.
By using SR to create a neurosymbolic AI various application domains in reinforcement learning, especially where both interpretability and generalisability are considered important elements, can be explored. The most direct use case could be safety critical robotics, such as disaster relief or exploratory vessels as these have the most similarity to the environments used in the creation of the system, otherwise having possible transference issues to other application domains in real world situations.

Model	Processing Method	Knowledge Representation	Main Algorithm	Advantages	Disadvantages
Neural Network	Inductive Learning	Distributed Representation	Back-propagation	-Finds patterns in data -Strong Robustness -Handles unstructured data	-Lacks interpretab -Large data requirement -Weak generalisat
Symbolic AI	Deductive Reasoning	Logical Representation	Logic inference engines	-Good generalization -Interpretable -Logically driven	-Cannot handle unstructured data -Weak robustness -Limited learning
Neurosymbolic AI	Inductive and Deductive	Logical Representation	Hybrid Models	-Pulls advantages from both	-Complex to integ -Computational complexity

Table 1.1: Comparison of AI Methods

Chapter 2

Technical Context

The technical context takes a closer look at the systems that the investigation will be applying to achieve the goal of creating a more interpretable system with greater generalisability.
This consists of broadly four areas, the PPO algorithm, Symbolic Regression (SR), what is neurosymbolic AI and the environments used for the investigation.

2.1 Reinforcement Learning

First, the overall concept of reinforcement learning must be understood to understand PPO, which is built on reinforcement learning. Reinforcement Learning [57, 116] is a branch of machine learning that deals with providing a learning loop, where the agent interacts with an environment and learns what are the most effective actions for reward maximisation, based upon the consequences of the actions.
In Figure 2.1, the learning loop is demonstrated, where the model learns based upon its interactions with the environment and uses those interactions to update the policy of the model, effectively mapping the action that the agent should take given the current state of the environment.
An example would be that the agent is a car, the environment is a hill, and the policy instructs the car to get over the hill based on the features of the envi-

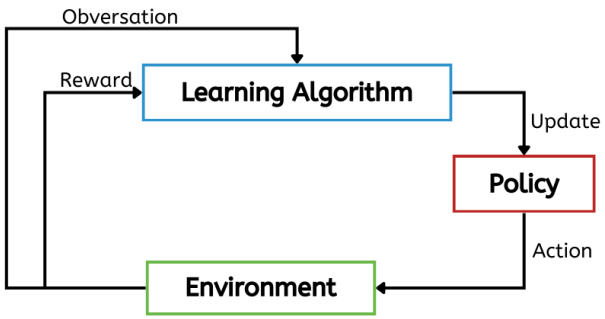

Figure 2.1: Reinforcement Learning System

ronment that it perceives. Then based off how well the agent gets over the hill informs the reward, which together with the observation, gives an indication of what elements of the policy worked and what did not, this leads to alterations through the learning algorithm of the policy and the next iteration of testing. However the learning is influence by a number of factors, such as the learning rate, the batch size, but especially the learning algorithm itself.

2.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [102] is an effective actor-critic reinforcement [45] learning algorithm. An actor-critic works by having an actor which learns policies, with the value of those policies being decided by the critic, which evaluates how positive of an outcome those actions have. Both of these inform each other in a learning cycle, incrementally improving the policies and therefor the actions the agent takes.
PPO was developed as an advancement over previous gradient-based methods [34, 103], working by navigating a multi-dimensional optimization landscape, aiming to find the point that corresponds to the lowest possible loss. The reason for PPO's development against pre-existing methods was to limit weaknesses in terms of learning instability. This causes safety issues due to how large policy updates can lead to implementation of detrimental policies due to unexplored regions and the computational complexity of the models.
The development path of PPO started with Deep Q Networks (DQN) [34] in 2014. However, while effective, the learning instability and the lack of a safe region of exploration led to the development of Trust Region Policy Optimization (TRPO) [103] building off of DQN. However, TRPO had its own issues, namely in the models computational complexity [32], which lead to the development of PPO. PPO is more computationally efficient while retaining the benefits of a safe region of exploration. Safety in exploration is another reason for the choice in PPO, as this research is taking a safety first approach and builds on PPO as PPO built on TRPO or TRPO built on DQN.

2.2.1 PPO Method

The safety comes from PPO having stable policy updates in the objective function. The objective function is a maximisation function that dedicates the loss L_t, representing the difference from gained value to the idealized value when taking an

action a_t in a state s_t, for each policy r update, allowing for improvement by decreasing the respective loss. The loss value is gained from the learning process, but is centred around the value found for the policy by the critic in the actor critic system and the policy ratio $r(\theta)$ between the old and new policies probabilities in various action states. The value is validated by using multiple t samples N, with the variables for the section are present in 9.3.1.
The objective function is built from the truncation of the TRPO objective function 9.1, this being one of the main ways the computational complexity was reduced, and comes in two main forms.

$$L_t = \frac{1}{N}\sum_{t=1}^{N}\left[\min\left(r(\theta)A_{\pi_{\text{old}}}(s_t,a_t)clipped(r_t(\theta))\right)\right] \tag{2.1}$$

One form is the restriction applied to $r_t(\theta)$ in equation 2.1, where the clipping function does not allow for policy updates that deviate too much from the current policy, but still giving space for exploration [116]. This is dictated by the hyper parameter, a value that is set prior to learning, ϵ. While this slows down learning it also provides stability by preventing excessively large updates, again aiding safety.

$$r(\theta), 1-\epsilon, 1+\epsilon)A_{\pi_{\text{old}}}(s_t,a_t) \tag{2.2}$$

The other form of restriction that PPO employs is the use of Advantage A, as seen in equations 2.1 and 2.2. Advantage is a central part of the actor Q critic V method where the algorithm looks at how beneficial it is to take an action compared to the average action in a particular state. This prevents extreme negative actions and allows faster learning of more beneficial policies.

$$A(s,a) = Q(s,a) - V(s) \tag{2.3}$$

However, the equation for the Advantage Estimation 2.3 [101] is commonly improved by reducing the variance and maintaining low bias. This allows for the true relationships to be captured and not just the training data, which in turn providing greater stability and generalisation to the gradient method, in this case PPO.
The Advantage Estimation does this by computing the weighted sum of the temporal differences δ_{t+l}, which gives the difference between the true return and the expected return from the value function V. This provides a smoother function outcome due to the reduced variance by using information from the next steps l reward and existing estimates. Even though bias is increased by the use of previous information this is counteracted by the benefits of a reduced variance. To control

this two hyper-parameter variables, set before learning that influence the learnings behaviour, are introduced, a discount factor for how much future rewards are considered γ, and trade-off between bias and variance λ.

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{2.4}$$

This results in the Generalized Advantage Estimation equation 2.4 [101], with δ_t representing the temporal difference, further represented by the logic $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$. Other stability methods can be found with the regularisation of the outcomes and using mini-batches, both of which reduce the noise present in the data [101]. Consequently, this aides the problem of overfitting and, in turn, enhances the generalisability of the program while increasing the stability of the overall learning process due to being less sensitive to changes in the training data, leading to less complex and more safe interoperable outcomes.
Stability, and therefore a degree of safety due to not exploring overtly harmful regions, is additionally gained through the use of A2C actor-critic system instead of A3C [45]. While A3C can have greater utilisation of resources A2C has greater stability in using synchronous updates by using larger batches which reduces variance. How the model utilizes resources is also evident with the online policy method [102, 100], meaning it can continuously interacts with the environment and gains additional information. This becomes important when the environment can change and leading to necessary modification of the model, and increasing its generalisability. These come together to form the PPO algorithm 5, gained from Stable Baselines [91], that is capable of solving the agent-based problems and being the most effective model for the problems explored.

2.3 Symbolic Regression

2.3.1 Background

First created in the early 90s, SR has been used for a wide range of tasks ranging from material science [117] creation to the discovery of novel physics equations [25]. SR works as a genetic algorithm [61] to discover novel relationships in data. This works by having parent inform the generation of children through mutations and combinations, with the children then becoming the new parents.
This means that as genetic algorithms evolved so did SR, especially with the further integration of other machine learning techniques [70]. Particularly consequential

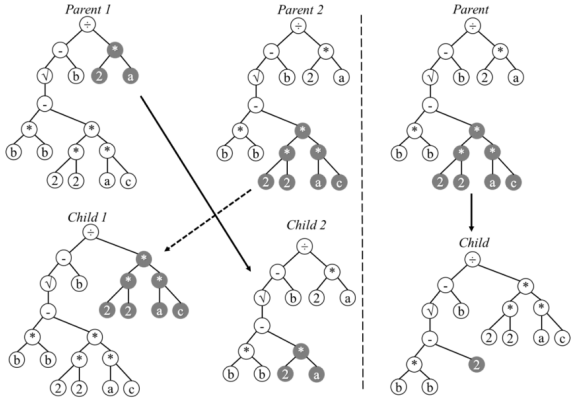

Figure 2.2: Symbolic Regression Diagram [26]

with neural networks, leading to more complex problem solving and a reduction in human intervention, allowing for greater freedom of learning. This makes SR a machine learning method that is used to pull mathematical expression from data, such as the previously mentioned fields of material and physics. However, this study presents the first application of SR to RL, by using environmental features, such as position, as variables to construct the equation equation from the model. To understand how this integration works, it is necessary to explain the method.

2.3.2 Method

As a genetic algorithm, SR uses previous generations selected based of how closely they resemble the data or fitness, to create further permutations.

The method displayed in 2.2 works by having a tournament style system that pits the permutations against each other, so that the fitness of each equation is tested and compared. Each new generation takes from the previous one and randomly modifies a section of it. In the Figure 2.2, there are only two parents for ease of understanding, however, there are realistically an N number of parents in a subset, as decided by balancing the computational efficiency of having multiple ongoing parents with the effectiveness drop off by including parents with lower fitness scores. The subset is continuously informed and refined as algorithms are retained or dropped, as decided by the fitness score, until the pre-decided number of iterations or acceptable loss is reached. This entire process is then condensed into algorithm 1 for ease of understanding how these various steps fit together.

Algorithm 1 Symbolic Regression Algorithm

1: **Input:** Dataset $D = (X, y)$, max generations G, population size N, crossover probability p_c, mutation probability p_m
2: **Output:** Best expression E^* approximating y given X
3: Initialize population P_0 with random expressions
4: **for** $g = 1$ to G **do**
5: Evaluate fitness of individuals in P_g
6: Select top individuals for reproduction
7: Create new population via crossover and mutation
8: **end for**
9: **Return:** Best individual E^* found

SR won't necessarily find the perfect method as a brute force method would. Due to the number of operators and permutations, however, the search would be uncontrollably infinite to try out every possible combination [24]. This basic SR method can be improved, however, with the method changes found in PYSR [24]. These changes come in three main forms, simulated annealing, independent evolution, equation simplification, which provides a good balance between effectiveness and interpretability compared to other SR methods which were explored [110].
Simulated annealing [59] works by rejecting certain permutations based on a temperature measure. This helps the program to avoid premature convergence to local minima which would limit the program's effectiveness due to not finding the optimal solution.

$$p = \exp\left(\frac{L_F - L_E}{\alpha T}\right) \tag{2.5}$$

This works by varying the temperature T in 2.5 throughout the run to allow for both periods of diversity and direct optimisation, as regulated by the hyperparameter α. This balances both exploration and effectiveness of the solution by comparing the fitness of the new mutation L_F to the original L_E, as summarized here 9.3.2. In terms of the algorithm 1, this step takes place in the creation of new population.
Another method that aids in the diversity of solutions is independent evolution, where genetic algorithms are allowed to evolve separately from each other. These evolutionary islands however exchange aspects through a migration process. In relation to algorithm 1 this takes place in how the populations are initialized, with multiple for loops running concurrently, with the crossover and mutation stage occasionally pulling from other evolutionary islands. This genetic diversity aids

the program in not getting stuck in a local optimum through the sharing of good solutions, aiding the overall population group.
The program then has to work to make sure these aspects of diversity do not harm effectiveness by overextending the computational complexity. This is mitigated by optimising the program. One approach is to optimise the constants present in the algorithm [81], making them more emblematic to the presented data by being effective methods for the parameter search of candidate models. The main way this is done is through BFGS [20]. This saves on computational search compared to the nominal symbolic regressive method.
The other way the program reduces computational overhead is by simplifying the equation.

$$\ell(E) = \ell_{\text{pred}}(E) \cdot \exp(\text{frecency}[C(E)]) \tag{2.6}$$

This is done by penalizing complexity, as measured by the number of operators and variables present, as seen in equation 2.6 with Complexity $C(E)$. The equation $l(E)$ is used to adjust the predicted loss $\ell_{\text{pred}}(E)$ associated at data point E, taking place in the evaluation stage of the algorithm 1.
The complex penalty 2.6 also looks at the frequency and recency. Seen in the equation 2.6 as combination word frecency, which means that as there are more of the same high complexity arguments and the newer they are, the more they are penalised, leading to fewer complex outcomes. A lower complexity also aids how interoperable the outcomes are, aiding the purposes of the investigation in interpretability. The program also goes into to simplifying the equation by using mathematical equivalences. Some complex outcomes can be represented by more simplified versions of themselves, such as how $x*x-x*x$ would simplify to 0, aiding the programs run and the resulting interpretability of the solution. However, this is only done occasionally in the program, as forcing constant simplification could detriment the program in finding the optimal solution, as a simplifiable outcome could be intermediate step.
Outside of these core features the program is also extremely customisable, allowing custom losses and operators to be given, giving much more flexibility in the utility and use of the program and contributing directly to its use.

2.3.3 Symbolic Regression to Neurosymbolic

The use of SR over a neural solution would give a symbolic outcome. Using this outcome to inform the learning this creates the neurosymbolic method. A neurosymbolic method refers to any method that is the combination of symbolic and

neural systems in order to create a unified output [124], done in this context of equation extraction to enhance the generalisability and interpretability of the model.

2.4 Environments

To test generalisability and interpretability , various environments were explored in Gymnasium [28]. Gymnasium was chosen for a few reasons. As with the use of Stable Baselines [91] the environments can be vectorized, allowing multiple environments to run in parallel, and in turn achieving a faster run time. Moreover, use of Gymnasium facilitated comparison to other related papers.
The present research was guided by literature on the current state of the art regarding environmments used for reinforcement learning [102] [6] and testing various inbuilt environments. These were chosen based upon how diverse the solutions are to each other, the varying complexities of the environments and how many papers apply them in order to gain a baseline output.
Additionally, the environments, like PYSR, are customisable, allowing for the modification of environments to further diversify the problem sets. This is particularly useful in the context of testing the generalisability of the model, where the model can be trained on a variation of the environment and tested on a slight variation to see how the model would perform, such as increased gravity.
The four key environments chosen were CartPole, Pendulum, MountainCar and Bipedal Walker as apart of the investigation, these were chosen based on characteristics of prevalence in related works and diversity in their problem sets environments specific challenges. This section will cover what makes these environments unique, what the problem is and the reward function.

2.4.1 CartPole

The simplest of the environments to solve, this is largely due to the lower number of variables for the system to consider, with it being the only environment where a feature extractor is not present, eliminating the need for an additional neural network to extract key environmental variables for agent learning.
The problem to solve is to balance the pole in the naturally unstable environment, based off the position, velocity, angle and angular velocity. Together with reward function that rewards +1 for every step taken without termination, such as angle is $\pm$ 12° or the position is $\pm$ 2.4, the model learns how to solve the environment using the actions of left and right.

2.4.2 Pendulum

Introduces a more complex environment then CartPol and like the following environments requires feature extraction. While the features cant be defined, as the extraction will vary, the action is the torque or negative-torque applied to the system in order to get the pendulum up right angle θ^2 and then keep it there against the angular velocity θ^2_{dt}. Effectively making it a two stage problem, leading to custom reward function.

$$r = -(\theta^2 + 0.1 \cdot \theta^2_{\text{dt}} + 0.001 \cdot \text{torque}^2) \tag{2.7}$$

In having more moving parts and two different stages a distinct reward equation must be created 2.7 in order to account for how the rewards between the two stages will shift. This is a more challenging problem due to how the model has to effectively learn two different solutions that have to work conjunctively.

2.4.3 MountainCar

MountainCar requires more epochs, learning cycles, and a stronger model compared to the previous two. This comes from how the environment does not give any rewards that indicate the model is coming close to solving the problem, but rather only give a reward based off how quickly the car gets over the hill, giving -1 for every time step. This is problematic as the car has to learn the counter intuitive behaviour that it has to back up the hill behind it to build enough momentum to climb the hill in front of it without a continuous reward to inform it. Completing the problem using the actions of accelerate left, accelerate right and neutral.

2.4.4 Bipedal Walker

The action spaces of Bipedal Walker however are much more complex, they consists of four motors that have values that can range between [1,-1], for maximum forward and backwards torque, where the agent is a walker going over an uneven terrain. The complex action space tests the program in more complex higher dimensional tasks, as there are more moving parts due to the agent having multiple motors and ligaments, necessitating a more multiplex solution. The reward being receiving points for how far the bipdeal walker goes until it is considered to have effectively solved the problem by a fix distance variable.

2.4.5 Environmental Summary

What is seen from the previously described environments is a varied list of different reinforcement learning challenges that each hold their own characteristics, including simple, multi staged, inconsistent rewards and high dimensional control problems. This allows for the desired varied problem set and giving opportunity for the accurate exploration of the novel model.

Chapter 3

Literature Review

3.1 Classification Overview

Neurosymbolic systems can be implemented in various ways [124]. To enable clearer organization and conceptual grouping, a novel categorization system is introduced, classifying these methods into three broad approaches: incorporating symbolic elements before, during, or after neural learning. This categorization highlights the fundamental differences between these approaches. In the figures below, neural components are marked in blue, while symbolic components are in green.

3.1.1 Symbolic in Data

The 'Symbolic in Data' approach refers to systems that begin with symbolic data either provided or created before the learning process starts [9, 21, 75]. These methods involve feeding symbolic data into a neural network, which constrains the network to produce a symbolic output, ultimately resulting in a Neurosymbolic AI [124].

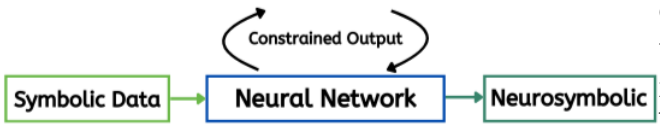

Figure 3.1: Symbolic in Data

The 'Symbolic in Data' approach can be time-consuming prior to training, as the data must be formatted into the proper symbolic format. This process may require significant effort and resources from individuals or the pro-

gram to organize the inputs as required. However, in some cases, data may already be structured symbolically, requiring minimal modification due to the environment of its origin. An example of the methodology, as shown in Figure 3.1, is demonstrated by the neurosymbolic Concept Learner proposed by Mao (2019) [74], which utilizes question-answer pairs, meaning that the data is structured symbolically from the start. The most straightforward method is to manually encode possible interactions using propositional logic [96, 18]. For example, the statement: 'If an autonomous car is approaching an intersection (I) and the traffic light is red (R), then the car must stop (S)' can be expressed in propositional logic as: $(I \wedge R) \rightarrow S$.
The introduction of temporal logic for environment-based agents [93] significantly expands the range of scenarios that must be considered, requiring the program to account for all possible state combinations. This expansion greatly increases computational complexity, leading to a need to automate this process. One effective approach relies on utilising autoencoders [11], which can be implemented in various ways. Among these, Latplan, proposed by Masataro (2018) [8], stands out as one of the most impactful. Latplan leverages deep autoencoders to convert high-dimensional, sub-symbolic data (such as images) into a latent symbolic representation that can be used for classical planning algorithms, creating a map. This approach enables the use of symbolic planning techniques in domains where only raw, unstructured sensory data is available, effectively bridging the gap between subsymbolic and symbolic AI.

3.1.2 Symbolic during Learning

This category encompasses Neurosymbolic methodologies that add symbolic elements during the learning process, in contrast to the approach 'Symbolic in Data,' where symbolic elements are added to the input data. There are two main ways to achieve this:

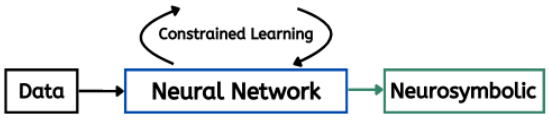

Figure 3.2: Learning with Constrainment

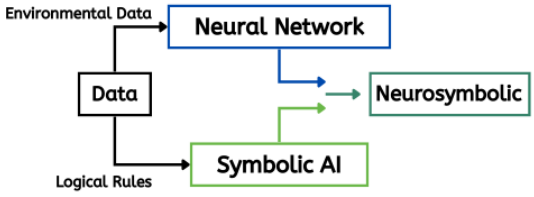

Figure 3.3: Hybrid Model Learning

- The neural network is constrained during learning to produce only symbolic outputs by banning solutions that do not conform to symbolic requirements, seen in Figure 3.2. [120, 85, 124]
- Alternatively, in a hybrid model [14, 124], the tasks are divided between the symbolic and neural networks, with each component handling different aspects of the solution through parallelization, as illustrated in Figure 3.3.

Figures 3.2 and 3.3 both illustrate active symbolic learning occurring concurrently with neural network training. Either method can accept a broader range of data than methods explored so far, making it applicable to the agent-based environmental simulations that are the focus of this paper.

3.1.3 Symbolic after Learning

The 'Symbolic after Learning' category involves methods where a model has completed its learning process and reached a solution. However, these solutions are not symbolic, since an additional step is needed to create a Neurosymbolic model [124, 14]. This distinction can be best demonstrated through the concept of distillation [3] [87], where a trained neural network (the teacher) transfers its learned behavior to an untrained symbolic AI (the student).

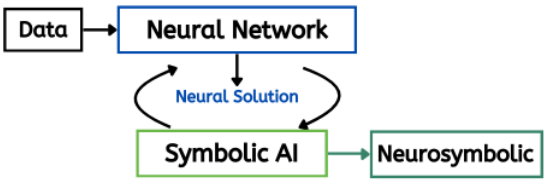

Figure 3.4: Symbolic after Learning

As shown in Figure 3.4, the neural solution guides the symbolic AI to produce a symbolic output that retains neural characteristics, resulting in a neurosymbolic solution. This process leads to some information loss between the neural solution and the neurosymbolic representation. As explored in the introduction, however, this can aid in generalizing the model as not all information captured by the neural model is useful for the model's application.

3.1.4 Comparison of Neurosymbolic Approaches

Neurosymbolic classifications exhibit varying applicability and trade-offs depending on when the symbolic aspect is introduced in relation to the learning process. The timing of symbolic integration - whether before, during, or after learning - significantly influences the system's outcome and computational complexity.

Introducing symbolic elements before or during learning can increase computational complexity due to the continuous conversion between symbolic logic and neural representations. However, this approach may aid in generalization by potentially discovering more generalized solutions. An illustrative example is the concept of shielding, as described by REVEL [6], where symbolic logic informs the system about unsafe regions to guide exploration. While this method increases computational complexity compared to post-learning symbolic application, it enhances safety by continuously guiding the agent's behavior.
Conversely, extracting behavioral equations after learning, as demonstrated by [25], offers lower computational cost since the neural-to-symbolic conversion occurs only once. However, this approach may be less suitable for applications requiring continuous safety assurance, such as shielding, highlighting the situational dependence of these methods.
The effectiveness of each approach is further influenced by factors such as data format and environmental complexity. Symbolic-before-learning approaches require data in a symbolic form, which may not always be available or practical. In novel or complex environments, encoding into a symbolic space may prove challenging, limiting the applicability of such methods.
Ultimately, the choice between these approaches depends on the nature of the task, available data format, computational resources, safety requirements, and desired level of generalization. Understanding these trade-offs is crucial for selecting the most appropriate neurosymbolic approach for a given application in academic research and practical implementations.

3.2 Generalisation and interpretability

Various Neurosymbolic approaches contribute to enhance both generalisation and interpretability. These two aspects as discussed, as discussed in the Introduction, are crucial for providing both AI that is adaptable and can be understood. The proposed classification system facilitates the creation of a taxonomy around models that address these important areas of focus, allowing for a more structured dissection of the related methods.
The representative works presented in Table 3.1 was chosen both for its impact on the areas or area of generalisation and interpretability. A broader overview of the field was chosen in general to inform methodology in achieving greater interpretability and generalisability for the field of Neurosymbolic AI, giving greater context to the report. The primary applications listed in Table 3.1 cover diverse

domains, such as vision, audio, movement, language, and equation extraction.

Representative Work	Learning-Grouping	Neural Network	Primary Application
NS-CL [74]	Symbolic Before	CNN	Vision
PIRL [114]	Symbolic Before	RL	Movement Agent
Energy LLN [43]	Symbolic Before	LLN-RL	Control Agent
Energy DDT [43]	Symbolic During	DDT-RL	Control Agent
ZeroC [118]	Symbolic During	CNN	Vision
NSS [51]	Symbolic During	LSTM	Audio
DooD [67]	Symbolic During	RNN	Vision
REVEL [6]	Symbolic During	RL	Movement Agent
PROLONETS [106]	Symbolic During	RL	Movement Agent
DUA [80]	Symbolic During	RL	Movement Agent
NeuroComparative [53]	Symbolic After	LLM	Language Models
Deep Symbolic GN [25]	Symbolic After	GNN	Equation Extraction
Interpret GNN[92]	Symbolic After	GNN	Outcome Breakdown

Table 3.1: Representative Work

3.2.1 Graph-Based Learning

Introduction

Graph-based learning for neurosymbolic models comes with distinct advantages. It shows relationships in a more interpretable way, while still capturing their complex connections, which generate a naturally generalized output [27]. These complex connections are then symbolically extracted to gain further understanding of the model, making it Symbolic After Learning.

A few methods exist, such as KG embedding [27], but one of the most interesting is the Graphical Neural Network (GNN) [97, 27] structure, due to its effectiveness and the inherent interpretability of graphical structures.

Relevant Taxonomy

An example from the taxonomy presented in Table 3.1 is Interpret GNN [92]. This method enhances the interpretability of GNNs by providing the activation patterns of the neural network neurons and uses external knowledge to contextualize their meanings, offering deeper insights into the model's internal workings. This approach can highlight specific edges and relevant knowledge, such as the significance of key variables across various scenarios, due to its generalized structure enabled by deep learning techniques [63]. In this case, the symbolic aspect is used solely to interpret the generated GNN, not to influence the learning process itself.

While Interpret GNN offers a a more general method to improving GNNs, another example relevant to this research is Deep Symbolic GN, also included in Table 3.1. This approach uses GNNs for the purpose of equation extraction. Interestingly, this resulted in the discovery of unknown equations, such as a novel dark matter density interpretation, which shows the impact neurosymbolic methods can have on innovation and discovery. This method, like Interpret GNN, is 'Symbolic After Learning' and works by creating a deep learning[63] GNN [27]. Deep Symbolic GN is used to capture physical relationships naturally with its structure [22] allowing for equation extraction. Symbolic regression[15] is applied over the GNN, similarly as it will be applied to RL in this investigation, to extract the equation. This allows it to be differentiable and for gradient descent [95]. This approach addresses the computational complexity - a recurring challenge in Neurosymbolic AI 1.1.

The result is a solution that has better out-of-distribution performance than contemporary graphical models such as KG embedding. The main reason for this is that the method encourages inductive biases [13], which, by avoiding static embedding and employing neighborhood aggregation [27], allowing for the grouping of knowledge, means that the model can use a set of assumptions gained from previous data to predict future data to allow for strong generalisation.

Deep learning further aids this generalisation [63], as it is able handle higher dimensional data. This allows for a higher dimensional representation of the solution to be found, resulting in latent variables[65] that give unobserved factors that influence the overall learning. Both aiding performance and generalizability as more complex models can be simulated effectively, additionally giving better results then the brute force alternative [98]. This greater flexibility in dealing with data struc-

tures allows for latent features [17, 65] to capture the hidden underlying behaviour and with its use in Neurosymbolic AI issues of overfitting can be negated [63], as capturing underlying behaviour allows for greater generalisation.
GNNs offer additional advantages, such as their structure, which promotes inductive biases [113, 13].These biases work as inherent learned assumptions that enable the model to be applied to previous unseen situations by constructing underlaying knowledge during learning. This makes the model composition an important consideration for generalisation. Moreover, for the goal of interpretability, compressed and therefore simpler outcomes can be favoured by encouraging lower complexity through modifications to the loss function.
The method does come with limitations. For example, it cannot be applied in many situations, such as the agent-based simulated environment handling. Other options exist that could have been more homogeneous in application, such as a Multilater Perceptrons (MLPs) [111] and convolutional neural networks [66] (CNNs). But these had major decreases in performance and desirable outcomes [25] due to how both models do not capture dependencies and relationational structures as well as GNNs. Additionally, the model's structure is computationally complex - a recurring issue with both deep learning and Neurosymbolic methods - meaning it is harder to create and requires more run time then comparable models [42].
However, GNNs come with certain limitations. They may not be applicable in all scenarios, such as handling agent-based simulated environments. Alternative models, like Multilayer Perceptrons (MLPs) [111] and Convolutional Neural Networks (CNNs) [66], offer more homogeneous applications but typically suffer from reduced performance and less desirable outcomes [25] since they do not capture dependencies and relational structures as effectively as GNNs. Additionally, the GNN's structure is computationally complex - a recurring issue with both deep learning and Neurosymbolic methods - making them more difficult to develop and requiring more runtime than comparable models [42].

3.2.2 Comparative Learning

Introduction

Comparative Knowledge [53, 108] is a new field of neurosymbolic learning, where the goal like GNN is knowledge extraction by providing association between different concepts. The application case is specific to language models such as ChatGPT[19] and LlaMa [112] where graphing the associated knowledge can be specifically useful in the learning process due to associated knowledge creation, additionally giving further interpretability.

Relevant Taxonomy

The NeuroComparative method [53] is specifically designed to create high-quality, well-generalized comparative knowledge.

The method works like most comparative learning models by first seeding entity pairs for comparison and then generating new comparatives using existing comparisons as a reference. It focuses on over-generating an abundance of outputs and then filtering these outputs to create a more diverse initial set. This culminates in the cleaning of the created comparatives by the removal of contradictory or compromised generations using symbolic logic.

The result is a stronger method than contemporary Neurosymbolic comparative methods[108], with an 32 percent gain in created knowledge and a 30 percent more diverse outcome compared to previous best methods. The more generalized, highly interpretable comparative outcomes and the higher performing results created highlight the reasoning behind the chosen method and the paper that uses it [53].

That focus nevertheless also comes with some limitations. Like the Deep Symbolic GN method, Neurocomparatives have a limited application space. They are only really applicable for language models, and specifically aiding the comparative aspect of such models, showing the relevance of tailoring the AI for the situation. Additionally, the method of over-generation can lead to increased run times compared to previous methods [108].

3.2.3 Neurosymbolic Concept Learners

Introduction

Other methods such as Neurosymbolic Concept Learners, focus on learning aspects from data, rather than just the interpretation of the learning like the previous methods. Observing aspects such as the colour, shape, or patterns, in more visually-based data and are Symbolic Before or During Learning [51, 74, 67]. All of these methods work by decomposition, breaking down the problem into simpler to handle sections. This is done symbolically to before learning is complete. The examples present in the taxonomy are NSS [51] with audio and NS-CL [74], Dood [67] learning a visual representation.

Concept Learners Audio

In the case of NSS [51], decomposition works by converting audio data into a visual string format, allowing the system to make symbolic assertions about the data. This process requires filtering and regularization of the audio, similar to the filtering in Comparative Learning and regularization techniques used in Deep Symbolic GN.
Both are done to aid learning, with the regularization in particular promoting greater generalisation [35]. This works by the simplification of the model and the reduction in noise that aids in overarching robustness. The method then segments created lines from the audio, with the use of unsupervised learning for clustering [29] the segments, allowing for latent representations to be found and aiding generalisation. A key takeaway from this method, however, is the use of sleep-wake methods [89]. These offer many benefits that this review is seeking, such as the effective learning of latent variables and improving generalisation, aided by the generation of new samples to learn outside of the training set during the sleep phase.

Concept Learners Visual

Stroke-based drawing methods, such as DooD [67], provide another example of neurosymbolic concept learning. Similar to NSS [51], DooD functions well with out of distribution data, as it learns to find general purpose representations in an unsupervised method.
An interesting difference and key take away is that Dood introduces the concept of amortised inference structure [126], effectively a function that rapidly maps novel data to parameters due to training off previous data, allowing for greater efficiency in problem handling. Which aids in directly reducing the computational complexity of neurosymbolic method 1.1 and outside of this investigation could be interesting to re-examine in the context of temporal logic[93] and agent based environmental exploration.
Another concept learner method that is appearing increasingly more in literature is Neurosymbolic Concept Learner (NS-CL) due to its effectiveness [74]. NS-CL works by learning representations of individual objects in a short question format around object-based concepts such as colour and shape. After extracting the scene and its latent representations, including orientation and the relations the objects have to each other a structure can be created. The benefit of this structure is that it describes the scene compositionally, like Deep Symbolic GN, aiding the generalisation of the method. It can take what it previously learnt and apply it in new

situations, iteratively making its knowledge more complete and complex. Because of this methodology, the model needs vastly less information than comparative methods [56, 54]. For example, when only 10 percent of the original information was given, NS-CL only had a 0.3 percent drop, while other models fell by more than 30 percent, which speaks to the strong generalisation of the model.
Additionally, NS-CL is unsupervised and does not rely on prior annotation [123]. Combined with deep learning, this allows NS-CL to discover latent variables and take advantage of their benefits.This results in attaching language to a non-language based problem, through concept embedding [104], which can can improve predictive accuracy and understanding for the user.
However, the method has its limitations. In more complex environments, NS-CL struggles to make logical assertions, with previous methods winning out [74]. But with all models currently tested performing suboptimally in these images, it shows the clear need of improvement and that these methods are still first steps.

3.2.4 Zero-Shot Learning

Background

One way to overcome the underperformance of models in complex environments is through Zero-shot learning methods [115], which can recognize novel aspects at inference time by leveraging high level abstract features that the model is given prior to learning. Working outside of training and prediction allows for a greater generalisation of the model, as it can adapt outside of its training set by using the general knowledge of the subject matter. This can be especially useful when the environment is constantly changing, such as automated driving systems.

Relevant Taxonomy

Various methods have attempted to apply zero-shot to Neurosymbolic architectures. A prominent example is ZeroC [118], which has interesting connections to other methods discussed. Like NS-CL, ZeroC works on image classification and the relationships between the various components in the image, with the methods paper making a connection to NS-CL for the similarity in compositionality. The structure of its relationships are also interestingly similar to Deep Symbolic GN, as it represents its learning in a graphical format, or more specifically a symbolic graph structure [88]. This demonstrates how seemingly unrelated methods can share underlying principles that may be applied across different domains.

Outside of these connections there are other vital takeaways, such as how ZeroC is energy-based [64], which helps compositional generation. This allows for the dynamic evaluation and refinement of generated compositions based on their energy scores, or in other words deciding connections based on their desirability. Similar techniques are reflected in other works [31], where regularization is used to maintain energy levels within a certain range, thereby influencing inductive biases towards simpler, more generalisable configurations.
Mirror is then used to allow for acquisition at inference time [118], with both the energy-based neural and symbolic graphical components of the model mimic each other to gain relational knowledge acquired from the onset. This mirroring shows the incorporation of symbolic reasoning during the learning process, enhancing the model's ability to integrate and apply new information effectively. The benefits of this structure can be found in how it can be applied across domains, after distribution shifts and begin learning at inference time, which out-performed the current state of the art [99], pointing to the effectiveness and successful generalisation of the method.
The ZeroC method, while innovative, faces limitations. Its testing in a single environment raises questions about generalizability, necessitating more comprehensive validation. Additionally, its focus on vision-based tasks restricts its applicability to agent-based environment exploration. Nevertheless, as a seminal method, in terms of how it compares to previous applications [115], ZeroC offers valuable insights for other zero-shot learning approaches. DARLA (Disentangled Representation Learning Agent) [48], built on Markov Decision Processes (MDPs) [84], exemplifies this potential. As a reinforcement learning neurosymbolic zero-shot method for agent environments, DARLA addresses the out-of-domain distribution problem in reinforcement learning.
Although DARLA may not match ZeroC's performance relative to baselines in their respective fields, principles from ZeroC could potentially enhance DARLA's capabilities. Additionally, being a reinforcement learning model, it has a more direct impact on the current focus of the thesis. This illustrates the potential for the cross-pollination of ideas in zero-shot learning methodologies across different machine learning subfields.

3.2.5 Reinforcement Learning

Background

More into Neurosymbolic reinforcement learning methods, this will focus on policy creation due to being most applicable to the investigations problem as they are

generally used for agent-based tasks, such as the core of this review and therefore the longest section. All reinforcement learning programs follow the same principle of learning and refinement, with the learning informing the refinement and the refinement informing the learning, leading to the programs being symbolic before or during learning.

General Learning

While neurosymbolic reinforcement learning methods vary in their specific approaches, some frameworks aim to provide a general structure for integrating symbolic reasoning with neural learning in RL contexts. Works like [43] overview the various approaches that neurosymbolic learning can take, covering models from Logical Neural Networks(LNN) [94], Differentiable Decision Trees (DDTs) [107] and integration into existing RL Frameworks, like this investigation is covering.

LNNs [94] present neural architectures that integrate logic statements, containing the operators AND, OR and NOT that allow for effective learning. A benefit of this strategy is that it is very clear to the see the logic behind the model's decision making, increasing interoperability, because it can be represented in predicate equation. LNNs in turn require data that is symbolically-based, making them symbolic in data and in turn restricting their application cases to scenarios where data is symbolically available. Additionally, generalisation is improved having the process include logical rules and general principals that can be learnt rather then just having the data memorized. Using fuzzy logic [60], moreover, allows for looser logical structures to be captured as they need do not have binary outcomes, giving a more nuanced solution. In practice, this approach has been applied to simple control tasks around the regulation of the buildings climate control, with the agent being the building itself. This enhanced interpretability, generalisation and improved decision-making allows for lowered costs from reduced energy waste.

Differentiable Decision Trees (DDTs) offer a similar improvement in interpretability and generalization. DDTs extend traditional decision trees by assigning thresholds at each node, with the leaf nodes representing the probabilities of different RL outcomes, thereby guiding the agentâs decisions. The core benefit of DDTs lies in its simplicity; their decision-making processes are straightforward and easily traceable, enhancing transparency. For generalisation, its simplicity forces the model to capture overall behaviours rather than memorizing data. Unlike LNNs, which are 'Symbolic in Data' and come with their own set of benefits and drawbacks 3.1.4, DDTs are âSymbolic in Learning,â offering a different balance between complexity and interpretability.

These examples show how a real world application, climate control, makes the case for improving the transparency of logical decision-making as a cost saving measure but also, more importantly, for the investigation to improve both the generalisation and transparency of learning

Compositional Structure

There are various approaches to applying neurosymbolic systems to reinforcement learning (RL), with integration being a key component as highlighted in previous studies [43]. Some methods, such as DUA, seek to combine multiple approaches into a unified model to maximize their benefits. Like NS-CL DUA is vision-based but, unlike NS-CL, the method deliberately structured in a compositional form, with three distinct modules: Detect, Understand, and Act, each functioning independently and feeding into the next. This comes with several advantages that are generally related to scalability and efficiency [47]. This modular structure offers several advantages, particularly in terms of scalability and efficiency [47]. Additionally, it allows for flexibility in replacing individual modules, such as changing the âDetectâ module or modifying the âActâ functions without altering the core learning component or its use of deep learning, which is explored further later in this review. A key take way from this method however is its use of meta policies [49, 78]. Created using inductive programming, they can guide policy application and, in turn, aid program effectiveness and robust generalisation for out of training distribution.
Conversely, there are limitations to the method's structure and outcomes. It is difficult to get the various components of the method to work seamlessly, making it computationally complex in implementation. Despite this, the main takeaways are that meta policies are an important tool in reinforcement learning. Also, for optimising learning, compositional programs have their clear advantages and deep learning continues to be a necessity for stronger generalisation.

Policy Verification

Other methods like PIRL [114] that deal more directly with agent based exploration also have key take ways. One is creating a domain-specific language for learning. By working with latent variables a higher-level interpretation can be created that both allows for a more effective outcome, strong generalisation and a highly interpretable for the user. Furthermore, the method also highlights that using a policy sketch [7], which is an abstracted representation of a policy, can aid regularization for policy creation. These components help making the method very

generalizable, effective and interpretable policies that can be transferred across environments. Although, the method does struggle at higher dimensional environments it is still a seminal method as it out preforms the baseline methods, with it further allowing for policy verification [119] for the issue of safety, creating less fail cases then the contemporary methods.
This idea of verification can also be found in other methods, such as REVEL [6]. While other methods do exist and follow a similar structure [4, 37, 38] of verifying the policy and trying to prevent improper policy use through the use of shielding [5], REVEL represents the current state-of-the-art and outperforms previous methods, including within the Gymnasium environment [28]. This was largely done by REVEL using continuous shield updating rather then static shielding, which helped lead to further optimisations and better safety handling; however, this raises computational costs, a common problem for policy verification [40, 2]. REVEL mitigates these costs by having two policy classes, a general one with bounds and a second one that gets verified. By giving bounds, the certified policy only has to verify its safety within said bounds rather then across all available possibilities. This drastically reduces computational cost, but importantly this is still more computationally taxing than not being concerned with verification. Other ways the method lowers computation cost are by working with gradient descent and having the policies be neurosymbolically based. This allows for quicker verification of symbolic logic rather than the more resource draining prospect of neural logic verification, due to its lack of interpretability to both the user and program.
Interestingly, properly applying safety can have other benefits. As discussed in PIRL, a safer program tends to have fewer failure states, effectively limiting exploration to more positive outcomes. This is partly why REVEL generally out performs benchmarks and why, when it loses to them, the margin is only slight, showing that work still needs to be done but that shielding makes the program generalize better. The key takeaway is that shielding offers advantages beyond safety, such as improved optimization and generalization, and that dynamically updating the shielding further benefits the program.

3.2.6 Review Outcomes

The reviewed works in neurosymbolic AI showcase diverse approaches to advancing interpretability and generalizability. These range from graph-based methods for equation extraction to concept learners for representation learning, as well as reinforcement learning techniques integrating symbolic reasoning for policy creation and verification.

Significant progress is evident in the field. The ability of NS-CLs to maintain performance with minimal data and the enhanced knowledge creation of NeuroComparatives demonstrate improved generalizability. In reinforcement learning, REVEL's dynamic shielding approach has enhanced both safety and optimization. ZeroC's performance in zero-shot learning across domains further underscores these advancements.

Despite progress, challenges persist. Computational complexity remains a concern, particularly in deep learning and policy verification methods. The seamless integration of symbolic and neural components and balancing interpretability with performance continue to be key issues, with generalisability and a lack of interpretability being areas in constant need of improvement, due to their importance in both safety and functionality.

Future directions in neurosymbolic AI aim to address these challenges. Research is likely to focus on enhancing symbolic-neural integration, improving computational efficiency, and developing more general-purpose architectures. Advances in safety mechanisms, scalability to complex environments, and explainable AI are also anticipated, with the key insight into reinforcement learning environments that can be used to test these features, such as Gymnasium. The field is moving towards transfer learning capabilities to further enhance generalizability and interpretability across tasks and domains, which leads to the methodology of the thesis, with its attempt at solving the issue.

Chapter 4

Neurosymbolic Algorithmic Discovery

4.1 Approach Overview

This research investigates the potential of integrating SR into reinforcement learning, specifically the Proximal Policy Optimization (PPO) algorithm, to enhance interpretability and generalizability. SR [70] was chosen for its ability to discover underlying mathematical relationships in data, which aligns well with the investigation's goal of extracting interpretable policies from reinforcement learning models.

The integration of SR occurs prior to the evaluation step of PPO. This approach allows us to influence the learning process itself, rather than merely interpreting the final learned policy. The aim is to guide the reinforcement learning algorithm towards more generalizable and interpretable solutions throughout the training process. The methodology builds upon the existing codebase for PPO [102], leveraging its proven effectiveness in reinforcement learning tasks. The SR component is implemented using the PySR library, which provides a flexible and efficient framework for SR.

To maintain the strengths of PPO while still integrating the SR into the evaluation step, a specific modification path is devised. Where the policy from PPO is extracted, SR is applied over the policy. The extracted equation is then used to make predictions for the policy values, with these predictions used to inform the learning, making the neurosymbolic learning system. This allows for PPO and aforementioned benefits of PPO, in terms of safe effective learning, to remain, while gaining the benefits of neurosymbolic AI in terms of generalisation and interpretability.

To further demonstrate this a high level pseudo-code 4.1 was created in order to showcase what changes would be required to take place and where.

1. Initialize the PPO algorithm with standard parameters.
2. At regular intervals during training (e.g., every N episodes):
   a. Extract the current policy representation from the PPO model.
   b. Apply symbolic regression to the extracted policy, attempting to find a compact mathematical expression that approximates the policy behavior.
   c. Use the mathematics expression to predict and replace policy related values.
   d. Implement the new policy values to influence PPO's learning process
3. Continue PPO training, periodically guided by insights from the symbolic regression step.
4. Evaluate the final learned policy for performance, interpretability, and generalizability.

(4.1)

The process in the pseudo-code 4.1 works by interleaving SR with the reinforcement learning process, making it Symbolic During Learning. The learning is steered towards solutions that are both generalised and interpretable, while allowing for the freedom of learning provided by neural networks and a moderate computational complexity due to not constantly going to SR.
The following sections will detail the specific implementation of this approach, including the PPO algorithm configuration, SR parameters, and the mechanism for integrating symbolic insights back into the reinforcement learning process.

4.2 Algorithms

The integration framework developed 4.1 led an exploratory period, in which various permutations were considered and explored in terms of effectiveness and feasibility This eventually resulted in the creation of two novel algorithms, each leveraging SR to enhance the interpretability and generalizability of the PPO method. The reasoning behind creating two different algorithms is multifaceted. As both are novel applications, there was no guarantee that they would work, allowing the investigation to continue if there one fails. Additionally, as will be discussed further, the algorithms have two different methods in which the high-level modification 4.1 is applied, one where the symbolic interference weight on the learning program can be adjusted in the loss function, the other where the policy is updated with SR through a less managed approach. The level of symbolic interference is

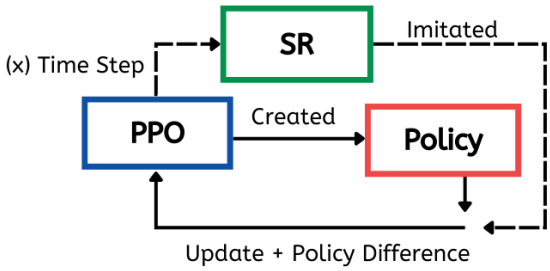

Figure 4.1: SLMA Work Flow

the main difference between the two approaches. Whether a higher or lower level of interference will lead to an optimum outcome for generalisability and interpretability will be explored in this investigation in order to gain the best algorithm possible.
These algorithms can be used independently or in combination, offering flexibility in how symbolic insights are incorporated into the reinforcement learning process.

4.2.1 Symbolic Loss Modification Algorithm (SLMA)

SLMA works by taking the two policies values, one created by the nominal process of PPO θ_k, and the other by applying SR to gain novel policy values θ_s. By then applying the Mean Square Error (MSE) over the two values data points n, the difference can be calculated, creating the work flow of SLMA 4.1.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\theta_k - \theta_s)^2 \tag{4.2}$$

The difference is calculated using MSE 4.2, where gradients are applied, to inform the backpropagation process, updating the neural network weights. This update is modulated by the value coefficient, which determines the proportion of value loss in the overall loss function. This balance helps maintain proper scaling throughout the learning process, preventing oversized gradient updates from the MSE loss that could disrupt learning as the model approaches optimal solutions. Together with the other elements of the PPO loss function 2.1, this informs the model for what to change for the next iterative run.
This process is represented in the algorithm SLMA 2. By building on the previously discussed PPO algorithm, two new hyperparameters are introduced K and N. K. This works by weighting the MSE value, which adjusts its impact on the overall loss and consequently on the gradients.

Algorithm 2 Proximal Policy Optimization - SLMA

Require: Initialize policy parameters θ_0, value function parameters ϕ_0

1: **for** iteration = 0, 1, 2, ... **do**

2: Collect trajectories $\mathcal{D} = \{\tau_i\}$ by running policy $\pi_{\theta_{old}}$ in the environment

3: Compute rewards-to-go $\hat{R}_t$ and advantage estimates $\hat{A}_t$ using the current value function V_ϕ

4: Update the value function by regression on mean-squared error (MSE):

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

5: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right)$$

6: **if** $iteration \mod N == 0$ **then**

7: **apply symbolic regression over found policies:**

8: ▷ Use Created Mathematical Expression to Predict New Policy Values

9: ▷ Apply MSE Between the PPO Policy Values θ_{k+1} and the Symbolic Regression Policy Values θ_{s+1}

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\theta_{k+1} - \theta_{s+1})^2$$

10: ▷ Add the MSE value to the Loss, weighted by ratio K

11: **end if**

12: Apply back propogation on the model using the Loss

13: Update old policy parameters: $\theta_{old} \leftarrow \theta$

14: **end for**

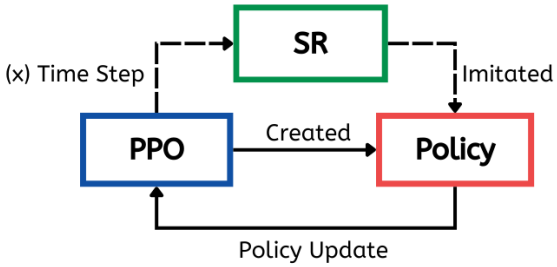

Figure 4.2: SLMA Work Flow

Hyper-parameter N is chosen to represent how often the model should go into symbolic state. This must be balanced. Too high N value has increased computational complexity and decreased freedom of learning, as the SR would effectively anchor the policy creation, with permutations in the policy being constantly resolved to a baseline. If the N is too low, however, the benefits of neurosymbolic AI 1.1 would be reduced, limiting the gains to generalisation and interpretability achieved through the model.

4.2.2 Symbolic Policy Update Algorithm (SPUA)

This same consideration takes place for SPUA, however, due to the algorithm having a different implementation, the values for optimum N would vary. The difference lies in how SPUA takes the SR policy value and, instead of finding the MSE value difference between it and the PPO policy value, the SR policy value is passed every N runs. This creates the SPUA workflow 4.2.
Again, this results in an algorithm 3. The functional difference between these two algorithms is that SPUA causes a more systemic difference in how the outcome is affected. This is significant because it is not just a factor affecting the loss. By passing back an entirely new policy, the algorithm influences the overall loss through its impact on policy evaluation and the subsequent value estimation from the critic. Importantly, SPUA is given no way to weight this as a contrast to SLMA with K.
This would likely mean that SPUA would need a lower N value present than SLMA, due to its comparatively greater impact on learning. This softer and harder approach between the algorithms does set up the investigation to provide a more definitive look at how much a neurosymbolic system should apply its symbolic teachings to the learning process, what is the trade off and which is more beneficial.

Algorithm 3 Proximal Policy Optimization - SPUA

Require: Initialize policy parameters θ_0, value function parameters ϕ_0

for iteration = 0, 1, 2, ... **do**

Collect trajectories $\mathcal{D} = \{\tau_i\}$ by running policy $\pi_{\theta_{old}}$ in the environment

Compute rewards-to-go $\hat{R}_t$ and advantage estimates $\hat{A}_t$ using the current value function V_ϕ

Update the value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right)$$

if $iteration \mod N == 0$ **then**

apply symbolic regression over found policies:

▷ Use Created Mathematical Expression to Predict New Policy Values

▷ Replace the PPO Polcies θ_{k+1} with the New Policy Values θ_{s+1}

▷ $\theta_{k+1} \leftarrow \theta_{s+1}$

end if

Apply back propogation on the model using the Loss

Update old policy parameters: $\theta_{old} \leftarrow \theta$

end for

4.2.3 Combination - Symbolic Loss and Policy Update (SLPU)

Due to how the algorithms are organised, SLAM and SPUA can be used in many different reinforcement learning algorithms and applied to existing real-world situations, but this flexibility also means they can be used conjunctively with each other.
SLPU 4 can allow for the benefit of both algorithms to be use simultaneously while effectively having the same computational complexity as it uses the same symbolic regressive step. SLPU is functionally a third algorithm due to how it changes the outcome and the work required in allowing for the use of both systems concurrently.

4.3 Discussion

SR has been implemented in various methodologies successfully in the field of equation discovery, as discussed in the Literature Review [97, 15]. However, no methodologies have applied SR in reinforcement learning, even though the solutions to reinforcement learning problems can just as easily be represented as an equation with the variables representing the features of the environment.
Furthermore, the process of using SR allows for the more generalisable behaviour to be captured. As previously stated, SR cannot capture the behaviour as fully as a neural network can. This allows for greater generalisability than would otherwise be possible with the retention of fine tuning.
In addition, as a modification of the overall reinforcement learning process, the use of SR allows the system to gain a core value: ease of integration into existing AI methodologies. This allows this work to be applied to other reinforcement techniques, such as TRPO [103], DQL [34] and DDPG [90], with ease. Moreover, this allows the system to be applied in real world situations to increase generalisation with the methods provided in this project.

Algorithm 4 Proximal Policy Optimization - SLPU

Require: Initialize policy parameters θ_0, value function parameters ϕ_0

for iteration = 0, 1, 2, ... **do**

Collect trajectories $\mathcal{D} = \{\tau_i\}$ by running policy $\pi_{\theta_{old}}$ in the environment

Compute rewards-to-go $\hat{R}_t$ and advantage estimates $\hat{A}_t$ using the current value function V_ϕ

Update the value function by regression on mean-squared error (MSE):

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right)$$

if $iteration \mod N == 0$ **then**

apply symbolic regression over found policies:

▷ Use Created Mathematical Expression to Predict New Policy Values

▷ Apply MSE Between the PPO Policy Values θ_{k+1} and the Symbolic Regression Policy Values θ_{s+1}

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\theta_{k+1} - \theta_{s+1})^2$$

▷ Add the MSE value to the Loss, weighted by ratio K

▷ Replace the PPO Polcies θ_{k+1} with the New Policy Values θ_{s+1}

▷ $\theta_{k+1} \leftarrow \theta_{s+1}$

end if

Apply back propogation on the model using the Loss

Update old policy parameters: $\theta_{old} \leftarrow \theta$

end for

Chapter 5

Implementing Neurosymbolic AI

5.1 Symbolic Regression Investigation

Various SR methods were investigated prior to PYSR being chosen for the methodology. This investigation consisted of reviewing literature, reviewing the various SR methodologies [70, 110], and some light testing of these methods within an reinforcement learning environment. The reasoning behind the need for testing the SR methods, unlike PPO, is their novel use in reinforcement learning, as while they may be extremely successful on the grounds of equation discovery reinforcement learning could have different considerations.

One of these considerations was performance, both in terms of resulting outcome and computation efficiency, as the reinforcement learning model would have to enter the symbolic regressive state many times throughout the algorithms run unlike how it is nominally applied after the learning process [25, 15]. Another crucial aspect is the previous mentioned customizability as SR had not been previously applied to reinforcement learning models, both in terms of parameters and data handling as to accommodate the diverse outcomes from different environments.

Key methods were identified from literature, this consisted of reviewing survey papers and ranking tables, potential SR Algorithms were identified, Deep Symbolic Regression (DSR) [62], Bayesian Symbolic Regression (BSR) [55], GPlearn [10] and finally PySR [24].

Each of these methods are powerful in their own right, with DSR's ability to handle complex relationship representations, important with the increasing complexity of the policies needed in the listed environments. The difference that BSR allows uncertainty to be integrated into the model, which is interesting for exploratory reinforcement learning. GPLearn is simply an inbuilt library, which allows for a

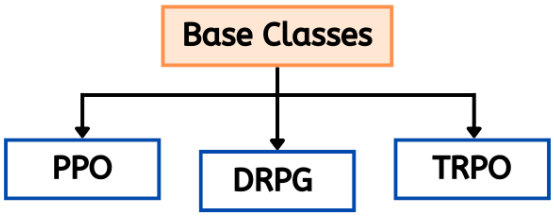

Figure 5.1: Base Class Representation

greater ease of integration into the model.
All of these models were then trailed within the various Gymnasium environments, using the previously designed algorithms of SLMA 2 and SPUA 3, the metrics of loss and computational complexity were used to benchmark.
However, the models fail to parallelize the run of their code and do not make a clear effort at reducing the computational overhead, unlike the previously explored PySR. This resulted in a computational explosion in the more complex models of BSR and DSR, limiting the results gathering and disqualifying their use in a learning cycle. While GPLearn was found to less computational demanding, the lack of customisability, flexibility and still having a higher computational cost resulted in it not being selected. This leaves PySR as the clear and only choice as a SR Model.

5.2 Stable Baseline Approach

To understand the system, it was first decomposed of what modules are required in what sections in order to locate the area of necessary modification. This was done by looking at individual imports and dependence's in areas of key interest, such as the sections related to the evaluation and updating of the policy, due to this being the areas where the algorithms SLMA 2, SPUA 3 and SPLU 4 vary from the PPO base case 5. From this understanding a select files were isolated to build the frameworks format off of.
Due to this greater understanding the format was created to allow for the ease of addition and configuration, models and hyper-parameters. This was achieved through the careful modification of the stable baseline framework, through modifying base-classes in python rather then the higher level aspects of the framework. Leveraging this approach, other models using the same base classes required minimal modifications to achieve functionality.
The reasoning behind this can be seen in Figure 5.1, as like PPO the prior mentioned TRPO [103] or the also capable DDPG [90] can use the same integrated

changes. While this required more work in the initial stages of the frameworks modification it allowed for a more fluid integration. The integration results in cleaner, more readable code through thought-out compartmentalisation and allows for this work to be built on in the future.

5.3 Algorithmic Implementation

Aided by cleaner code, multiple checks were required for the various algorithms to exist within the same system without causing conflicts between them. This framework used variable checks and counters to adapt to different run types and data availability, enabling features like MSE exclusion in SPUA runs and switching between normal and modified loss functions, all within a single, flexible structure that accommodates various scenarios seamlessly. While complex to implement, as there are many variables and factors that do not exist at varying stages of the learning, it allowed for a unified easier to use learning system with everything to using base classes.

By using the flexible learning system PySR customisability was utlized to produce a more useful symbolic outcome by balancing the number of operators that PySR was allowed to use. Fewer operators reduces the time complexity, as more operators require exponentially more iterations for solution discovery due to the greater number of combinational possibilities, but limit the the freedom of representation that more operators provide. To limited time complexity issues, that were presented with other SR systems [62, 55], and encourage greater interpretability fewer, but still key, operators were chosen for SR. Another modifications to the SR system involved giving it a custom loss function, effectively a variation of MSE 4.2, in order give a simple, to reduce training time, but still effective loss to gain the best results possible.

There is however a break in the learning system, where the SR aspect is introduced, creating a few challenges. The primary challenge was that the gradients needs to be removed from the policy data in order to apply the SR aspect to them, as the data is moving from a neural solution to a data point array for the SR creating symbolic representation that needed to be moved back into a neural solution, this presented issues in a break in the gradients in the learning system. To reapply gradients effectively, the system needed to modify both the type and organization of data, in multiple stages to get it back to the correct format for the neural solution from a symbolic one, leading to a more complex data handling that had to be adaptive to the changes in the environmental outcomes as the solutions would vary. Due

to the complexity and repeated use of SR, there are instances where the outcome would be faulty, having gaps in the data and occurring roughly in every hundredth instance, is then detected by implemented checks and the program is asked to rerun.
To see if this system worked the Weight and Gradient Update 6.5 and Drift 6.4 metrics were used to see if the system was updating. What was observed from the weights and gradients of the neural network is that the algorithms were was having an effect on and were modifying the learning process. Observed again with the Drift Value, as the policies were found to have shifted after its application.
Additionally, by removing and reapplying gradients the metric data that would normally come with the gradients, as the gradients also hold history of the updates, are no longer relevant as they would only represent the metric information since the last removal. This resulted in the need of a system for gathering the results to be designed in order to properly evaluate the outcomes of the algorithms.

5.4 Results Gathering

One issue that the framework presented was the gathering of necessary information, in terms of the metrics previously presented 9.5. As stable baselines does readily provide metrics for the outcome of the models, but for the purposes of the investigation the learning process metrics are additionally required. This becomes an issue when the speed of learning and the point of convergence, where a solution is found, are to be discussed. To solve this a monitor was created to catalog and output the metrics during the systems run and storing them in a csv. However, due to the shifting nature of the data, in which the organisation of the metrics would change, precautions had to be taken in order to take the right data and drop the unnecessary. Accomplished by using a search algorithm in the data output in order to make sure the right data was extracted.
To gain the data and due to the scale of hyper-parameter testing, as there are effectively four algorithms to explore, SLMA, SPUA, SPLU and the base case PPO, a system was created for the automated testing of hyper-parameter discovery to improve the user experience. The need comes from how individual tests could require ten hours to run, by allowing tests to run concurrently the speed of research can be drastically improved. The system involved the tester filling in the hyper-parameters that they wanted tested in a table, which would then be used to update the configurations of the models run by directly editing the prior made configuration file. By this type of implementation the user is given a central point to modify

a regular run, with the configuration file, and to add multiple hyper-parameters to test in a user friendly format, drastically improving ease and speed of testing, as otherwise everything would be an individual run. To make this even easier on the user, the files in which the data is stored are automatically created, averaged, and if the same increment already exists are cleaned in order to give the most seamless experience possible, where only the variables in the configuration are to cared about.
For the data to be robust, mentioned in the methodology, an average was taken. This was done by first dividing by the amount of data points that are to be put in and then continuously adding the data to a csv file. This system again is customised for the user, with the number of runs with original seed values to be configured, moreover giving the ability to reproduce results. Additionally configurable is how the model can run different types of tests, single, multiple, custom environments and if the data gets cleared after each run all to the users choice.

5.4.1 Result Robustness

To ensure the results robustness multiple precautions were taken. One of these precautions was to run the learning multiple times, five times each, in order to gain a stable average value, giving greater validity to the results. To guarantee that the seed value of the environment, essentially the slight randomisation in position that each environment has to its start, does not vary in to give some tests an advantage over others all tests were given a pre-chosen set of seed values.
When obtaining values from the custom environments a similar precaution was taken, where the seed values were again a fix set and the environment was ran over ten times, for each test. This is due to the greater instability in the modified learning environments, due to their modification, requiring more test cases to gain a stable average and meaning that the averages obtained from the custom tests are a fifty run average, with ten runs per test and five tests.

5.5 Hyper-parameter Discovery

The hyper-parameters were discovered by doing rapid testing on simpler environments to find a basic set of hyper-parameters that would work well. These however were not chosen outright due to them taking place on the simplest environment, but were used as a guiding set.
From this guiding set the hyper-parameters were given different permutations individually in order to explore which ones were the most effective, in terms of the

performance and generalisability as an easier measure then interpretability, which is mostly relates to the number of neurosteps N, as this would have the greatest effect on how similar the solution is to the outcome.
This saw the modification of key hyper-parameters, with the ratio to the policy K, neurostep N, iterations to the symbolic program, complexity C, gamma γ [63] and learning rate α [63].
The ratio to policy was chosen as it has a large effect on the effect on the size of the neurosymbolic component. As previously described comes with benefits and draw backs, such as interpretability against freedom of learning. The same is true with the neurostep, representing how many neural steps are taken against symbolic, as a lower number would mean that the program goes more often into a symbolic space, additionally increasing time complexity. Additionally increasing time complexity are the number of iterations given for SR to find a symbolic representation, with the benefit of possibly finding a more optimum solution.
The complexity hyper-parameter, which represents the number of constructive arguments allowed in variables and operators, has an additional aspect to consider. If it is set too low, the relationship would not be captured properly for learning to be effective. However, a higher value leads to less interpretable results and more computationally intensive processes. From these variables a pattern emerges that there is a theme of balance in terms of the values chosen to capture the right aspects of neurosymbolic AI effectively.
The other hyper-parameters relate to the PPO component of the model, as the neurosymbolic components may influence what are the most effective variables. Starting with gamma, a new concept, as a discount factor that determines the importance of future rewards, with values between 0 - 1 but staying in largely higher then 0.9 [63] as otherwise learning can be impacted due to not prioritising future rewards and only short term gains. This instability also applies to the learning rate, which determines how much current knowledge is updated due to new information. This metric was primarily used to gauge the potential prior-described anchor policy of the SR method, as to bring greater stability to the learning program due to its more restrictive nature of learning.
Iterations of all these variables were explored with each algorithm to maximize advantages. When choosing between performance, based on the previously described metrics, and generalization, generalization was prioritized in situations where the two diverged in terms of which variable was most beneficial.

Chapter 6

Experimental Setup

Experimental setup covers the metrics used in order to validate the created algorithms, explaining the choice behind them and the testing in reinforcement learning environments, with their modifications, from which the metrics were gained and additionally listed in the appendix 9.3.4.

6.1 Metric Selection and Creation

Many metrics exist for measuring the performance and outputs of an artificial intelligence program [16, 83]. This section will look the focus of this review - how to measure and quantify generalisation and interpretability in order to gage the effectiveness of the modifications made to the PPO program, through both the examination of related literature [116, 57, 6] and creation of the relevant metrics.

6.1.1 Common Metrics

A key metric would be Loss, as primary indicator of the overall effectiveness of the model [116]. Loss is calculated, using the objective function equation **??** given previously and the environments reward function. Together, they provide a outcome of how much the model deviates from the idealised solution.
Due to the fact that some of the environments do not have a consistent reward structure (such as MountainCar, where loss can be minimised through an absence of action [**Developers˙Gymnasium**]), Mean Reward 6.1 was chosen as another primary indicator. Mean Reward provides additional insight into the average effectiveness of the model in the environment and a direct understanding of its perfor-

mance against the reward function of the given environment [116].

$$\mu_R = \mathbb{E}[R] \tag{6.1}$$

Loss 2.1 and mean reward 6.1 are both important. Loss gives insight into learning performance, reflecting stability and algorithmic efficiency, while a high mean reward indicates the model's effectiveness in maximizing the reward function. Mean reward provides insights into the method's performance from the expected E reward R, offers different perspectives . Using these metrics in conjunction allows for a more nuanced evaluation of the model's overall performance.

6.1.2 Metrics for Generalisation

Generalisability in reinforcement learning refers to an agent's ability to perform well in environments or situations that differ from those encountered during training. This section discusses key metrics used to quantify generalizability, with a focus on standard reward and mean reward in environments with intentional domain shifts, by modifying the environment on which the model is originally trained.
Standard Deviation of Reward σ_R measures the agentâs consistency across different environments.

$$\sigma_R = \sqrt{\mathbb{E}[(R - \mathbb{E}[R])^2]} \tag{6.2}$$

σ_R measures the reward R against expected $\mathbb{E}$ reward $\mathbb{E}[R]$.
A lower standard deviation indicates more consistent performance, suggesting better generalizability within the training environment. This metric is crucial as it captures the stability of an agent's performance, which is essential for reliable and predictable behaviour in real-world applications and relates to the investigationâs overarching idea of safety.
Mean Reward here is the same metric presented previously 6.1; however, in this case it is specifically applied in the modified environment, where the generalisation of the method is tested to see if SLMA and SPUA allow for a higher reward value.

6.1.3 Metrics for interpretability

The loss in this case is the difference between the solutions S predictions, what action would be taken under what input, when influenced by SR S_r and when only with PPO S_p.

$$\text{Loss} = \frac{1}{n}\sum_{i=1}^{n}(S_{pi} - S_{ri})^2 + \lambda \cdot C(\theta_s) \tag{6.3}$$

By using a modified version of MSE, with the previously described complexity C 2.6 and benefits of regularisation λ, a usable and stable symbolic loss is created. This allows comparison to see if the program, by being influenced by SR, confirms that the smaller the loss in the extracted equation, the closer the estimations are to the true relationship.
Another difference is measured between the equation predictions θ_k and the true policies θ_k with Drift Value D_f.

$$D_f = \frac{1}{n}\sum_{i=1}^{n}|\theta_{k_i} - \theta_{s_i}| \tag{6.4}$$

Drift Value 6.4 is an important metric that was created for two distinct reasons â first, to validate that the policy is being updated in the modified learning program; and second, to understand the implications of those drift values on capturing behaviour. For further insight, high, medium and low policy values were separated to see the effect of capturing behaviour in regions of higher volatility, each representing a third of the possible policy value range 0-1.

6.1.4 Metrics for Validation

This leads directly into the metrics for validation. As this is a new theoretical algorithm and involves direct modification of the learning program, the method could simply not work and use stored values, such as the previous gradients, during the neurosymbolic steps in order to continue learning. By gaining the drift values 6.4, the policy update can be observed.
The other metric created is the weight and gradient value difference between updates after the neuro-step. By examining the sub-processes of the learning program, we can extract the weights w and gradients of the model updates.

$$\Delta \mathbf{w}_i = w_{2,i} - w_{1,i}, \quad \text{for } i = 1, 2, \ldots, n \tag{6.5}$$

The equation 6.5 allows observation of the learning process in real-time, providing validation that the necessary changes are taking place and that the model is performing as expected.
All these metrics come together, with the time difference it takes to run the program, to form a comprehensive view over the entirety of the learning process 9.5.

6.2 Environments

Now that the metrics have been established, it is important to look at how they will be gained. The reasoning behind how to measure generalisation lies in the concept of a distribution shift, as described previously. The most effective way to achieve this is through the modification of the training to testing environments.
This occurs after learning, with only minor changes due to the pronounced effect those minor changes have on the outcome metrics. The model cannot adapt once it is fully trained, even as an online system, because learning is no longer taking place. This means that a too great of change would result in a constant fail condition, as was found in initial testing.
Each environment was given two modifications related to the dimensions of the agent and the effects of the environment, such as gravity or agent length, as mentioned previously. This approach was chosen to explore how different types of dimensionality shifts affect the model's ability to generalize. By considering the agent and its environment as two distinct aspects of change, we can investigate the model's adaptability to various alterations in the given problem.

6.2.1 Modification

To modify the environments the classes of the environments are retrieved. This is accomplished by downloading Gymnasium [28] natively and taking the specific environments and storing them in an environmental folder. From here, they were given custom variables for their features relating to the agent and the environment to test the generalisability.
To accomplish the environmental modification successfully within the framework, implementation changes take place in order for the system to function. Normally environments in Gymnasium [28] can be simply passed on into the model, trained and then run. However, with the addition of the modified environments, a registration system had to be implemented. This required integrating the custom environments into the Stable Baseline registry, effectively making them apart of the system. Importantly, this allows for the vectorisation of the environments to take place, as Stable Baselines can only handle the vectorisation of integrated Gymnasium environments, reducing the training time required.
The first quality of life improvement is how the testing of custom environments functions in the Environmental Handler. As the user can give several environments for the model, which was trained on the base environment . Where the models are saved, the user is able to test against them at the same time in order to measure the

generalisability of the overall system, rather than having to do individual runs. This allows for more tests to be run in a shorter period of time, permitting for variations to be tested and in turn giving a better view of how the created algorithms influence the generalisability of the program.

Chapter 7

Experimental Results

Due to the volume of results obtained from the investigation and the repetition of results across environments, this chapter focuses on one representative environment, Pendulum, to increase clarity. Pendulum was chosen since it is sufficiently complex to yield meaningful results while not being so complicated as to prevent a full range of agent and environmental changes without causing the learning to fail. More complex environments, such as Bipedal Walker, presented difficulties due to the higher number of control aspects.

7.1 Interpretability

This section examines two critical aspects of interpretability: the ease of understanding the outcomes and how representative they are of the actual solution.
To show the effect of complexity on the loss between the SR output and the original policies, a sample is taken from the Pendulum Environment, giving the extracted equations with the rotational speed $x1$, position $x2$ and $x0$ current torque:

- $$\sin\left((\sin(x_1)\cdot x_2)\cdot\left(\frac{\sin(0.41658697)}{x_2}\right)\right) \tag{7.1}$$

  Complexity of 10 - Loss: 0.041929103

- $$\sin\left((x_2\cdot\sin\left(x_1+\sin(-0.12720555)\right))\cdot\left(\frac{\sin\left(\sin(0.41658697)\right)}{x_2}\right)\right) \tag{7.2}$$

  Complexity of 15 - Loss: 0.040418785

- $$\sin\left((x_2\cdot(\sin(x_0)\cdot 0.63439834+x_1))\cdot\left(\frac{0.25317}{x_2}\div(\cos(1.1621197)\cdot 0.25317+0.51962)\right)\right) \tag{7.3}$$

  Complexity of 20 - Loss: 0.040232357

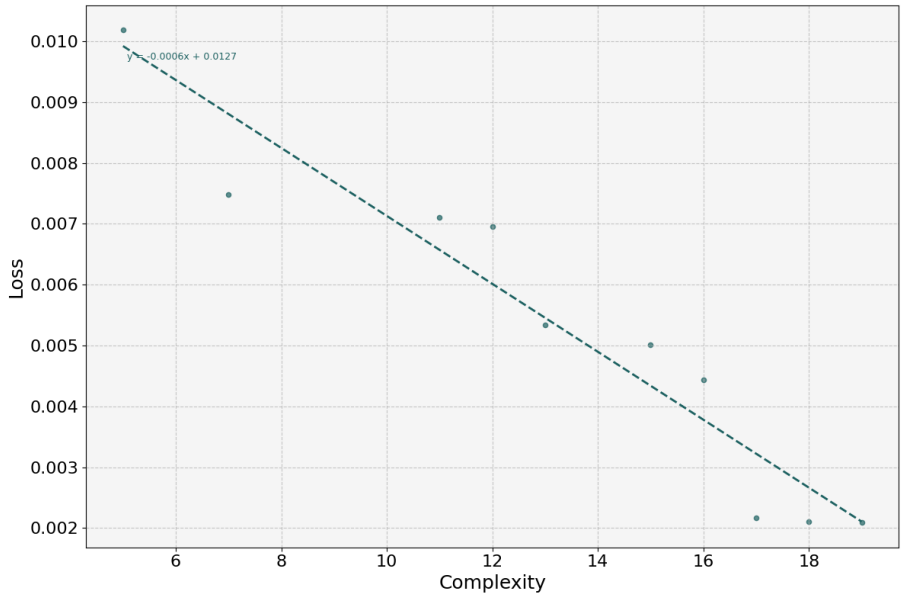

Figure 7.1: Loss Against Complexity - SLMA

A key observation is the legibility of the resulting equations. While subjective, this attribute, as previously described with the PySR structure, was sought out and gained through reducing variables and operators present to provide a clear solution.

Even though the equations have varying complexities, again affecting interpretability, there is a clear commonality between them, and the loss changes indicate that majority of behaviour is already captured. The loss change from Complexity 10 7.1 to Complexity 15 7.2 has a loss difference of 0.0016, which is important considering the precision required in the policy prediction. However, the difference between Complexity 15 and 20 is only 0.000186, showing a ten-fold decrease, suggesting diminishing returns with increasing complexity. This pattern was consistent across different runs.

To show this behaviour on a larger scale, the loss was graphed against complexity in Figure 7.2.

The starting complexities were removed because of the logarithmic nature of the relationship between loss and complexity 9.2. After a loss of 4, a relatively linear relationship emerges, $y = -0.0006x + 0.0127$. This relationship appears consistent across algorithms and environments. Importantly, complexities that did not show improvement were excluded from the results.

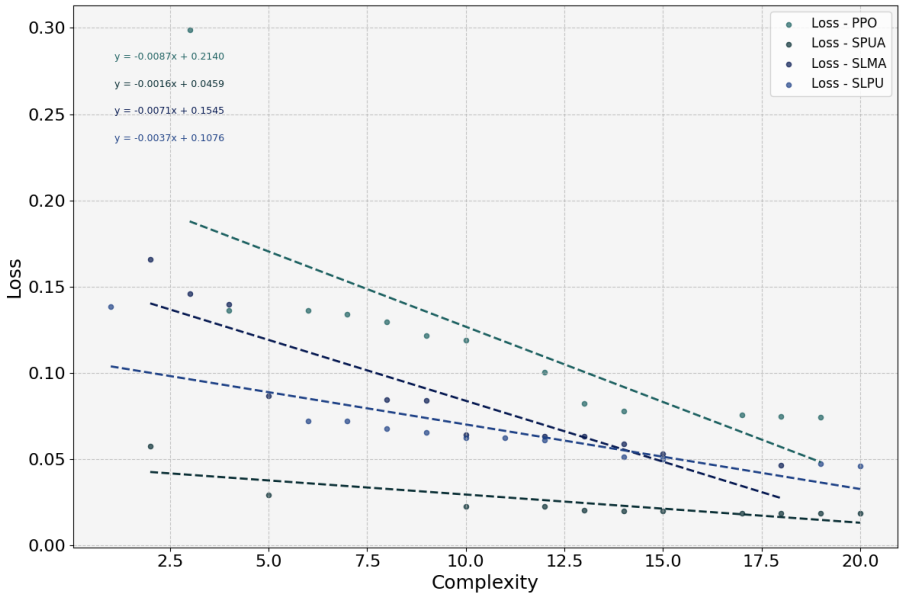

Figure 7.2: Loss Against Complexity

To understand how different systems further affect this loss, various algorithms were graphed and compared:
PPO was found to have the lowest interpretability, with the largest divergence between the SR output to the true values; followed by SLMA, with SPUA and SPLU demonstrating the best interpretability.

7.1.1 Discussion

Representation

The outcome was found to be more representative of the actual solution when influenced by SLMA, SPUA or SPLU, as shown in the loss values of Figure 7.2. This correlates with the level of Neurosymbolic interference, such as the increasing the ratio K, of the SR MSE Loss 2 against the other loss components; or the number of neurosteps N against regular time steps, both of which increased neurosymbolic interference during learning.
Not all algorithms have the same impact, which is most evident in the difference between SLMA and SPUA, with SPLU incorporating elements of both. This is represented in the loss difference between the extracted equation and the true policies,

with SPUA having a more significant effect on the learning than SLMA, as mentioned previously.
SPUA has a more direct and less manageable impact on the learning outcome. This causes a greater shift in the learning toward an outcome that is more easily represented symbolically. In contrast, SLMA still successfully steered the learning to symbolic solutions using the loss modification. The extracted outcome, in turn, becomes more representative of the final solution compared to PPO. Due to SLMA being a more measured approach, however, it did not achieve the same interpretability outcome.

Comprehension

Another consideration is how easily the solution is understood. Preliminary results suggest that higher complexity improves both generalisability and performance. However, the goal of reducing fine-tuning in the model seems to have been achieved, though perhaps excessively.
The highest complexity rating of 20 was chosen, which made comprehension more challenging. The constraints imposed by PySR on operators and variables, where more complicated operators were given a higher cost to insertion and excessive variables were trained against, resulted in an equation that is easily understood 7.3.
An interesting aspect is the non-linear change in loss between different symbolic complexities. There are clear jumps in loss between some complexities while others show minimal differences,. Moreover, the program additionally drops complexities that do not have an impact, represented by the gaps in Figure **??**. While exploring the reasons for these jumps is beyond this investigationâs scope, they provide greater insight into ideal complexity values and suggest that a more effective system would adapt to the complexity required to capture the behaviour, in order to save on computational intensity and give greater interpretability.
Furthermore, the system's modifiability allows users to understand the most critical features and their effects, reducing complexity and enhancing comprehension. This allows for key feature isolation and clarifies more complex equations.
The drift values also provide insight into the difference between what is captured and what is shown. Higher absolute policy values in Figure 7.3, which increase in frequency as the program learns, indicate a correlation with fine-tuning. This makes the behaviour harder to capture by SR, likely due to the increasing variance between the policy values as the model fine-tunes, with SR working against this effect, thereby promoting generalization.

7.2 Generalisability

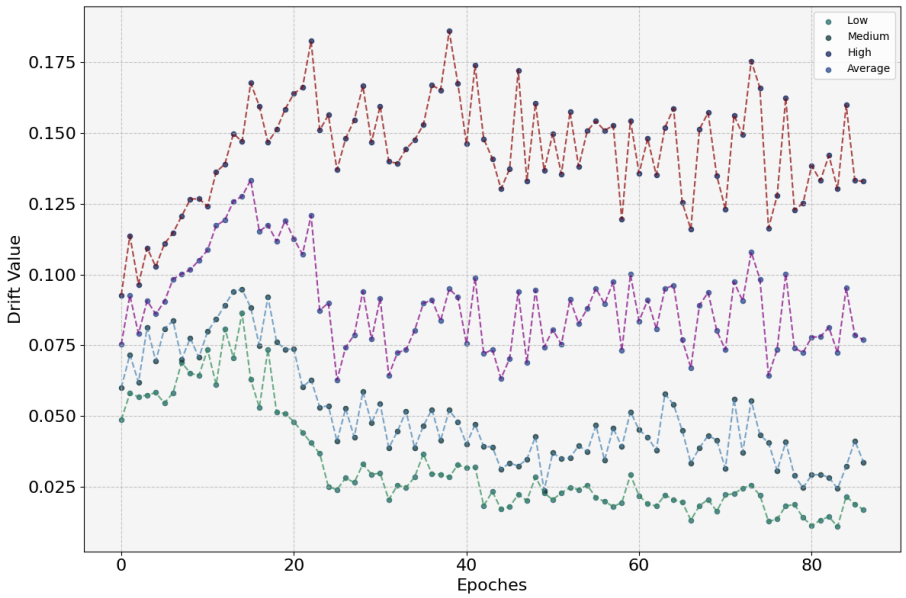

Figure 7.3: Drift Values

Figure 7.3 shows that larger drift values correlate with greater differences between the SR and PPO policies. The drift starts relatively small, peaks at the highest exploration point, and then stabilizes. Interestingly, there is a not a significant difference between low and medium drift values compared to high.

These changes influence the rest of the learning and push the model to generalise its behaviour more, partially due to this counter acting drift. This can be seen when looking at how the model acts in cases of dimensionality shifts, with small 12.5% and large 25% increases in the relative size.

Pendulum Change	SLMA	SPUA	SPLU	PPO
Length Increase - Small	**-183.010**	-190.221	-196.530	-183.229
Length Increase - Large	-529.968	-523.942	-502.113	**-471.758**
Gravity Increase - Small	**-217.951**	-277.181	-328.342	-288.640
Gravity Increase - Large	**-1021.026**	-1026.873	-1031.612	-1024.474

Table 7.1: Generalisation Mean Reward

Pendulum Change	SLMA	SPUA	SPLU	PPO
Length Increase - Small	153.621	**135.0511**	164.167	138.900
Length Increase - Large	**304.189**	307.982	358.343	316.977
Gravity Increase - Small	181.276	149.379	143.945	**135.951**
Gravity Increase - Large	276.830	276.245	275.445	**219.087**

Table 7.2: Generalisation Standard Deviation of Reward

These tables illustrate how the four algorithms respond to changes in both the environment and agent dimensions. SLMA demonstrates the most positive results when looking at Mean Reward, while PPO shows the best Standard Deviation of Reward (Table 7.1). Both PPO and SLMA display strong performance overall in both metrics.

7.2.1 Discussion

The greater the drift, the higher the fine-tuning and the lower the generalisation. The drift, or the difference in performance, is effectively captured by the algorithms as they influence backpropagation toward a more generalized solution. Thus, the larger the drift, the larger the difference, and the more do the algorithms correct the divergence back to the baseline. Another important note about drift is that it increases rapidly, peaks and then stabilizes slightly below the peak. This reflects a period of exploration before refinement, where higher variance makes values harder to capture.

Another aspect to note is that all small changes outperformed large changes across the environments. As expected, larger dimensionality shifts result in more significant errors, due the model operating outside of its training data. This is most evident with a large increase in gravity, where the model fails to adapt. The model becomes stuck in a failure position, suggesting that it cannot adapt to such a drastic change due to either its learning constraints or physical limitations.This pattern is also seen in more complex environments, where even small changes can have significant knock-on effects.

The difference in Mean Reward between changes in gravity and agent dimensions suggests that the model is more resistant to changes in the agent than in the environment. Small increases in length result in a smaller error increase compared to a proportionate increase in gravity. Additionally, the large increase in length does not cause the model to enter a constant failure state, unlike the gravity increase.

In terms of the algorithms, the metrics show that SLMA performs best in Mean Reward, while PPO excels in Standard Deviation of Reward. SLMAâs significantly

higher Mean Reward compared to PPO demonstrates that the investigation successfully improved the algorithm's generalisation, as SLMA outperforms PPO in custom environments due to the advantages conferred by neurosymbolic AI. While the performance gap is relatively minor in the examples shown here, it becomes more pronounced in other reinforcement learning environments such as CartPole 9.3 and Mountain Car 9.2, where SLMA showed a up to a 50% increase in the Mean Reward. Although the Mean Reward will differ across environments due to varying reward functions, the difference between PPO and the developed algorithms provides insight into the relative gains achieved.

The higher standard deviation in the created algorithms could be attributed to the learning instability introduced by the neurosymbolic aspect, as the standard deviation measures outcome consistency. The learning instability arises from interruptions in the regular learning cycle, and is additionally represented in the appendix data 9.3 9.4, showing a more stable outcome in general from PPO.

While the previous sections point out the differences between the algorithmic outcomes, they remain fairly comparable overall, which is a positive sign for the investigation. The goal was to make an improvement upon PPO without compromising its effectiveness, and the results in Tables 7.1 and 7.2 suggest that this objective has largely been achieved. Similar conclusions can be drawn in the performance section.

7.3 Performance

Performance was not the primary focus of this investigation; instead, it was deprioritized in favor of interpretability and generalization. However, evaluating performance remains essential to understand how the developed algorithms compare to existing systems. To assess this, two graphs â one depicting loss over time and the other showing mean reward â provide an overview of how well the models perform in a standard environment.

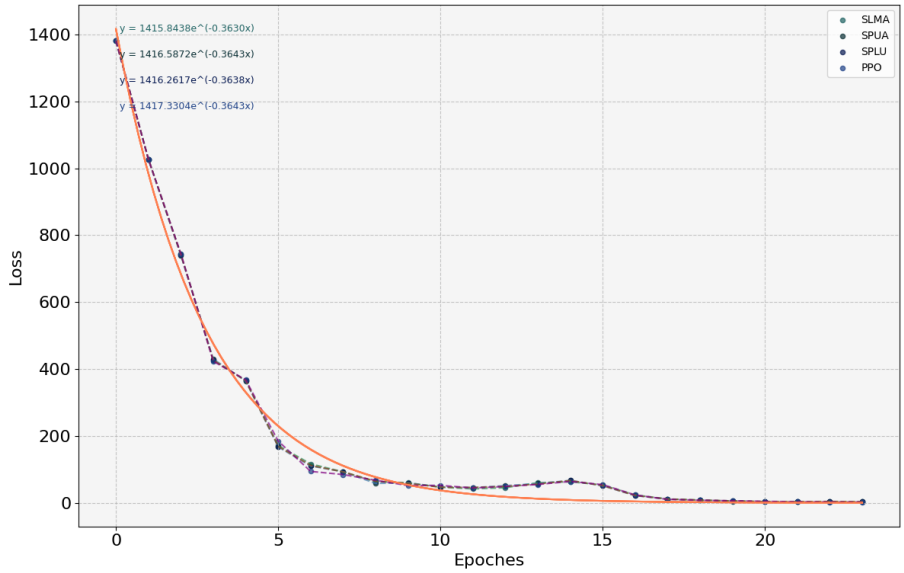

Figure 7.4: Loss Over Time

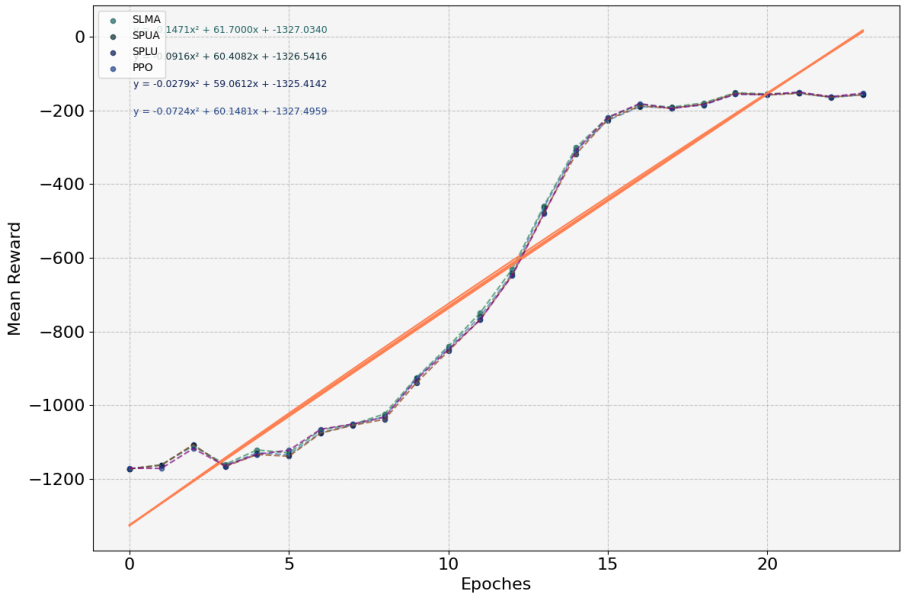

Figure 7.5: Mean Reward Over Time

The graphs 7.4, 7.5 reveal that the algorithmic outcomes are closely clustered, with the most significant deviation observed in PPO, which also exhibits the best average performance in terms of both loss and mean reward.

7.3.1 Discussion

While PPOâs advantage is apparent from the graphs, the addition of trendlines provides further insight into how the different systems perform and relate to one another. The trendlines confirm that PPO maintains the best average performance, a conclusion less evident in the original graph due to the close grouping of the data points.
Figure 7.4 also shows a slight increase in loss, particularly noticeable after epoch 5. While this deviation warrants mention, the minor difference is still a positive indicator for the learning system, as it demonstrates effective improvements in interpretability and generalization without substantial performance loss. The difference is again tied to the slight learning instability introduced by the created algorithms due to the level of neurosymbolic interference, as further illustrated by their trendlines, which differ from PPOâs optimal trendline.
While the new models either match or perform slightly worse than the base PPO system, they still perform remarkably well, especially considering their gains in other aspects. Algorithms like SLMA exhibited a negligible negative effect on performance while enhancing generalisation and interpretability. Moreover, all algorithms still solve all the environmental problems presented, converging to a solution at similar points, as seen where both figures stabilize in terms of learning 7.4 7.5. The tight clustering of the data points, with minimal variation, indicates that no supplemental steps were needed. This outcome lends support to the conclusion that the designed systems were effective in learning AI models.
The graphs 7.4 7.5 additionally provide important comparison metrics for the custom environment outputs, illustrating the loss when the model is applied to the intended environment. The significant changes indicate a need for further generalisation, even though SLMA was found to promote generalisation; there remain clear areas for growth.

7.4 Experimental Results Overview

The results indicate that there is no single best algorithm; each comes with tradeoffs. For interpretability, a stronger approach with SPUA and SPLU was found to

be most effective, as expected, given their direct impact on the learning process. The best balanced algorithm, however, was SLMA, which achieved a higher generalisation than both the newly developed and existing algorithms while also offering better interpretability than the standard PPO. Nevertheless, PPO demonstrated better consistency in outcomes, as indicated by the standard deviation in reward, particularly in custom environments, and still maintains an edge in performance against SLMA.
While these trade-offs are anticipated, there remains room for improvement, such as the issue identified previously of the program requiring SR to have a higher complexity limit in order to achieve the greatest outcome. It is important to note that all the conclusions gathered here also are also supported by the outcomes that took place in the other environments, as shown in the appendix for the loss 9.3, 9.5, 9.4 and mean reward 9.6, 9.7, 9.8. These results demonstrate a broader overall pattern of PPO performing better. However, SLMA's advantage continue to grow in both complex environments, such as Mountain Car 9.2, and simpler ones like Cartpole 9.3.

7.5 Initial Results

The preliminary results were used to explore the hyper-parameters, using the prescribed metrics (Table 9.5), and to gauge the overall viability of the system.
By employing the hyper-parameter testing system described earlier, and first examining PPO to determine the number of epochs needed for each environment to find a solution, the investigation systematically tested various parameters to identify the most effective values.

Table 7.3: Optimal Hyperparamters

	SLMA	**SPUA**	**SPLU**
Policy Ratio K	0.5	X	0.5
Neurostep N	2500	4000	4000
Complexity C	20	20	20
Gamma γ	0.98	0.98	0.98
Learning Rate α	0.0003	0.0003	0.0003

The selection criteria, as noted earlier, took a "generalisation first" approach, focusing on the mean reward obtained from custom environments. Due to generali-

sation being the most quantifiable and interpretability, even with a generalisation taking prominence, saw significant improvement. Additionally, it was found that the hyperparameters shared with PPO, such as the learning rate and gamma, were the most efficient for the new algorithms.

Based on these identified hyperparameters, the viability of the algorithms were then demonstrated by exploring the success rates of the different algorithms in the different environments. This was tested through 20 individual test runs on the standard environments, understanding that, if the algorithms cannot solve the environments then their testing capability will be severely limited.

Table 7.4: Algorithm Success Rate

	SLMA	**SPLU**	**SPUA**	**PPO**
CartPol	100.0%	100.0%	100.0%	100.0%
Pendulum	100.0%	95%	95%	100.0%
MountainCar	95.0%	90%	85.0%	95%
Bipedal Walker	95.0%	90.0%	90.0%	95.0%

As seen in Table 7.4, SLMA and PPO emerged as the leading algorithms in terms of success rates. This is to be expected considering the results of the investigations. However, the findings were novel at the start of this investigation and provided valuable insights into which processes would be most effective.

To assess computational efficiency, the runtime of each algorithm was compared to that of PPO.

Table 7.5: Algorithm Time Complexity

	SLMA	**SPLU**	**SPUA**	**PPO**
Run Time	125.0%	115.0%	115.0%	100.0%

As demonstrated in Table 7.5, there was a linear increase in time complexity corresponding to the number of neurosteps, as each additional neurostep results in an SR step. The higher number of neurosteps in SLMA led to greater computational time compared to the other algorithms.

7.5.1 Discussion

The choice of hyperparameters reveals important insights. One key finding is that a complexity level of 20 was optimal for both performance and generalization. This suggests that if higher complexity levels were available, they might have been selected, indicating that the SR algorithm may not be capturing behavior optimally and leaving room for further improvement by enhancing the SR algorithm itself. Other aspects, such as the number of neurosteps, align with expectations. SPUA has a stronger effect on the outcome than SLMA, because it is not weighted, as discussed when designing the algorithms. The optimal policy ratio weighting was found to be 0.5, emblematic of the balance between neural to symbolic interference sought in its implementation.

It is crucial to note that even with the symbolic interference and the fact that the model solved the problem, this does not mean that the model gave the best solution. This is further corroborated by the performance section. Even though the algorithms only slightly underperformed against PPO, the impact becomes less negligible as more difficult models are introduced, such as Bipedal Walker 7.4. This again suggests that the system has greater issues capturing higher levels of complexity required as the problems become more advanced.

The drift values also showed promise. Although not displayed here, the results were consistent with the findings in the previous section 7.3, validating that the system indeed alters the policy values generated. As shown in Figure 7.3, there is a clear difference between the new and old values, with the most substantial changes occurring with higher drift values, as discussed earlier.

Chapter 8

Conclusion

This thesis explored the integration of symbolic regression into reinforcement learning, specifically the Proximal Policy Optimization (PPO) algorithm, to enhance interpretability and generalizability. Novel algorithms - SLMA, SPUA, and SLPU - were developed and tested across various Gymnasium environments, demonstrating improvements in both interpretability and generalisation compared to standard PPO, albeit with some trade-offs in performance.

The results indicate that incorporating SR during learning can steer reinforcement learning models towards more interpretable and generalisable solutions. SPUA showed the most substantial gains in interpretability, while SLMA provided a better balance between interpretability, generalizability, and performance. The combined SLPU algorithm was particularly promising, as it could leverage the strengths of both approaches if given the proper modification for asynchronous use of its SLMA and SPUA features.

However, this research also revealed several challenges. A key issue was implementing symbolic regression into existing systems, namely data issues in moving from neural to symbolic representation and vice versa, were resolved throughout the report by modifying the system handling the detaching and reattaching of the gradient and correcting format and data type, these were new problems due to novel nature of the work.

A limiting aspect is computational complexity learning, which showed the need for more refined SR techniques tailored to reinforcement learning contexts that are effectively able to capture more complex relationships. Despite these challenges, the potential benefits of safety and explainability make this a promising direction for future research.

As AI systems become increasingly prevalent in critical applications, the ability

to understand and trust their decision-making processes will only grow in importance. This work represents a significant step towards more interpretable and generalisable reinforcement learning models and contributes to the responsible development of AI technologies. Additionally, it is usable in reinforcement learning systems now for interpretability and generalisability gains, aided by the easy integration capabilities of the user-friendly framework. For this reason, the code base will be made publicly available in the future.

8.1 Future Work

Future research should address several issues that were found throughout the investigation procedure. One aspect to address is the computational efficiency challenges by adapting SR methods, like PySR, for GPU compatibility. This would significantly reduce training times and, in turn, allow for a greater number of use cases.

Efforts should also be made to refine SR techniques to better capture the unique behaviours of reinforcement learning, this would involve increasing the maximum complexity obtainable by SR models. The computational weight of this aspect can be reduced by developing adaptive complexity systems, that adjust the required complexity based on the environment and task difficulty. This could lead to more efficient and interpretable models and raises the potential for real-time learning to further enhance the model's generalisation, as it can then adapt during use.

Enhancing the interpretability of the extracted equations is another crucial area for future work. This could involve providing more detailed information about the variables used in these equations, making them more accessible and understandable to end-users.

Using SR as an ”anchor” for policy creation also warrants further investigation, as it could potentially lead to more stable and interpretable learning processes. Even though the algorithms were found to increase learning instability, this does not mean that they do not successfully restrict the solution, as was found with the increase in generalisability.

Future research should also focus on determining optimal complexity values that adapt to the required behaviour capture while minimising computational intensity. This could lead to more efficient and interpretable models. In addition, exploring other potential neurosymbolic algorithms that were conceptualised but not implemented in this study could uncover new approaches to combining symbolic and neural methods in reinforcement learning. Developing asynchronous implementa-

tions of these algorithms could address performance overhead issues while allowing for different neurosymbolic steps for SPUA and SLMA, providing even greater customisation options.
Finally in terms of testing, exploring the application of these algorithms in a broad range of reinforcement learning scenarios and with other base algorithms beyond PPO would also yield valuable insights into their interpretability and generalisation capabilities across a broader range of environments and tasks would provide a more comprehensive understanding of their strengths and limitations. This holistic approach to future work aims to address the current limitations and further advance the field of interpretable and generalisable reinforcement learning, ultimately contributing to developing more trustworthy and transparent AI systems to create truly safe AI.

Bibliography

[1] Oludare Isaac Abiodun et al. “State-of-the-art in artificial neural network applications: A survey”. In: *Heliyon* 4.11 (2018) (cit. on p. 1).

[2] Joshua Achiam et al. “Constrained policy optimization”. In: *International conference on machine learning*. PMLR. 2017, pp. 22–31 (cit. on p. 30).

[3] Gustavo Aguilar et al. “Knowledge distillation from internal representations”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 7350–7357 (cit. on p. 19).

[4] Mohammed Alshiekh et al. “Safe reinforcement learning via shielding”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018 (cit. on p. 30).

[5] Mohammed Alshiekh et al. “Safe reinforcement learning via shielding”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018 (cit. on p. 30).

[6] Greg Anderson et al. “Neurosymbolic reinforcement learning with formally verified exploration”. In: *Advances in neural information processing systems* 33 (2020), pp. 6172–6183 (cit. on pp. 3, 14, 20, 21, 30, 46).

[7] Jacob Andreas, Dan Klein, and Sergey Levine. “Modular multitask reinforcement learning with policy sketches”. In: *International conference on machine learning*. PMLR. 2017, pp. 166–175 (cit. on p. 29).

[8] Masataro Asai and Alex Fukunaga. “Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary”. In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 32. 1. 2018 (cit. on p. 18).

[9] Samy Badreddine et al. “Logic Tensor Networks”. In: *Artificial Intelligence* 303 (Feb. 2022), p. 103649. ISSN: 0004-3702. DOI: 10.1016/j.artint.2021.103649. URL: http://dx.doi.org/10.1016/j.artint.2021.103649 (cit. on p. 17).

[10] Francisco Baeta et al. “Speed benchmarking of genetic programming frameworks”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2021, pp. 768–775 (cit. on p. 40).

[11] Dor Bank, Noam Koenigstein, and Raja Giryes. “Autoencoders”. In: *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook* (2023), pp. 353–374 (cit. on p. 18).

[12] Kanadpriya Basu et al. “Artificial intelligence: How is it changing medical sciences and its future?” In: *Indian journal of dermatology* 65.5 (2020), p. 365 (cit. on p. 1).

[13] Peter W Battaglia et al. “Relational inductive biases, deep learning, and graph networks”. In: *arXiv preprint arXiv:1806.01261* (2018) (cit. on pp. 22, 23).

[14] Michael van Bekkum et al. *Modular Design Patterns for Hybrid Learning and Reasoning Systems: a taxonomy, patterns and use cases*. 2021. arXiv: 2102.11965 [cs.AI] (cit. on p. 19).

[15] Luca Biggio et al. “Neural symbolic regression that scales”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 936–945 (cit. on pp. 22, 38, 40).

[16] Kathrin Blagec et al. “A critical analysis of metrics used for measuring progress in artificial intelligence”. In: *arXiv preprint arXiv:2008.02577* (2020) (cit. on p. 46).

[17] Denny Borsboom. “Latent variable theory”. In: (2008) (cit. on p. 23).

[18] Susan Bovair and David E Kieras. “A guide to propositional analysis for research on technical prose”. In: *Understanding expository text*. Routledge, 2017, pp. 315–362 (cit. on p. 18).

[19] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL] (cit. on p. 23).

[20] Charles George Broyden. “The convergence of a class of double-rank minimization algorithms 1. general considerations”. In: *IMA Journal of Applied Mathematics* 6.1 (1970), pp. 76–90 (cit. on p. 13).

[21] Andres Campero et al. “Logical rule induction and theory learning using neural theorem proving”. In: *arXiv preprint arXiv:1809.02193* (2018) (cit. on p. 17).

[22] Michael B. Chang et al. *A Compositional Object-Based Approach to Learning Physical Dynamics*. 2017. arXiv: 1612.00341 [cs.AI] (cit. on p. 22).

[23] Nicholas Chen et al. “Global economic impacts associated with artificial intelligence”. In: *Analysis Group* 1 (2016) (cit. on p. 1).

[24] Miles Cranmer. “Interpretable machine learning for science with PySR and SymbolicRegression. jl”. In: *arXiv preprint arXiv:2305.01582* (2023) (cit. on pp. 5, 12, 40).

[25] Miles Cranmer et al. *Discovering Symbolic Models from Deep Learning with Inductive Biases*. 2020. arXiv: 2006.11287 [cs.LG] (cit. on pp. 10, 20, 21, 23, 40).

[26] Y Cui, M Zhang, and Q Tang. “SYMBOLIC REGRESSION FOR FORMULATION OF SHEAR RESISTANCE OF BEARING-TYPE BOLTED CONNECTIONS”. In: *The 17th World Conference on Earthquake Engineering* (2020) (cit. on p. 11).

[27] Lauren Nicole DeLong et al. “Neurosymbolic ai for reasoning on graph structures: A survey”. In: *arXiv preprint arXiv:2302.07200* (2023) (cit. on pp. 21, 22).

[28] The Gymnasium Developers. *Gymnasium: A Toolkit for Developing and Comparing Reinforcement Learning Algorithms*. https://github.com/Farama-Foundation/Gymnasium. 2023 (cit. on pp. 5, 14, 30, 49).

[29] Happiness Ugochi Dike et al. “Unsupervised learning based on artificial neural network: A review”. In: *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*. IEEE. 2018, pp. 322–327 (cit. on p. 25).

[30] Honghua Dong et al. “Neural Logic Machines”. In: *CoRR* abs/1904.11694 (2019). arXiv: 1904.11694. URL: http://arxiv.org/abs/1904.11694 (cit. on p. 2).

[31] Yilun Du, Shuang Li, and Igor Mordatch. “Compositional visual generation and inference with energy based models”. In: *arXiv preprint arXiv:2004.06030* (2020) (cit. on p. 27).

[32] Logan Engstrom et al. “Implementation matters in deep policy gradients: A case study on ppo and trpo”. In: *arXiv preprint arXiv:2005.12729* (2020) (cit. on p. 8).

[33] Mirco Fabbri and Gianluca Moro. “Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks.” In: *Data*. 2018, pp. 142–153 (cit. on p. 1).

[34] Jianqing Fan et al. “A theoretical analysis of deep Q-learning”. In: *Learning for dynamics and control*. PMLR. 2020, pp. 486–489 (cit. on pp. 8, 38).

[35] Jesse Farebrother, Marlos C. Machado, and Michael Bowling. *Generalization and Regularization in DQN*. 2020. arXiv: 1810.00123 [cs.LG] (cit. on p. 25).

[36] Vitaly Feldman and Chiyuan Zhang. "What neural networks memorize and why: Discovering the long tail via influence estimation". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2881–2891 (cit. on p. 2).

[37] Nathan Fulton and André Platzer. "Safe reinforcement learning via formal methods: Toward safe control through proof and learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018 (cit. on p. 30).

[38] Nathan Fulton and André Platzer. "Verifiably safe off-model reinforcement learning". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2019, pp. 413–430 (cit. on p. 30).

[39] Javier García and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *J. Mach. Learn. Res.* 16 (2015), pp. 1437–1480. URL: https://api.semanticscholar.org/CorpusID:2497153 (cit. on p. 3).

[40] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480 (cit. on p. 30).

[41] Marta Garnelo and Murray Shanahan. "Reconciling deep learning with symbolic artificial intelligence: representing objects and relations". In: *Current Opinion in Behavioral Sciences* 29 (2019), pp. 17–23 (cit. on p. 2).

[42] Justin Gilmer et al. "Neural message passing for quantum chemistry". In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272 (cit. on p. 23).

[43] Peter Graf and Patrick Emami. *Three Pathways to Neurosymbolic Reinforcement Learning with Interpretable Model and Policy Networks*. 2024. arXiv: 2402.05307 [cs.AI]. URL: https://arxiv.org/abs/2402.05307 (cit. on pp. 21, 28, 29).

[44] Kevin Gurney. *An introduction to neural networks*. CRC press, 2018 (cit. on p. 1).

[45] Dong Han et al. “A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation”. In: *Sensors* 23.7 (2023). ISSN: 1424-8220. URL: https://www.mdpi.com/1424-8220/23/7/3762 (cit. on pp. 8, 10).

[46] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. “A survey and critique of multiagent deep reinforcement learning”. In: *Autonomous Agents and Multi-Agent Systems* 33.6 (Oct. 2019), 750â797. ISSN: 1573-7454. DOI: 10.1007/s10458-019-09421-1. URL: http://dx.doi.org/10.1007/s10458-019-09421-1 (cit. on p. 3).

[47] Hans-Martin Heyn, Eric Knauss, and Patrizio Pelliccione. “A compositional approach to creating architecture frameworks with an application to distributed AI systems”. In: *Journal of Systems and Software* 198 (2023), p. 111604 (cit. on p. 29).

[48] Irina Higgins et al. “Darla: Improving zero-shot transfer in reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1480–1490 (cit. on p. 27).

[49] Takuya Hiraoka et al. “Meta-Model-Based Meta-Policy Optimization”. In: *Proceedings of The 13th Asian Conference on Machine Learning*. Ed. by Vineeth N. Balasubramanian and Ivor Tsang. Vol. 157. Proceedings of Machine Learning Research. PMLR, 2021, pp. 129–144. URL: https://proceedings.mlr.press/v157/hiraoka21a.html (cit. on p. 29).

[50] Robert Hoehndorf, Núria Queralt-Rosinach, et al. “Data science and symbolic AI: Synergies, challenges and opportunities”. In: *Data Science* 1.1-2 (2017), pp. 27–38 (cit. on p. 2).

[51] Matthias Hofer et al. “Learning evolved combinatorial symbols with a neuro-symbolic generative model”. In: *arXiv preprint arXiv:2104.08274* (2021) (cit. on pp. 21, 24, 25).

[52] Vasant Honavar. “Symbolic artificial intelligence and numeric artificial neural networks: towards a resolution of the dichotomy”. In: *Computational architectures integrating neural and symbolic processes: a perspective on the state of the art*. Springer, 1995, pp. 351–388 (cit. on p. 5).

[53] Phillip Howard et al. “NeuroComparatives: Neuro-Symbolic Distillation of Comparative Knowledge”. In: *arXiv preprint arXiv:2305.04978* (2023) (cit. on pp. 21, 23, 24).

[54] Drew A Hudson and Christopher D Manning. “Compositional attention networks for machine reasoning”. In: *arXiv preprint arXiv:1803.03067* (2018) (cit. on p. 26).

[55] Ying Jin et al. “Bayesian symbolic regression”. In: *arXiv preprint arXiv:1910.08892* (2019) (cit. on pp. 40, 42).

[56] Justin Johnson et al. “Inferring and executing programs for visual reasoning”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2989–2998 (cit. on p. 26).

[57] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285 (cit. on pp. 7, 46).

[58] Janis Keuper and Franz-Josef Preundt. “Distributed training of deep neural networks: Theoretical and practical limits of parallel scalability”. In: *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*. IEEE. 2016, pp. 19–26 (cit. on p. 2).

[59] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680 (cit. on p. 12).

[60] Bart Kosko and Satoru Isaka. “Fuzzy logic”. In: *Scientific American* 269.1 (1993), pp. 76–81 (cit. on p. 28).

[61] John R Koza. “Genetic programming as a means for programming computers by natural selection”. In: *Statistics and computing* 4 (1994), pp. 87–112 (cit. on p. 10).

[62] Mikel Landajuela et al. “A unified framework for deep symbolic regression”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 33985–33998 (cit. on pp. 40, 42).

[63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444 (cit. on pp. 22, 23, 45).

[64] Yann LeCun et al. “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0 (2006) (cit. on p. 27).

[65] Yingming Li, Ming Yang, and Zhongfei Zhang. “A Survey of Multi-View Representation Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 31.10 (2019), pp. 1863–1883. DOI: 10.1109/TKDE.2018.2872063 (cit. on pp. 22, 23).

[66] Zewen Li et al. “A survey of convolutional neural networks: analysis, applications, and prospects”. In: *IEEE transactions on neural networks and learning systems* (2021) (cit. on p. 23).

[67] Yichao Liang, Josh Tenenbaum, Tuan Anh Le, et al. “Drawing out of Distribution with Neuro-Symbolic Generative Models”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 15244–15254 (cit. on pp. 21, 24, 25).

[68] Baihan Lin, Djallel Bouneffouf, and Irina Rish. “A Survey on Compositional Generalization in Applications”. In: (2023). arXiv: 2302.01067 [cs.AI] (cit. on p. 2).

[69] Yifang Ma et al. “Artificial intelligence applications in the development of autonomous vehicles: a survey”. In: *IEEE/CAA Journal of Automatica Sinica* 7.2 (2020), pp. 315–329. DOI: 10.1109/JAS.2020.1003021 (cit. on p. 1).

[70] Nour Makke and Sanjay Chawla. “Interpretable scientific discovery with symbolic regression: a review”. In: *Artificial Intelligence Review* 57.1 (2024), p. 2 (cit. on pp. 10, 32, 40).

[71] Spyros Makridakis. “The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms”. In: *Futures* 90 (2017), pp. 46–60. ISSN: 0016-3287. DOI: https://doi.org/10.1016/j.futures.2017.03.006. URL: https://www.sciencedirect.com/science/article/pii/S0016328717300046 (cit. on p. 1).

[72] Robin Manhaeve et al. *DeepProbLog: Neural Probabilistic Logic Programming*. 2018. arXiv: 1805.10872 [cs.AI] (cit. on p. 2).

[73] Jiayuan Mao et al. *The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision*. 2019. arXiv: 1904.12584 [cs.CV] (cit. on p. 2).

[74] Jiayuan Mao et al. *The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision*. 2019. arXiv: 1904.12584 [cs.CV] (cit. on pp. 18, 21, 24–26).

[75] Giuseppe Marra and Ondřej Kuželka. “Neural markov logic networks”. In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 908–917 (cit. on p. 17).

[76] Giuseppe Marra et al. “Integrating Learning and Reasoning with Deep Logic Models”. In: *CoRR* abs/1901.04195 (2019). arXiv: 1901.04195. URL: http://arxiv.org/abs/1901.04195 (cit. on p. 2).

[77] Kevin R. McKee et al. “Quantifying the effects of environment and population diversity in multi-agent reinforcement learning”. In: *Autonomous Agents and Multi-Agent Systems* 36.1 (Mar. 2022). ISSN: 1573-7454. DOI: 10.1007/s10458-022-09548-8. URL: http://dx.doi.org/10.1007/s10458-022-09548-8 (cit. on p. 3).

[78] Russell Mendonca et al. “Guided meta-policy search”. In: *Advances in Neural Information Processing Systems* 32 (2019) (cit. on p. 29).

[79] Ann Meulders, Kristof Vandael, and Johan W.S. Vlaeyen. “Generalization of Pain-Related Fear Based on Conceptual Knowledge”. In: *Behavior Therapy* 48.3 (2017), pp. 295–310. ISSN: 0005-7894. DOI: https://doi.org/10.1016/j.beth.2016.11.014. URL: https://www.sciencedirect.com/science/article/pii/S0005789416301174 (cit. on p. 2).

[80] Ludovico Mitchener et al. “Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework”. In: *Machine Learning* 111.4 (2022), pp. 1523–1549 (cit. on p. 21).

[81] P Mogensen and A Riseth. “Optim: A mathematical optimization package for Julia”. In: *Journal of Open Source Software* 3.24 (2018) (cit. on p. 13).

[82] Alexandre Moreira Nascimento et al. “A Systematic Literature Review About the Impact of Artificial Intelligence on Autonomous Vehicle Safety”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.12 (2020), pp. 4928–4946. DOI: 10.1109/TITS.2019.2949915 (cit. on p. 1).

[83] MZ Naser and Amir H Alavi. “Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences”. In: *Architecture, Structures and Construction* (2021), pp. 1–19 (cit. on p. 46).

[84] Mausam Natarajan and Andrey Kolobov. *Planning with Markov decision processes: An AI perspective*. Springer Nature, 2022 (cit. on p. 27).

[85] Andreas Nieder. “Neural constraints on human number concepts”. In: *Current opinion in neurobiology* 60 (2020), pp. 28–36 (cit. on p. 19).

[86] Alessandro Oltramari et al. *Neuro-symbolic Architectures for Context Understanding*. 2020. arXiv: 2003.04707 [cs.AI] (cit. on p. 5).

[87] Alessandro Oltramari et al. “Neuro-symbolic architectures for context understanding”. In: *arXiv preprint arXiv:2003.04707* (2020) (cit. on p. 19).

[88] Fernando Orejas. “Symbolic graphs for attributed graph constraints”. In: *Journal of Symbolic Computation* 46.3 (2011), pp. 294–315 (cit. on p. 26).

[89] Joao Palotti et al. “Benchmark on a large cohort for sleep-wake classification with machine learning techniques”. In: *NPJ digital medicine* 2.1 (2019), p. 50 (cit. on p. 25).

[90] Chengrun Qiu et al. “Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications”. In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8577–8588 (cit. on pp. 38, 41).

[91] Antonin Raffin et al. *Stable Baselines3*. `https://github.com/DLR-RM/stable-baselines3`. 2021 (cit. on pp. 10, 14).

[92] Kislay Raj. “A neuro-symbolic approach to enhance interpretability of graph neural network through the integration of external knowledge”. In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023, pp. 5177–5180 (cit. on pp. 21, 22).

[93] Nicholas Rescher and Alasdair Urquhart. *Temporal logic*. Vol. 3. Springer Science & Business Media, 2012 (cit. on pp. 18, 25).

[94] Ryan Riegel et al. “Logical neural networks”. In: *arXiv preprint arXiv:2006.13155* (2020) (cit. on p. 28).

[95] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 22).

[96] Adam Santoro et al. “Symbolic Behaviour in Artificial Intelligence”. In: *CoRR* abs/2102.03406 (2021). arXiv: `2102.03406`. URL: `https://arxiv.org/abs/2102.03406` (cit. on pp. 2, 18).

[97] Franco Scarselli et al. “The graph neural network model”. In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80 (cit. on pp. 22, 38).

[98] Michael Schmidt and Hod Lipson. “Distilling free-form natural laws from experimental data”. In: *science* 324.5923 (2009), pp. 81–85 (cit. on p. 22).

[99] Edgar Schonfeld et al. “Generalized zero-and few-shot learning via aligned variational autoencoders”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8247–8255 (cit. on p. 27).

[100] Julian Schrittwieser et al. “Online and Offline Reinforcement Learning by Planning with a Learned Model”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 27580–27591. URL: `https://proceedings.neurips.cc/paper_files/paper/2021/file/e8258e5140317ff36c7f8225a3bf9590-Paper.pdf` (cit. on p. 10).

[101] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2018. arXiv: 1506.02438 [cs.LG]. URL: https://arxiv.org/abs/1506.02438 (cit. on pp. 9, 10).

[102] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017) (cit. on pp. 5, 8, 10, 14, 32).

[103] John Schulman et al. *Trust Region Policy Optimization*. 2017. arXiv: 1502.05477 [cs.LG]. URL: https://arxiv.org/abs/1502.05477 (cit. on pp. 8, 38, 41).

[104] Gesina Schwalbe. "Concept embedding analysis: A review". In: *arXiv preprint arXiv:2203.13909* (2022) (cit. on p. 26).

[105] A. Sharma et al. "Artificial Intelligence-Based Data-Driven Strategy to Accelerate Research, Development, and Clinical Trials of COVID Vaccine". In: *Biomed Res Int* 2022 (July 6, 2022), p. 7205241. DOI: 10.1155/2022/7205241 (cit. on p. 1).

[106] Andrew Silva and Matthew Gombolay. *Neural-encoding Human Experts' Domain Knowledge to Warm Start Reinforcement Learning*. 2020. arXiv: 1902.06007 [cs.LG] (cit. on p. 21).

[107] Andrew Silva et al. "Optimization methods for interpretable differentiable decision trees applied to reinforcement learning". In: *International conference on artificial intelligence and statistics*. PMLR. 2020, pp. 1855–1865 (cit. on p. 28).

[108] Niket Tandon, Gerard De Melo, and Gerhard Weikum. "Webchild 2.0: Fine-grained commonsense knowledge distillation". In: *Proceedings of ACL 2017, System Demonstrations*. 2017, pp. 115–120 (cit. on pp. 23, 24).

[109] Yuchi Tian et al. "Deeptest: Automated testing of deep-neural-network-driven autonomous cars". In: *Proceedings of the 40th international conference on software engineering*. 2018, pp. 303–314 (cit. on p. 1).

[110] Tony Tohme. "Advances in Symbolic Regression: From Generalized Formulation to Density Estimation and Inverse Problem". PhD thesis. Massachusetts Institute of Technology, 2024 (cit. on pp. 12, 40).

[111] Ilya O Tolstikhin et al. "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272 (cit. on p. 23).

[112] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023) (cit. on p. 23).

[113] Paul E Utgoff. *Machine learning of inductive bias*. Vol. 15. Springer Science & Business Media, 2012 (cit. on p. 23).

[114] Abhinav Verma et al. "Programmatically Interpretable Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 5045–5054. URL: https://proceedings.mlr.press/v80/verma18a.html (cit. on pp. 2, 3, 21, 29).

[115] Wei Wang et al. "A survey of zero-shot learning: Settings, methods, and applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–37 (cit. on pp. 26, 27).

[116] Xu Wang et al. "Deep reinforcement learning: A survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.4 (2022), pp. 5064–5078 (cit. on pp. 7, 9, 46, 47).

[117] Yiqun Wang, Nicholas Wagner, and James M Rondinelli. "Symbolic regression in materials science". In: *MRS Communications* 9.3 (2019), pp. 793–805 (cit. on p. 10).

[118] Tailin Wu et al. *ZeroC: A Neuro-Symbolic Model for Zero-shot Concept Recognition and Acquisition at Inference Time*. 2022. arXiv: 2206.15049 [cs.LG] (cit. on pp. 21, 26, 27).

[119] Weiming Xiang et al. "Verification for machine learning, autonomy, and neural networks survey". In: *arXiv preprint arXiv:1810.01989* (2018) (cit. on p. 30).

[120] Xuan Xie, Kristian Kersting, and Daniel Neider. *Neuro-Symbolic Verification of Deep Neural Networks*. 2022. arXiv: 2203.00938 [cs.AI] (cit. on p. 19).

[121] Hanhui Xu and Kyle Michael James Shuttleworth. "Medical artificial intelligence and the black box problem: a view based on the ethical principle of âdo no harmâ". In: *Intelligent Medicine* 4.1 (2024), pp. 52–57. ISSN: 2667-1026. DOI: https://doi.org/10.1016/j.imed.2023.08.001. URL: https://www.sciencedirect.com/science/article/pii/S2667102623000578 (cit. on p. 1).

[122] Yongjun Xu et al. "Artificial intelligence: A powerful paradigm for scientific research". In: *The Innovation* 2.4 (2021), p. 100179. ISSN: 2666-6758. DOI: https://doi.org/10.1016/j.xinn.2021.100179. URL: https://www.sciencedirect.com/science/article/pii/S2666675821001041 (cit. on p. 1).

[123] Kexin Yi et al. “Neural-symbolic vqa: Disentangling reasoning from vision and language understanding”. In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 26).

[124] Dongran Yu et al. “A survey on neural-symbolic learning systems”. In: *Neural Networks* 166 (2023), pp. 105–126. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2023.06.028. URL: https://www.sciencedirect.com/science/article/pii/S0893608023003398 (cit. on pp. 2, 14, 17, 19).

[125] Chiyuan Zhang et al. “A Study on Overfitting in Deep Reinforcement Learning”. In: *CoRR* abs/1804.06893 (2018). arXiv: 1804.06893. URL: http://arxiv.org/abs/1804.06893 (cit. on p. 1).

[126] He Zhao et al. “Topic modelling meets deep neural networks: A survey”. In: *arXiv preprint arXiv:2103.00498* (2021) (cit. on p. 25).

Chapter 9

Appendix

9.1 Equations

$$\eta(\tilde{\pi}) - \eta(\pi) = \mathbb{E}_{s_0, a_0, s_1, a_1, \ldots} \left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t) \right]$$

Figure 9.1: TRPO Objective Function

Algorithm 5 Proximal Policy Optimization (PPO)

Require: Initialize policy parameters θ_0, value function parameters ϕ_0

1: **for** iteration = 0, 1, 2, ... **do**

2: Collect trajectories $\mathcal{D} = \{\tau_i\}$ by running policy $\pi_{\theta_{old}}$ in the environment

3: Compute rewards-to-go $\hat{R}_t$ and advantage estimates $\hat{A}_t$ using the current value function V_ϕ

4: Update the value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

5: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t, \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right)$$

6: Update old policy parameters: $\theta_{old} \leftarrow \theta$

7: **end for**

9.2 Data

9.2.1 Loss

This section of the appendix focuses on the loss output from the various environments, importantly unmodified, with trend lines where appropriate for added clarity. Small is a 12.5% increase from the base value, large is a 25% increase, with unspecified being small, required as more would break the environment.

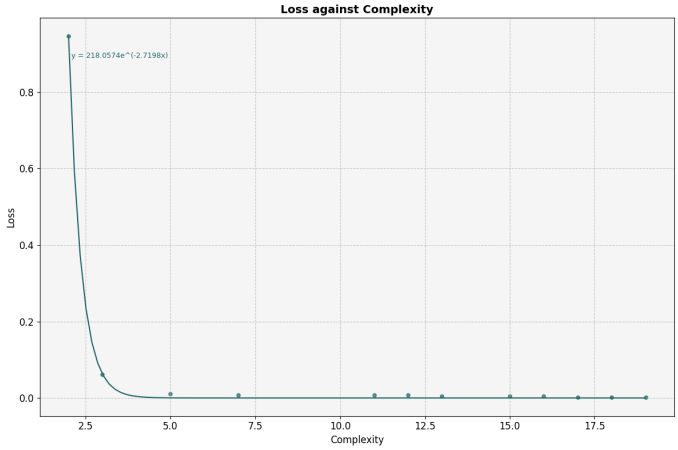

Figure 9.2: Loss Against Complexity - Full Data

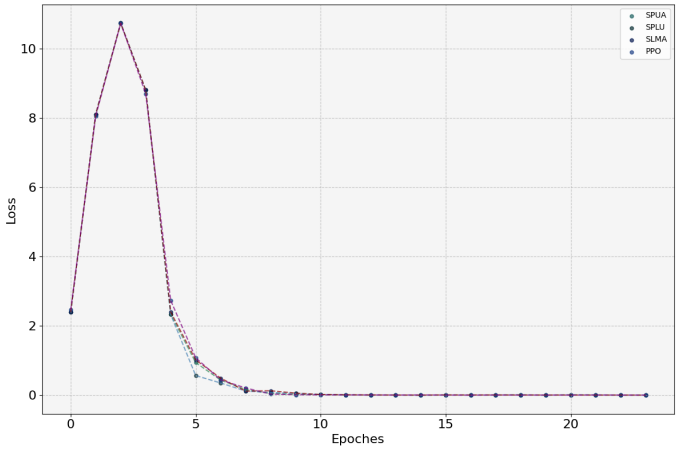

Figure 9.3: Loss - Cartpol

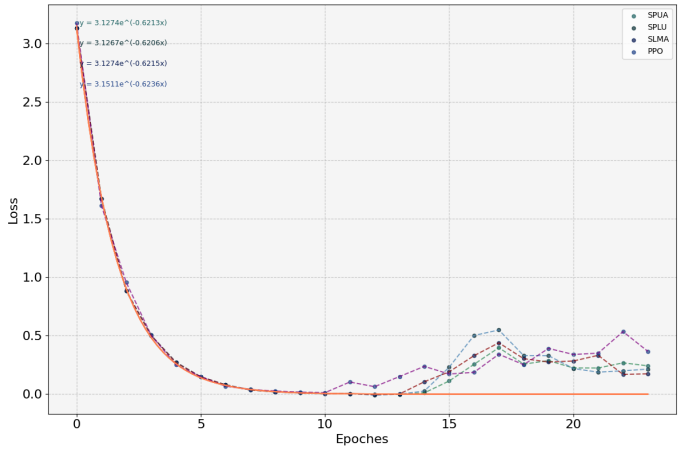

Figure 9.4: Loss - Mountain Car

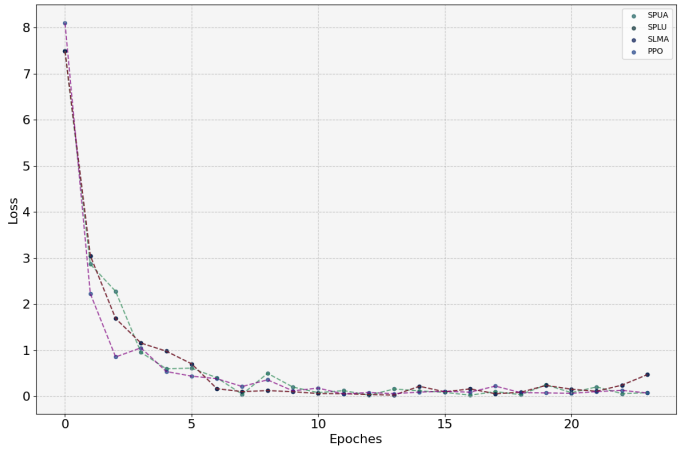

Figure 9.5: Loss - Bipedal Walker

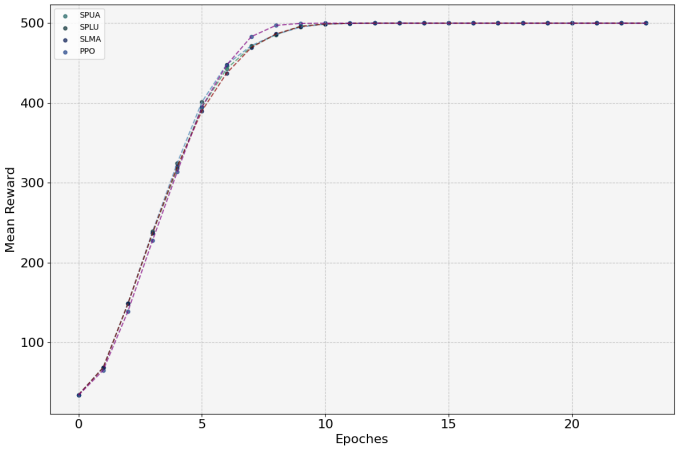

Figure 9.6: Mean Reward - Cartpol

9.2.2 Mean Reward

The same is done in this section for mean reward over the different environments, with the mean reward reflecting the environment and its specific reward function, making the difference between PPO and the created algorithms the most important aspect for insight.

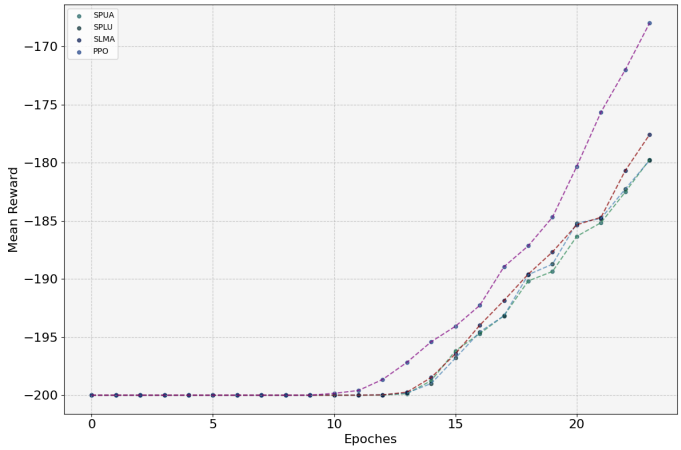

Figure 9.7: Mean Reward - Mountain Car

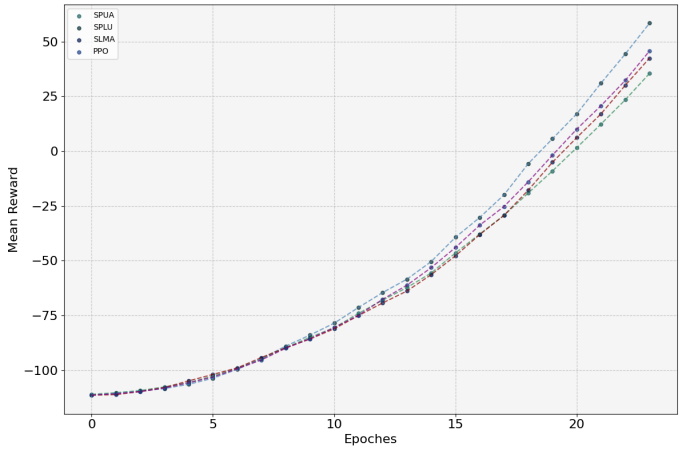

Figure 9.8: Mean Reward - Bipedal Walker

9.2.3 Mean Reward - Custom Environments

Cartpol Change	SLMA	SPUA	SPLU	PPO
Length Increase - Small	200	200	200	200
Length Increase - Large	102.32	141.46	**95.821**	153.875
Gravity Increase - Small	200	200	200	200
Gravity Increase - Large	200	200	200	200

Table 9.1: Cartpol - Mean Reward

Mountain Car Change	SLMA	SPUA	SPLU	PPO
Speed Decrease	-166.52	-173.28	-167.68	**-156.14**
Gravity Increase	**-162.98**	-177.08	-177.32	-172.38

Table 9.2: Mountain Car - Mean Reward

9.2.4 Stand Deviation of Reward - Custom Environments

Cartpol Change	SLMA	SPUA	SPLU	PPO
Length Increase - Small	0	0	0	0
Length Increase - Large	8.678614646	18.528	**4.103**	4.32
Gravity Increase - Small	0	0	0	0
Gravity Increase - Large	0	0	0	0

Table 9.3: Cartpol - Standard Deviation of Reward

Mountain Car Change	SLMA	SPUA	SPLU	PPO
Speed Decrease	**8.372315**	12.096	12.32899114	18.485
Gravity Increase	**2.201093**	4.567	4.601980958	15.279

Table 9.4: Mountain Car - Standard Deviation of Reward

9.3 Variables

9.3.1 Investigation Context - PPO

- **State** s: The current situation or configuration in which the agent finds itself.

- **Action** a: A choice made by the agent that affects the state.
- **Policy** π: A strategy used by the agent to determine the next action based on the current state.
- **State Value Function** $V(s)$: The expected cumulative reward from being in state s.
- **Action Value Function** $Q(s, a)$: The expected cumulative reward from taking action a in state s and following the policy π thereafter.
- **Advantage Function** $A(s, a)$: The relative value of taking action a in state s compared to the average action in that state, defined as $A(s, a) = Q(s, a) - V(s)$.
- **Temporal Difference Error** δ_t: The difference between the expected reward and the actual reward, calculated as $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$.
- **Reward** r_t: The immediate reward received after taking action a_t in state s_t.
- **Discount Factor** γ: A factor between 0 and 1 that determines the present value of future rewards.
- **GAE Parameter** λ: A parameter controlling the trade-off between bias and variance in Generalized Advantage Estimation (GAE).

9.3.2 Investigation Context - Symbolic Regression

- **Probability** p: Probability of rejecting a mutated individual in the simulated annealing process, aiding exploration by allowing less fit solutions.
- **Fitness of Mutated Individual** L_F: Fitness of the mutated individual, measuring how well it meets the optimization criteria.
- **Fitness of Original Individual** L_E: Fitness of the original individual before mutation, used to evaluate the mutation's impact.
- **Scale Parameter** α: Hyperparameter scaling the temperature T, controlling the sensitivity of acceptance probability to fitness changes.
- **Temperature** T: Temperature in the simulated annealing algorithm, governing the acceptance likelihood of less fit mutations for exploration.

- **Adjusted Loss** $\ell(E)$: The loss function of the expression E, incorporating both predictive loss and an adaptive complexity penalty.
- **Predictive Loss** $\ell_{\text{pred}}(E)$: The error of the expression E in fitting the data.
- **Frecency** frecency$[C(E)]$: A measure combining the frequency and recency of expressions with complexity $C(E)$ in the population, used to dynamically adjust the complexity penalty.
- **Complexity** $C(E)$: The complexity of the expression E, often defined by the number of nodes or operations in the expression tree.

9.3.3 Method

- **Number** n: Total number of observations in the dataset.
- **PPO Policy Value** θ_k: Policy value for the i-th observation.
- **SR Policy Value** θ_s: SR Policy value from the PPO Policy for the i-th observation.
- **Old Policy Value** θ_{old}: Used to generate trajectories and as a baseline in the PPO-Clip objective.
- **Set of Trajectories** D: Collected under the policy $\pi_{\theta_{\text{old}}}$, where each trajectory τ is a sequence of state-action-reward tuples.
- **Rewards-to-go** R_t: Computed as the sum of discounted future rewards from time t.
- **Advantage estimates** A_t: At time t, calculated as the difference between R_t and the value function estimate $V_\phi(s_t)$.
- **Value function** V_ϕ: Parameterized by ϕ.
- **Updated value function**ϕ_{k+1}: Updated value function parameters after regression on MSE.
- **Policy parameters** θ_{k+1}: Updated after optimizing the PPO-Clip objective.
- **Clipping parameter** ϵ: Limits the ratio of new policy to old policy probabilities within the range $[1-\epsilon, 1+\epsilon]$.

- **Frequency** N: Of applying symbolic regression to update policies.
- **Weighting factor** K: For the MSE in the overall loss function.

9.3.4 Experimental Setup

- **Expected Return** μ_R: The expected value of the return R. This is a statistical measure indicating the average potential return of an investment.
- **Standard Deviation of Return** σ_R: The standard deviation of the return R, representing the volatility or risk associated with the investment.
- **PPO Values** S_{pi}: The predicted values by the model for each data point.
- **Symbolic Regression Values** S_{ri}: The actual observed values for each data point.
- **Sample Size** n: The number of data points over which the loss is averaged.
- **Regularization Parameter** λ: A scalar that balances the trade-off between fitting the data well and keeping the model complexity low.
- **Complexity** $C(\theta)$: A measure of model complexity, parameterized by θ.
- **Feature Distance** D_f: Represents the average distance between the feature values θ_{ki} and θ_{si} across n samples, often used in comparing two sets of data or model parameters.
- **Feature Value from First Set** θ_{ki}: The feature value from the first set or model for the i-th data point.
- **Feature Value from Second Set** θ_{si}: The feature value from the second set or model for the i-th data point.
- **Sample Size** n: The number of data points over which the feature distance is averaged.
- **Weight** w : The weight given to moderate a particular output.

9.4 Metrics of Investigation

Table 9.5: Metrics for Evaluating Reinforcement Learning Models

Metric	Description
Loss 2,6.3	Measures the discrepancy between the predicted outcomes of a model and the actual results.
Mean Reward 6.1	Total reward accumulated over a series of episodes, used to evaluate long-term effectiveness, and a measure of the Domain Adaptation and Robustness
Standard Deviation of Reward 6.2	Quantifies the variation in rewards the agent receives, reflecting the consistency of its performance in an environment, and a measure of the domain adaptation and robustness
Drift Value 6.4	A Measure of how much the policies between Symbolic Regression and PPO vary, giving insight into what makes a behaviour harder to capture and allowing the tuning of the model to capture it more effectively. This is a metric that was invented to aid this investigation.
Weight and Gradient Update 6.5	The weight difference metric, calculated as the change in weights between neurostep updates provides real-time insight into the learning process by quantifying model adjustments, thereby validating that the expected changes are occurring during training.
Time	The simplest metric and will be simply looking at how long it takes for the program to run