ADRL Project Proposal

Thorben Klamt
`github.com/ThorKlm/ATRL_project`

Abstract

Deep Reinforcement Learning (DRL) has demonstrated remarkable capabilities, often surpassing human performance in various domains. However, its inherent lack of interpretability poses challenges, especially in safety-critical and real-world applications where understanding decision processes is essential. This project proposes a novel method for enhancing the interpretability of DRL agents by extracting symbolic policies from pre-trained models, using distinct activation formulas for each action. The approach simplifies decision-making while preserving performance. Through controlled experiments in environments with simple dynamics, such as CartPole-v1 and LunarLander-v2, it could be shown that symbolic policies can achieve performance comparable to DRL baselines, with significant reductions in model complexity, thus making them more suitable for real-world, interpretable applications.

1 Introduction

Deep Reinforcement Learning (DRL) has shown great potential in solving complex decision-making tasks. However, its lack of transparency limits its application in safety-critical systems. The challenge is to make DRL models more interpretable without sacrificing too much performance. Symbolic regression addresses this by converting the decision-making processes of pre-trained DRL agents into interpretable mathematical expressions, reducing complexity while retaining effectiveness. This study explores the trade-offs between simplicity and performance in symbolic policies, using environments with simple dynamics and aims to enhance the interpretability while utilizing key advantages of DRL.

2 Related Work

There has been significant efforts to integrate symbolic reasoning with DRL to enhance interpretability. For instance, "Discovering Symbolic Policies with Deep Reinforcement Learning" Landajuela et al. [2021] generates complex symbolic policies, but this method simplifies it by using distinct activation formulas for each action. Unlike DeepSynth Hasanbeig et al. [2024], which addresses complex, sparse-reward tasks, this project focuses on simpler environments like CartPole-v1 and LunarLander-v2, optimizing for low complexity and efficiency. This approach also contrasts with methods in "Optimization Methods for Interpretable Differentiable Decision Trees in Reinforcement Learning" Silva et al. [2020], "Explainability in DRL" Hickling et al. [2024], and "Deep Generative Symbolic Regression" Holt et al. [2024], which focus on optimizing decision trees or leveraging deep learning. This project instead extracts symbolic policies from pre-trained DRL agents to enhance transparency and simplicity in already-trained models. Although decision trees and symbolic models have their advantages, the choice depends on the task. Symbolic models extracted here would be particularly useful for continuous action spaces, despite the discrete environments used. PySR was employed for symbolic regression, similar to its original application for scientific insight discovery Cranmer [2023], offering a streamlined approach for simpler tasks.

3 Approach

The process to retrieve symbolic policies to solve a given task can be separated into two main steps: first, by either training DRL agents on a given task in a given environment or by utilizing accessible trained agents. In the second step, the symbolic policies are distilled using symbolic regression.

Algorithm 1 Symbolic Policy Distillation

```
Require: Pre-trained DRL policy $\pi_{DRL}$, environment $env$, sample number $m$, sampling probability
    $p$, complexity levels $C$, utility function $u$, evaluation episodes $n$
 1: Initialize dataset $D \leftarrow \emptyset$
 2: for $i \in 1..m$ do
 3:     Collect state-action pair $(s_i, a_i)$ from $\pi_{DRL}$ in $env$ during inference
 4:     Add $(s_i, a_i)$ to $D$ with probability $p$
 5: end for
 6: for each action $a \in A$ do
 7:     Retrieve symbolic models $M_{a,c}$ of different complexities $c$ using dataset $D$
 8: end for
 9: for each complexity level $c \in C$ do
10:     Construct symbolic policy $\pi^c_{SP}(s) \leftarrow \arg\max_a M_{a,c}(s)$
11: end for
12: Evaluate policies for $n$ episodes and return the policy $\pi^*_{SP}$ based on $u(\text{complexity}, \text{performance})$
```

State-action pairs are sampled from trained DRL agents at a low frequency (p=0.1) during inference to promote data diversity. Symbolic regression is then employed to formulate mathematical descriptions of decision-making strategies at various complexity levels, producing unique equations for each action to enhance interpretability. The optimal symbolic policy is selected either manually or via a utility function that balances simplicity with performance. The Symbolic Policy Distillation algorithm (Algorithm 1) can be embedded within a larger framework to derive interpretable symbolic policies from any MDP, by training a DRL-based policy from scratch utilizing any RL algorithm following Algorithm 2 (supplemental material).

4 Experiments

The proposed method for extracting symbolic policies from pre-trained Deep Reinforcement Learning (DRL) agents was evaluated in two environments: CartPole-v1 and LunarLander-v2. The evaluation focused on two key metrics: performance, measured by the evaluation reward, and model complexity, assessed by the number of symbols in the derived equations.

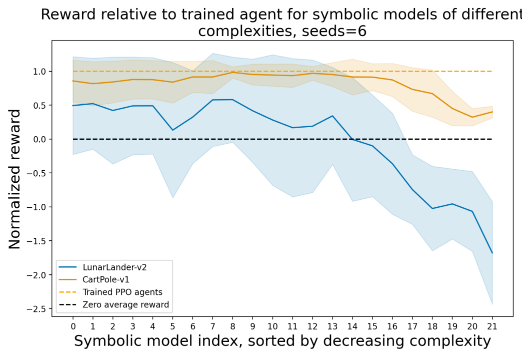

Figure 1: Performance analysis of symbolic models in CartPole-v1 and Lunar Lander-v2 environments using Stable Baselines3 Raffin et al. [2021] and openai gym Brockman et al. [2016], based on 6 seeds. In CartPole, the performance (red function) remains close to the DRL agent's baseline (orange dotted line) across most complexities, but declines at lower complexities. For Lunar Lander, performance starts moderate and decreases with simpler models. The reward variation between environments might be due to Lunar Lander allowing negative rewards, unlike CartPole.

In the here conducted experiments pre-existing Proximal Policy Optimization (PPO) models were utilized to enhance reproducibility and comparability and obtained from the huggingface framework from "sb3/ppo-CartPole-v1" and "sb3/ppo-LunarLander-v2", respectively. Comparisons between symbolic policies and DRL baselines were conducted in both environments, utilizing 25 evaluation episodes in the Cartpole environment and 10 evaluation episodes in the LunarLander environment over six random seeds. The PySR Cranmer [2023] library was employed for symbolic regression, with a limit of 10,000 samples to balance computational efficiency and accuracy. The experiments were conducted on Jupyter Notebooks Google Colab for easier reproducibility. The symbolic regression using pySR and the evaluation of retrieved symbolic policies was done on both CPU servers with additional RAM and T4 GPU servers. Multiple jupyter notebooks with all used parameters, configurations and additional info are available at the corresponding github repository for this project: `github.com/ThorKlm/ATRL_project`.

4.1 Cartpole environment

The CartPole environment, characterized by simple dynamics, enabled symbolic policies to achieve performance comparable to the DRL baseline across most complexity levels, though performance declined as models were strongly simplified.

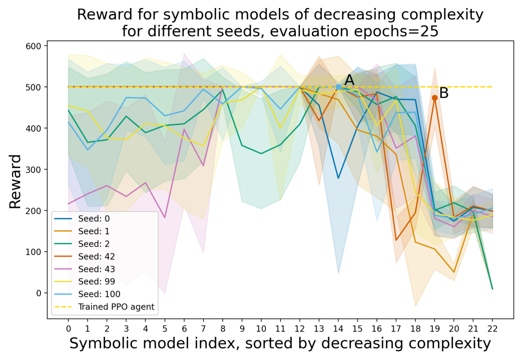

Figure 2: The relationship between model complexity and performance. Models A and B exhibit similar performance, but B has significantly lower complexity, making it a better candidate for applications prioritizing interpretability.

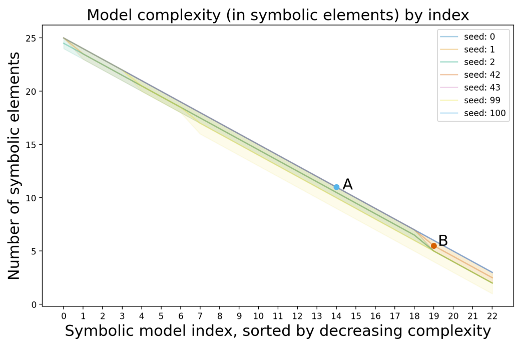

Figure 3: Symbolic model complexity by number of symbols relative to the model index. The index closely, but not entirely, reflects the inverse of complexity. Despite similar performances, models **A** and **B** (as in Figure 2) show significant differences in complexity.

Models A and B, shown in Figure 2 and Figure 3, exhibited similar performance, with model B displaying significantly lower complexity, making it preferable for applications prioritizing interpretability.

Table 1: Selected sample model equations

action name	equations model A	equations model B
going left	$relu(tanh(0.448 - ((x3 + (x2/0.558))/0.058)))$	$relu(cos(53.163^{x3}))$
going right	$relu(cos(0.879 - tanh(tanh((x2 + x3)/0.059))))$	$relu(x3 + x2)^{0.077}$

In the CartPole environment, the observation space consists of cart position, velocity, pole angle, and angular velocity.

4.2 Qualitative Analysis of Symbolic Policies in the CartPole environment

Models A and B (Table 1) both use feedback control mechanisms based primarily on cart velocity and pole angular velocity but differ in their complexity. Model A employs a more intricate feedback system, combining variables like pole angle and cart velocity with a tanh function for smoother left action responses and a cos term for more precise right action corrections. In contrast, Model B relies on simpler equations, focusing on rather noisy periodic stabilization using a cos function for the left action and a smaller exponent for more gradual control on the right. Despite these differences in complexity, both models achieve comparable performance.

4.3 Additional experiments and Lunar Lander environment

The analysis of state variable frequency in the extracted symbolic models revealed that pole angular velocity (x3) is the most frequently used variable across all complexities. As model complexity increases, additional variables such as cart position (x0) and cart velocity (x1) are incorporated, reflecting a broader capture of environmental dynamics. In the more complex LunarLander-v2 environment, which features a multi-dimensional action space and intricate physics, symbolic models struggle more compared to CartPole. Performance deteriorates more rapidly with reduced complexity, indicating that simple symbolic equations are less effective at capturing the dynamics of this environment.

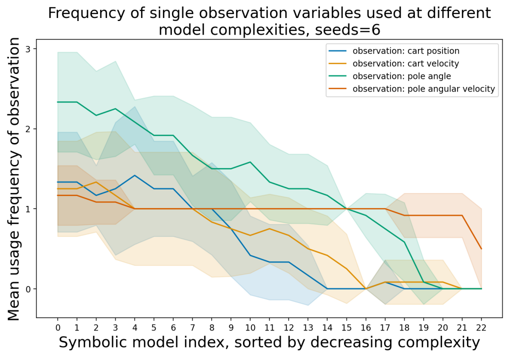

Figure 4: Frequency of state variables used in symbolic models across different complexities. Pole angular velocity (x3) is consistently utilized, while other variables become more prominent in models of higher complexity.

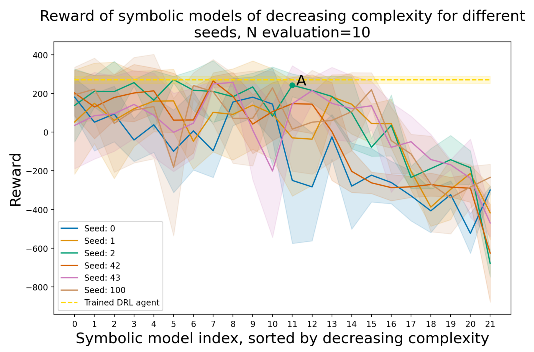

Figure 5: Performance comparison between symbolic models and the DRL baseline in LunarLander-v2. Performance declines faster with reduced complexity than in CartPole, suggesting more complex symbolic models are required for higher-dimensional environments.

However, certain models, such as represented by sample point A in Figure 5, achieve a reasonable balance between performance and complexity. This demonstrates that symbolic policy extraction remains feasible in complex settings, though more nuanced methods may be needed. LunarLander's diverse observation space-including positions (x0, x1), velocities (x2, x3), angle (x4), and angular velocity (x5) and left and right leg contact (x6, x7)-adds to the challenge of creating simple yet high performing symbolic representations that adequately capture the system's dynamics for stable maneuvers.

Table 2: Action Names and corresponding symbolic model of sample **A**

action name	equation
doing nothing	$abs(tanh(((-0.39965034 - tanh(x3))/(3.039439 - x1))/x3))$
firing left engine	$atanh\_clip(relu(0.15576722 - ((x7 + x5) + ((x4 - x2) \cdot 1.2659489))))$
firing main engine	$(tanh(relu(((-10.091324 \cdot x3) - 2.1895328) - x1)) - x4) - x5$
firing right engine	$relu(x4 + (sqrt(relu(x5) + 0.0095013315) - ((x2 + x0) + x7)))$

4.4 Qualitative Analysis of Symbolic Policies in the Lunar Lander environment

The symbolic equations for each action in the LunarLander environment for the LunarLander sample A (Table 2, Figure 5) (e.g., "doing nothing," "firing main engine") display reasonable decision-making patterns, using altitude (x1) and vertical velocity (x3). However, essential variables such as x-position (x0) were sometimes omitted in actions like "firing left engine" suggesting possible reasons for a slightly sub-optimal policy.

5 Discussion

This work demonstrates the feasibility of extracting symbolic policies from pre-trained DRL agents, achieving comparable performance to standard models in relatively simple environments like CartPole-v1 while significantly improving interpretability. However, in more complex environments like LunarLander-v2, limitations emerged. Particularly in capturing multi-dimensional dynamics with simplified symbolic models. In some cases, the retrieved policies also did not manage to incorporate underlying symmetries of the problem. Nevertheless, the results show that symbolic regression can offer transparent policies without major performance loss in the right contexts. As the main contribution a method was introduced balancing complexity and interpretability by generating distinct activation formulas for each action, enhancing transparency and making DRL models more applicable to safety-critical or simple tasks with the desire for high interpretability. Immediate future work should focus on refining the symbolic extraction process to handle more complex environments and multi-dimensional action spaces more effectively. This includes exploring continuous action spaces to fully utilize the benefits of analytical expressions over decision trees. Additional immediate future work could also be the investigation of symbolic intersections in clusters of symbolic regression solutions to discover and prioritize highly conserved symbolic policy fragments to further increase interpretability.

In the long term, expanding the framework to larger, more diverse tasks could reveal conserved symbolic patterns across policies. This might help to develop more advanced interpretable DRL systems. Integrating symbolic models with deep learning techniques may also scale this approach for real-world applications, enabling both improved interpretability and simplified learning in complex environments.

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

Miles D. Cranmer. Interpretable machine learning for science with pysr and symbolicregression.jl. *CoRR*, abs/2305.01582, 2023. doi: 10.48550/ARXIV.2305.01582. URL `https://doi.org/10.48550/arXiv.2305.01582`.

Hosein Hasanbeig, Natasha Yogananda Jeppu, Alessandro Abate, Tom Melham, and Daniel Kroening. Symbolic task inference in deep reinforcement learning. *J. Artif. Intell. Res.*, 80:1099–1137, 2024. doi: 10.1613/JAIR.1.14063. URL `https://doi.org/10.1613/jair.1.14063`.

Thomas Hickling, Abdelhafid Zenati, Nabil Aouf, and Phillippa Spencer. Explainability in deep reinforcement learning: A review into current methods and applications. *ACM Comput. Surv.*, 56 (5):125:1–125:35, 2024. doi: 10.1145/3623377. URL `https://doi.org/10.1145/3623377`.

Samuel Holt, Zhaozhi Qian, and Mihaela van der Schaar. Deep generative symbolic regression. *CoRR*, abs/2401.00282, 2024. doi: 10.48550/ARXIV.2401.00282. URL `https://doi.org/10.48550/arXiv.2401.00282`.

Mikel Landajuela, Brenden K. Petersen, Sookyung Kim, Cláudio P. Santiago, Ruben Glatt, T. Nathan Mundhenk, Jacob F. Pettit, and Daniel M. Faissol. Discovering symbolic policies with deep reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5979–5989. PMLR, 2021. URL `http://proceedings.mlr.press/v139/landajuela21a.html`.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, 22:268:1–268:8, 2021. URL `https://jmlr.org/papers/v22/20-1364.html`.

Andrew Silva, Matthew C. Gombolay, Taylor W. Killian, Ivan Dario Jimenez Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 1855–1865. PMLR, 2020. URL `http://proceedings.mlr.press/v108/silva20a.html`.

Supplemental material

Algorithm 2 DRL Symbolic Policy Retrieval

```
Require: DRL algorithm $A$, environment $env$, training steps $t$, sample number $m$, sampling probability $p$, complexity levels $C$, utility function $u$, evaluation episodes $n$
1: Initialize agent for policy $\pi_{DRL}$
2: for $i \in 1..t$ do
3:     Train $\pi_{DRL}$ in $env$ using $A$
4: end for
5: Compute total reward $R_c$ over $k$ episodes
6: Distill symbolic policy $\pi^*_{SP}$ from $\pi_{DRL}$ using $env$, $m$, $p$, $C$, $u$, $n$
7: return $\pi^*_{SP}$ with optimal tradeoff
```

Checklist

1. General points:
   (a) Do the main claims made in the abstract and introduction accurately reflect your contributions and scope? [Yes]
   (b) Did you cite all relevant related work? [Yes]
   (c) Did you describe the limitations of your work? [Yes]
   (d) Did you include a discussion of future work? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you run at least 5 repetitions of your method? [Yes]
   (d) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (e) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you make sure the license of the assets permits usage? [Yes]
   (c) Did you reference the assets directly within your code and repository? [Yes]