Evaluating Interpretable Reinforcement Learning by Distilling Policies into Programs

Hector Kohler, Quentin Delfosse, Waris Radji, Riad Akrour, Philippe Preux

Keywords: Interpretability, Evaluation, Benchmark, Baselines, Distillation, Imitation

Summary

There exist applications of reinforcement learning like medicine where policies need to be "interpretable" by humans. User studies have shown that some policy classes might be more interpretable than others. However, it is costly to conduct human studies of policy interpretability. Furthermore, there is no clear definition of policy interpretabiliy, i.e., no clear metrics for interpretability and thus claims depend on the chosen definition. We tackle the problem of empirically evaluating policies interpretability without humans. To advance research in interpretable reinforcement learning, we contribute a new methodology to evaluate policy interpretability. We distillate neural networks policies into small `Python` programs that we use as baselines. We use our methodology to conduct a large-scale empirical evaluation of policy interpretability and study how our empirical conclusions relate to previous work.

Contribution(s)

1. We provide a sound methodology to evaluate the interpretability of policies for Markov decision processes without human studies.
   Context: There is no methodology to study interpretability without humans that both, supports results that interpretability depends on the policy class and on the number of policy parameters (Freitas, 2014; Freitas et al., 2010; Martens et al., 2011; Verbeke et al., 2011; Lavrač, 1999), and, claims that policy interpretability depends on the chosen metrics (Lipton, 2018). Hence providing such a methodology is an open problem in interpretable reinforcement learning (Glanois et al., 2024).
2. Using our methodology, we show that, across the tested tasks, and across tested interpretability metrics, there is no policy class that is better in terms of cumulative reward, or better in terms of interpretability, or better in trading off reward and interpretability.
   Context: Human studies have shown that for particular users and particular tasks, trees are considered more interpretable than policies involving equations (Freitas et al., 2010; Martens et al., 2011; Verbeke et al., 2011). However, Lipton (2018) argues that this claim depends on the choice of interpretability definition. Since there is no definite interpretability ranking of policy classes, having a methodology to compare the interpretability-performance trade-offs of policies of different classes is crucial.
3. We show that to properly evaluate policy interpretability it is necessary to "unfold" policies into a similar language such as Python to standardize the execution of policy inference.
   Context: Previous works have attempted to compare policies interpretability without grounding them into a common language leading to results that depend on hardware and software choices (Bastani et al., 2018; Luo et al., 2024).
4. We open source the interpretable policies that we use to validate our methodology: `https://github.com/KohlerHECTOR/interpretable-rl-zoo`.
   Context: Similarly to open source baselines for reinforcement learning (Raffin et al., 2021; Raffin, 2020), we hope to advance research in interpretable reinforcement learning.

Evaluating Interpretable Reinforcement Learning by Distilling Policies into Programs

Hector Kohler[1], **Quentin Delfosse**[2], **Waris Radji**[1], **Riad Akrour**[1], **Philippe Preux**[1]

hector.kohler@inria.fr, quentin.delfosse@cs.tu-darmstadt.de, {waris.radji,riad.akrour,philippe.preux}@inria.fr

[1]**Université de Lille, Inria, CRISTaL UMR 9198, CNRS, France**
[2] **AI & ML Lab, Computer Science Department, TU Darmstadt, Germany**

Abstract

There exist applications of reinforcement learning like medicine where policies need to be "interpretable" by humans. User studies have shown that some policy classes might be more interpretable than others. However, it is costly to conduct human studies of policy interpretability. Furthermore, there is no clear definition of policy interpretabiliy, i.e., no clear metrics for interpretability and thus claims depend on the chosen definition. We tackle the problem of empirically evaluating policies interpretability without humans. Despite this lack of clear definition, researchers agree on the notions of "*simulatability*": policy interpretability should relate to how humans understand policy actions given states. To advance research in interpretable reinforcement learning, we contribute a new methodology to evaluate policy interpretability. We distillate expert neural networks policies into small programs that we use as baselines. We then show that using our methodology to evaluate the baselines interpretability leads to similar conclusions as user studies. Most importantly, we show that there is no policy class that better trades off interpretability and performance across tasks.

1 Introduction

There is increasing research in developing reinforcement learning algorithms that return "interpretable" policies such as trees, programs, or linear maps (Bastani et al., 2018; Verma et al., 2018; Mania et al., 2018; Delfosse et al., 2023; Milani et al., 2024; Glanois et al., 2024; Kohler et al., 2024). Indeed, interpretability has been useful for different applications: policy verification (Bastani et al., 2018), mis-alignment detection (Delfosse et al., 2024b; Marton et al., 2025) and features importance analysis (Wabartha & Pineau, 2024; Alver & Precup, 2024; Acero & Li, 2024).

User studies have established the common beliefs that decision trees are more "interpretable" than linear maps, oblique trees (trees where nodes are tests of linear combinations of features), and multilayer perceptrons (MLPs) (Freitas, 2014; Freitas et al., 2010; Martens et al., 2011; Verbeke et al., 2011). Furthermore, for a fixed class of models, humans give different values of interpretability to models with different numbers of parameters (Lavrač, 1999). However, survey works argue that every belief about interpretability needs to be verified with user studies and that interpretability evaluations are grounded to a specific set of users, to a specific application, and to a specific definition of interpretability (Doshi-Velez & Kim, 2017; Lipton, 2018). For example, Lipton (2018) claims that depending on the notion of *simulatability* studied, MLPs can be more interpretable than trees, since deep trees can be harder for a human to read than compact MLPs. Hence, even with access to users it would be difficult to research interpretability. More realistically, since the cost of user studies is high (time, variety of subjects required, ethics, etc.), designing proxies for interpretability in

machine learning has become an important open problem in both supervised (Doshi-Velez & Kim, 2017) and reinforcement learning (Glanois et al., 2024).

In this work, we propose a methodology to evaluate the interpretability of reinforcement learning without human evaluators, by measuring inference times and memory consumptions of policies as programs. We show that those measures constitute adequate proxies for the notions of "*simulatability*" described in Lipton (2018), which relates the interpretability of policy to humans ability to understand the inference of actions given states. In addition to the contributions summarized next, we open source some of the interpretable baselines to be be used for research and teaching[1].

In Section 3.2, we distill deep neural network experts into more compact programs that can be used as baselines for interpretability evaluation. In Section 4, we validate the proposed methodology and show that it is necessary to ground policies into a common language to obtain meaningful interpretability measures. In Section 4.2, we show how our methodology can be used to tackle interpretable reinforcement learning problems : I) Is it possible to obtain a fully interpretable policy for MDPs with large state-action spaces? II) Do policies have to trade off rewards for interpretability? III) Can interpretable policies be formally verified efficiently?

2 Methodology Overview

In this section, we explain our methodology for the evaluation of interpretability. Our approach consists of three main steps: (1) obtaining deep neural network policies trained with reinforcement learning that obtain high cumulative rewards, (2) distilling those policies into less complex ones to use as baselines (3) after parsing baselines from different classes into a common comparable language, we evaluate the interpretability of the policies using proxy metrics for *simulatability*.

Deep Neural Network Policies In reinforcement learning, an agent learns how to behave in an environment to maximize a cumulative reward signal (Sutton & Barto, 1998). The environment is defined by a Markov decision process (MDP) $M = (\mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state-transition function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function. At each time step t, an agent observes the current state s_t and chooses an action a_t according to its policy π. The agent executes a_t, receives reward r_t, and observes the next state s'_{t+1}. The goal is to find an optimal policy π^* that maximizes the expected discounted future return: $\pi^* = \arg\max_\pi Q^\pi(s, a) = \arg\max \mathbb{E}[r + \gamma Q^\pi(s', a)]$, with γ a discount factor in $[0, 1)$. For large or continuous state spaces like the MDPs we consider in this work, MLPs are used to represent Q^π or π. While these MLPs can be trained efficiently to obtain high cumulative rewards (Schulman et al., 2017; Mnih et al., 2015), they are too complex for interpretability considerations.

Distilling into Interpretable Policies To obtain interpretable policies, we distill the complex neural networks into simpler models using imitation learning, as described in Algorithm 1. This approach transforms the reinforcement learning task into a sequence of supervised learning problems.

Algorithm 1 inputs an environment, that simulates taking steps in an MDP, an expert policy to imitate, also called a teacher, and an (interpretable) policy class to fit, also called student. The hyperparameters of Algorithm 1 are: the number of times we fit a student policy, the total number of samples to be collected, and whether or not to use importance sampling. At each iteration of Algorithm 1 the student policy is fitted to a dataset of states collected with the expert at iteration 1 or with the previously fitted student (see Line 5). The actions are always given by the expert (see Line 9). When using importance sampling, the states are further re-weighted by the worst state-action value possible in the given state. When the number of iteration is 1, Algorithm 1 is behavior cloning (Pomerleau, 1988). When we use importance sampling, Algorithm 1 is Q-DAgger (Bastani et al., 2018). In other cases, Algorithm 1 is DAgger (Ross et al., 2010).

[1] https://github.com/KohlerHECTOR/interpretable-rl-zoo

Algorithm 1: Imitate Expert (Pomerleau, 1988; Ross et al., 2010; Bastani et al., 2018)

Input: Expert policy π^*, MDP M, policy class Π, number of iterations N, total samples S, importance sampling flag I

Output: Fitted student policy $\hat{\pi}_i$

Initialize dataset $\mathcal{D} \leftarrow \emptyset$;
Initialize $\hat{\pi}_1$ arbitrarily from Π;
for $i \leftarrow 1$ **to** N **do**
  if *i = 1* **then** $\pi_i \leftarrow \pi^*$;
  else $\pi_i \leftarrow \hat{\pi}_i$;
  Sample S/N transitions from M using π_i;
  if I *is True* **then** $w(s) \leftarrow V^{\pi *}(s) - \min_a Q^{\pi *}(s, a)$;
  else $w(s) \leftarrow 1$;
  Collect dataset $\mathcal{D}_i \leftarrow \{(s, \pi^*(s), w(s))\}$ of states visited by π_i and expert actions;
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$;
  Fit classifier/regressor $\hat{\pi}_{i+1}$ on $\mathcal{D}$;
return $\hat{\pi}_N$;

Measuring Policy Interpretability After obtaining interpretable policy baselines using Algorithm 1, we use two metrics to evaluate policy interpretability without requiring human intervention. Those metrics are proxies for the notion of *simulatability* from Lipton (2018) that gives insights on how a human being would read a policy to understand how actions are inferred. In particular, *simulatability* admits two sub-definitions. The first one is a measure of how difficult it is for a human to reproduce the computations of the policy to infer actions given states. The second one measures how difficult it is for a human to read through the entire policy. Lipton (2018) argues that this nuance is key when measuring interpretability because a tree is not read entirely to compute a single action and because there is no consensus on what is easier for a human to read between an MLP and a tree.

1. Policy Inference Time: to measure how a human would compute the action of a policy given a state at each environment step, we measure policy step inference time in seconds.

2. Policy Size: to measure how easily a human can read the entire policy, we measure its size in bytes. While this correlates with inference time for MLPs and linear models, tree-based policies may have large sizes but quick inference because they do not traverse all decision paths at each step.

As these measurements depend on many technical details (programming language, the compiler if any, the operating system, versions of libraries, the hardware it is executed on, etc), to ensure fair comparisons, we translate all student policies into a simple representation that mimics how a human being "reads" a policy. We call this process of standardizing policies language “unfolding”. In Figure 1, 2, and 3, we present some unfolded policy programs. Other works have distilled neural networks into programs (Verma et al., 2018) or even directly learn programmatic policies (Qiu & Zhu, 2022) from scratch. However, those works directly consider programs as a policy class and could compare a generic program (not unfolded, with, e.g., while loops or array operations) to, e.g, a decision tree (Trivedi et al., 2021). We will discuss later on the limitations of unfolding policies.

3 Computing Baseline Policies

3.1 Setup

All the experiments presented next run on a dedicated cluster of Intel Xeon Gold 6130 (Skylake-SP), 2.10GHz, 2 CPUs/node, 16 cores/CPU with a timeout of 4 hours per experiment. Codes to reproduce our results are given in the supplementary material. In the future, we will open source a python library with all the tools of our methodology. Using Algorithm 1, we distill deep neural network expert policies into less complex policy classes.

```python
import gymnasium as gym

env = gym.make("MountainCar")
s, _ = env.reset()
done = False
while not done:
    y0 = 0.969*s[0]-30.830*s[1]+0.575
    y1 = -0.205*s[0]+22.592*s[1]-0.63
    y2 = -0.763*s[0]+8.237*s[1]+0.054
    max_val = y0
    action = 0
    if y1 > max_val:
        max_val = y1
        action = 1
    if y2 > max_val:
        action = 2
    s, r, terminated, truncated, \
    infos = env.step(action)
    done = terminated or truncated
```

Figure 1: Unfolded linear policy interacting with an environment.

```python
def play(x):
    h_layer_0_0 = 1.238*x[0]+0.971*x[1
                  +0.430*x[2]+0.933
    h_layer_0_0 = max(0, h_layer_0_0)
    h_layer_0_1 = -1.221*x[0]+1.001
                  *x[1]-0.423*x[2]
                  +0.475
    h_layer_0_1 = max(0, h_layer_0_1)
    h_layer_1_0 = -0.109*h_layer_0_0
                  -0.377*h_layer_0_1
                  +1.694
    h_layer_1_0 = max(0, h_layer_1_0)
    h_layer_1_1 = -3.024*h_layer_0_0
                  -1.421*h_layer_0_1
                  +1.530
    h_layer_1_1 = max(0, h_layer_1_1)

    h_layer_2_0 = -1.790*h_layer_1_0
                  +2.840*h_layer_1_1
                  +0.658
    y_0 = h_layer_2_0
    return [y_0]
```

Figure 2: Tiny ReLU MLP for Pendulum.

```python
def play(x):
    if (x[3] - x[4]) <= -0.152:
        if (x[2] - x[3]) <= 0.049:
            return 3
        else:
            if x[5] <= 0.030:
                return 2
            else:
                if (x[2] - x[4]) <= 0.061:
                    return 3
                else:
                    #Fire main engine
                    return 2
    else:
        if (x[0] - x[6]) <= -0.840:
            return 0
        else:
            if (x[0] - x[4]) <= -0.007:
                # Fire right engine
                return 3
            else:
                # Fire left engine
                return 1
```

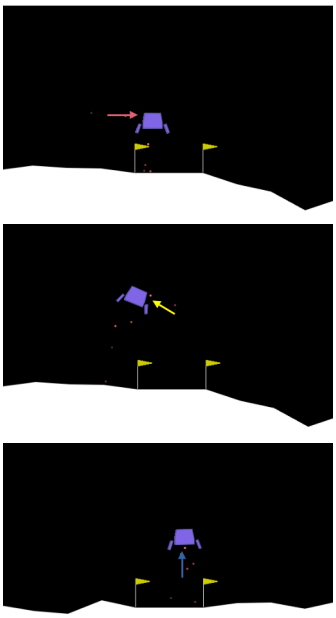


Figure 3: An unfolded oblique tree policy's actions obtaining 250 rewards on Lunar Lander.

Policy Class	Parameters	Training Algorithm
Linear Policies	Determined by state-action dimensions	Linear/Logistic Regression
Decision Trees	[4, 8, 16, 64, 128] nodes	CART (2× nodes maximum leaves)
Oblique Decision Trees	[4, 8, 16, 64, 128] nodes	CART (2× nodes maximum leaves)
ReLU MLPs	[2×2, 4×4, 8×8, 16×16] weigths	Adam optimization (500 iterations)

Table 1: Summary of baseline policy classes parameters and fitting algorithms (used in Line 11).

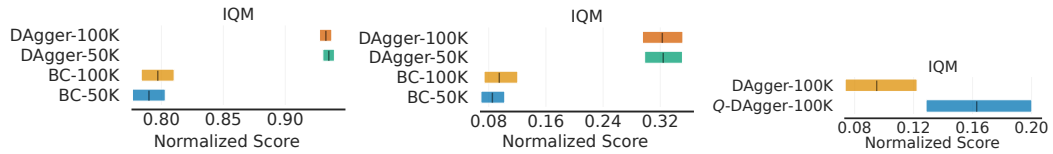
Policy classes We consider four policy classes for our baselines. We choose those policy classes because there exist efficient algorithms to fit them with supervised data which is a required step of imitation learning in Line 11. We consider linear policies that have been shown to be able to solve Mujoco tasks (Mania et al., 2018). We fit linear policies to expert policies using simple linear (logistic) regressions with scikit-learn (Pedregosa et al., 2011) default implementation. We also consider decision trees (Breiman et al., 1984) and oblique decision trees (Murthy et al., 1994). (Oblique) Decision trees are often considered the most interpretable model class in machine learning (Lipton, 2018) and reinforcement learning (Bastani et al., 2018; Topin et al., 2021; Glanois et al., 2024; Milani et al., 2024). We train trees using the default CART (Breiman et al., 1984) implementation of scikit-learn with varying numbers of parameters (number of nodes in the tree). We also consider MLPs with ReLU activations (He et al., 2015) with varying number of parameters (total number of weights). This class of policy is often considered the least interpretable and is often used in deep reinforcement learning (Haarnoja et al., 2018; Chen et al., 2021; Hiraoka et al., 2022). We train ReLU MLPs using the default scikit-learn implementation of Adam optimization (Kingma & Ba, 2017) with 500 iterations. The 15 baseline policy classes that we consider are summarized in Appendix 1.

Neural network experts We do not train new deep reinforcement learning agents (Mnih et al., 2015; Schulman et al., 2017; Haarnoja et al., 2018) but rather re-use ones available at the stable-baselines3 zoo (Raffin, 2020). Depending on the environments described next, we choose neural network policies from different deep reinforcement learning agents. Some may argue that during the imitation learning, ReLU MLPs baselines may obtain better performance because they are often from the same class as the expert they imitate unlike trees. But this is not of our concern as we do not benchmark the imitation learning algorithms. Furthermore, it is important to note that not all experts are compatible with all the variants of imitation learning Algorithm 1. Indeed, SAC experts (Haarnoja et al., 2018) are not compatible with Q-DAgger (Bastani et al., 2018) because it only works for continuous actions; and PPO experts, despite working with discrete actions do not compute a Q-function necessary for the re-weighting in Q-DAgger.

Environments We consider common environments in reinforcement learning research. We consider the classic control tasks from gymnasium (Towers et al., 2024), MuJoCo robots from Todorov et al. (2012), and Atari games from Bellemare et al. (2013). For Atari games, since the state space is frame pixels that can't be interpreted, we use the object-centric version of the games from Delfosse et al. (2024a). In Appendix 3 we give the list of environments we consider in our experiments with their state-action spaces as well as a cumulative reward threshold past which an environment is consider "solved".

3.2 Ablation study of imitation learning

In this section, we present the results of the expert distillation into smaller policies. For each environment, we fit all the policy classes. To do so, we run different instantiations of Algorithm 1 multiple times with different total sample sizes. For each environment and each imitation learning variant, we summarize the number of times we fit all the baselines to an expert and which expert we use. The number of runs and imitation algorithm variants of Algorithm 1 are summarized in Appendix 4. After running the imitation learnings, we obtain roughly 40000 baseline policies (35000 for classic control, 5000 thousands for MuJoCo and 400 for OCAtari). A dataset with all the baselines measurements is given in the supplementary material.



(a) Classic control, PPO expert (b) MuJoCo, SAC expert (c) OCAtari, DQN expert

Figure 4: Performance of imitation learning variants of Algorithm 1 on different environments. We plot the 95% stratified bootstrapped confidence intervals around the IQMs.

What is the best imitation algorithm? Even though the focus of our work is to evaluate trained policies, we still provide some insights on the best way to obtain interpretable policies from experts. Using the reinforcement learning evaluation library rliable (Agarwal et al., 2021), we plot on Figure 4 the interquartile means (IQM, an estimator of the mean robust to outliers) of the baseline policies cumulative rewards averaged over 100 episodes. For each imitation algorithm variant, we aggregate cumulative rewards over environments and policy classes. We normalize the baselines cumulative rewards between expert and random agent cumulative rewards.

The key observation is that for tested environments (Figures 4a,4b), Behavior Cloning is not an efficient way to train baseline policies compared to DAgger. This is probably because Behavior Cloning trains a student policy to match the expert's actions on states visited by the expert while DAgger trains a student to take the expert's actions on the states visited by the student (Ross et al., 2010). An other observation is that the best performing imitation algorithms for MuJoCo (DAgger, Figure 4b) and OCAtari (Q-Dagger, Figure 4c) obtain baselines that in average cannot match well the performances of the experts. However baseline policies almost always match the expert on simple tasks like classic control (Figure 4a).

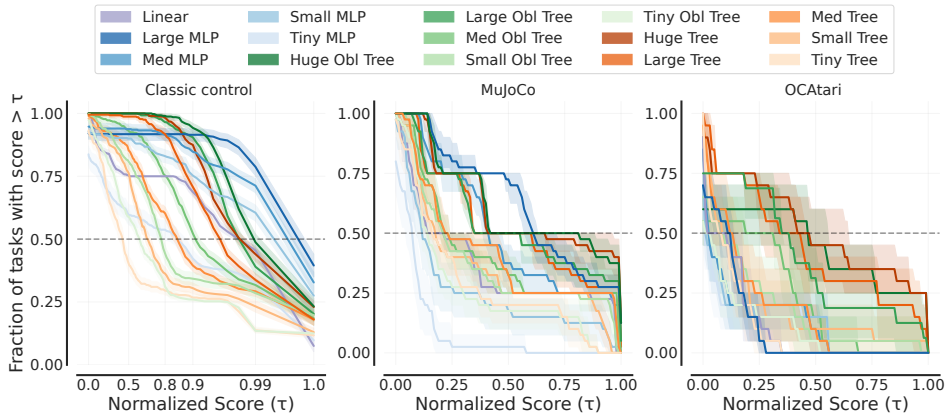


Figure 5: Performance profiles of different policy classes on different environments.

What is the best policy class in terms of reward? We also wonder if there is a policy class that matches expert performances more often than others across environments. For that we plot performance profiles of the different policy classes obtained with a fixed expert and fixed imitation learning algorithm. In particular, for each environments group we use the baseline policies obtained from the best performing imitation learning algorithm from Figure 4. From Figure 5 we see that on classic control environments, MLPs tend to perform better than other classes while on OCAtari games, trees tend to perform better than other classes. Now we move on to interpretability evaluation of our programmatic policies.

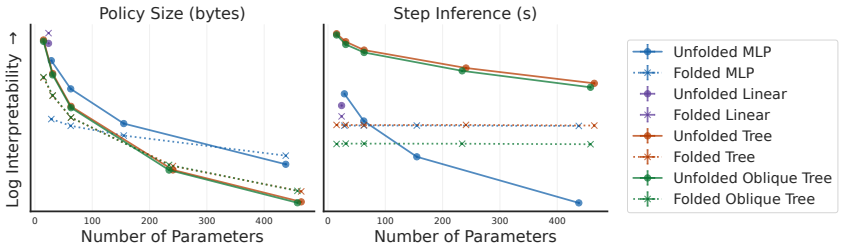


Figure 6: Policies interpretability with different proxies on classic control environments. We plot 95% stratified bootstrapped confidence intervals around means in both axes.

4 Measuring Policy Interpretability

4.1 From Policy to Program

In this section, we compute the step inference times, as well as the policy size for both the folded and unfolded variant of each policy obtained for classic control environments with DAgger-100K. To unfold policies, we convert them into Python programs formatted with PEP 8 (comparing other unfolding formats such as ONNX `https://github.com/onnx/onnx` is left to future work). We ensure that all policies operations are performed sequentially and compute the metrics for each policy on 100 episodes using the same CPUs.

Is it necessary to unfold policies to compute interpretability metrics? We see on Figure 6 that folded policies of the same class almost always give similar interpretability values (dotted lines) despite having very different number of parameters. Hence, measuring folded policies interpretability would contradict established results from user studies such as, e.g., trees of different sizes have different levels of interpretability (Lavrač, 1999).

Is there a best policy class in terms of interpretability? User studies from Freitas et al. (2010); Martens et al. (2011); Verbeke et al. (2011) show that decision trees are easier to understand than models involving mathematical equations like oblique trees, linear maps, and MLPs. However, Lipton (2018) states that for a human wanting to have a global idea of the inference of a policy, a compact MLP can be more interpretable than a very deep decision tree. In Figure 6, we show that inference speed and memory size of programs help us capture those nuances: policy interpretability does not only depend on the policy class but also on the metric choice. Indeed, when we measure interpretability with inference times, we do observe that trees are more interpretable than MLPs. However, when measuring interpretability with policy size, we observe that MLPs can be more interpretable than trees for similar number of parameters. Because there seem to not be a more interpretable policy class across proxy metrics, we will keep studying both metrics at the same time.

4.2 Interpretability-performance trade-offs

Now that we trained baseline policies and validated the proposed methodology, we use the latter to tackle open problems in interpretable reinforcement learning. For each environment, we fix the imitation learning algorithm and save the best baseline policy of each class in terms of episodic rewards after unfolding them. Each single Python policy is then **run again on the same dedicated CPU** for 100 new environment episodes (similarly to choosing a classifier with validation score and reporting the test score in the context of supervised learning).

Is it possible to compute interpretable policies for high-dimensional environments? Glanois et al. (2024) claim that computing an interpretable policy for high dimensional MDPs is difficult since it is similar to program synthesis which is known to be NP-hard (Gulwani et al., 2017). Using our measures of interpretability, we can corroborate this claim. On Figure 7, we can indeed observe

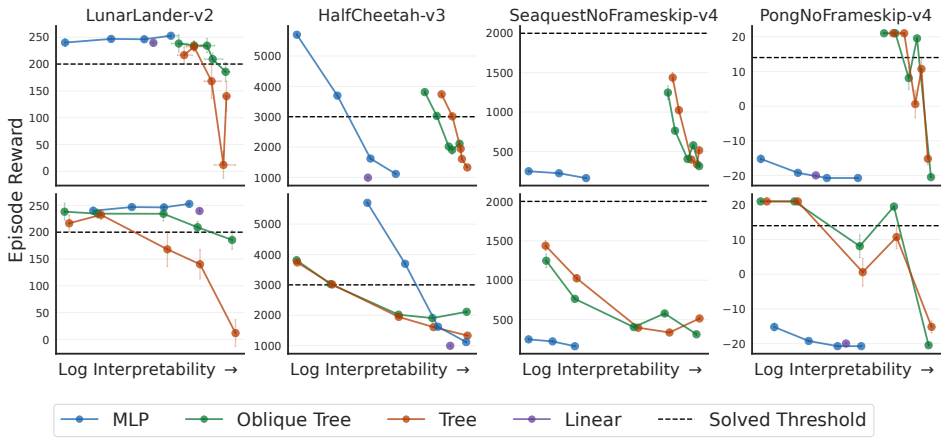


Figure 7: Interpretabality-Performance trade-offs. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size. We plot 95% bootstrapped confidence intervals around means on both axes.

Environment Attributes	Importance for Step inference	Importance for Policy size
States dimension	**80.87**	**35.52**
Expert episodes lengths	11.39	9.28
Episode reward of random	2.26	4.75
Expert episode reward	1.51	16.80
Episode reward to solve	1.41	14.26
Actions dimension	1.41	2.02
Expert reward - Solve reward	1.15	17.37

Table 2: Environment attributes importance to predict interpretability using either of our metrics.

that some relatively interpretable policies can solve Pong (20 state dimensions) or HalfCheetah (17 state dimensions) while for very high-dimensional environments like Seaquest (180 state dimensions), no baseline can solve the game.

For what environment are there good interpretable policies? We fitted a random forest regressor (Breiman, 2001) to predict the interpretability values of our baseline policies using environment attributes. In Table 2 we report the importance of each environment attribute when it comes to accurately predicting interpretability scores. We show that as hinted previously, the states dimensions of the environment is determining to predict the interpretability of good policies. Unsurprisingly, expert attributes also influence interpretability: for the environments where there is a positive large gap between expert and threshold rewards, the task could be considered easy and vice-versa.

How does interpretability influence performance? Mania et al. (2018); Mansour et al. (2022) show the existence of linear and tree policies respectively that solve MuJoCo and continuous maze environments respectively; essentially showing that there exist environments for which policies more interpretable than deep neural networks can still compete performance-wise. Our evaluation indeed shows the existence of such environments. On Figure 7 we observe that on, e.g., LunarLander, increasing policy interpretability up to a certain point does not decrease reward. Actually, we can observe that for Pong a minimum level of interpretability is required to solve the game. Indeed, as stated in (Freitas, 2014), optimizing interpretability can also be seen as regularizing the policy which can increase generalization capabilities. The key observation is that the policy class achieving the best interpretability-performance trade-off depends on the problem. Indeed, independent of the interpretability proxy metric, we see on Figure 7 that for LunarLander it is an MLP that achieves

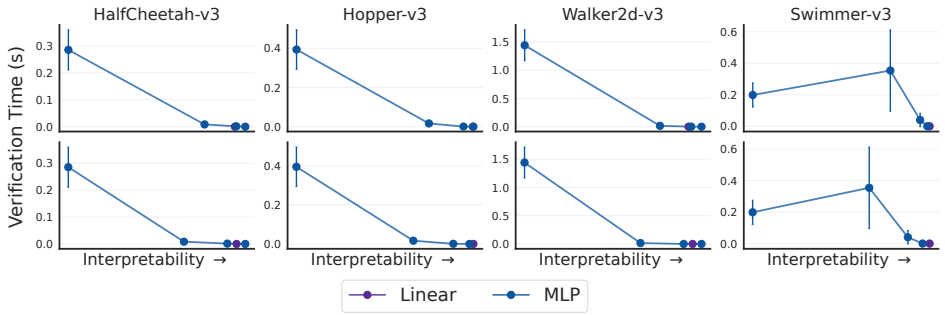


Figure 8: Verification time as a function of policy interpretability. Top row, interpretability is measured with step inference times. Bottom row, the interpretability is measured with policy size. We plot 95% confidence intervals around means on both axes.

the best trade-off while for Pong it is a tree. Next, we compare our proxies for interpretability with another one; the verification time of policies used in Bastani et al. (2018); Barceló et al. (2020).

4.3 Verifying interpretable policies

Barceló et al. (2020) states that the cost of formally verifying properties of MLPs scales exponentially with the number of the parameters. Hence, they propose to measure interpretability of a policy as the computations required to verify properties of actions given state subspaces, what they call local explainability queries (Guidotti et al., 2018). Before Barceló et al. (2020), Bastani et al. (2018) also compared the time to formally verified properties of trees to the time to verify properties of MLPs to evaluate interpretability. In practice, this amounts to passing states and actions bounds and solving the SAT problem of finding a state in the state bounds for which the policy outputs an action in the action bounds. For example, for the LunarLander problem, a query could be to verify if when the y-position of the lander is below some threshold value, i.e, when the lander is close to the ground, there exists a state such that the tested policy would output the action of pushing towards the ground: if the solver outputs "SAT", then there is a risk that the lander crashes.

Designing interesting queries covering all risks is an open problem, hence to evaluate the verification times of our baseline policies, we generate 500 random queries per environment by sampling state and action subspaces uniformly. Out of those queries, we only report the verification times of "UNSAT" queries since to verify that, e.g., the lander does not crash we want the queries mentioned above to be "UNSAT". We also only verify instances of ReLU MLPs and Linear predictors using (Wu et al., 2024) for this experiment as verifying decision trees requires a different software (De Moura & Bjørner, 2008) for which verification times would not be comparable.

On Figure 8, we can observe that verification time decreases exponentially with MLP interpretability, both memory and inference speed, as shown in Barceló et al. (2020). This is another good validation of our proposed methodology as well as a motivation to learn interpretable policies.

5 Conclusions

We have shown that our proposed methodology provides researchers with a sound way of evaluating policy interpretability. In particular, we have shown that unfolding policies in a common language such as Python is a key component of our methodology to ensure that interpretability depends on the policy complexity. Furthermore, we were able to show that policy size in bytes and policy inference speed are good interpretability proxies as they can corroborate findings from user studies. Using the proposed methodology, we were able to illustrate the trade-offs between episodic reward and interpretability of policies from different classes and showed the crucial need of our methodology as there is no better off policy class across tasks and metrics.

References

Fernando Acero and Zhibin Li. Distilling reinforcement learning policies for interpretable robot locomotion: Gradient boosting machines and symbolic regression. In *Workshop on Interpretable Policies in Reinforcement Learning @RLC-2024*, 2024. URL https://openreview.net/forum?id=fa3fjH3dEW.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Safa Alver and Doina Precup. An attentive approach for building partial reasoning agents from pixels. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=S3FUKFMRw8.

Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux. Model interpretability through the lens of computational complexity. *Advances in neural information processing systems*, 2020.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Neural Information Processing Systems*, 2018.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: an evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013. ISSN 1076-9757.

L Breiman, JH Friedman, R Olshen, and CJ Stone. *Classification and Regression Trees*. Wadsworth, 1984.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

Leonardo De Moura and Nikolaj Bjørner. Z3: an efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, pp. 337–340, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3540787992.

Quentin Delfosse, Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. Interpretable and explainable logical policies via neurally guided symbolic abstraction. *Advances in Neural Information Processing (NeurIPS)*, 2023.

Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. OCAtari: Object-centric Atari 2600 reinforcement learning environments. *Reinforcement Learning Journal*, 1:400–449, 2024a.

Quentin Delfosse, Sebastian Sztwiertnia, Mark Rothermel, Wolfgang Stammer, and Kristian Kersting. Interpretable concept bottlenecks to align reinforcement learning agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=ZC0PSk6Mc6.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Alex A. Freitas. Comprehensible classification models: a position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014. ISSN 1931-0145. DOI: 10.1145/2594473.2594475. URL https://doi.org/10.1145/2594473.2594475.

Alex A. Freitas, Daniela C. Wieser, and Rolf Apweiler. On the importance of comprehensible classification models for protein function prediction. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 7(1):172–182, January 2010. ISSN 1545-5963. DOI: 10.1109/TCBB.2008.47. URL https://doi.org/10.1109/TCBB.2008.47.

Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *Machine Learning*, pp. 1–44, 2024.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51 (5), August 2018. ISSN 0360-0300. DOI: 10.1145/3236009. URL https://doi.org/10.1145/3236009.

Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xCVJMsPv3RT.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Hector Kohler, Quentin Delfosse, Riad Akrour, Kristian Kersting, and Philippe Preux. Interpretable and editable programmatic tree policies for reinforcement learning, 2024. URL https://arxiv.org/abs/2405.14956.

Nada Lavrač. Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16(1):3–23, 1999. ISSN 0933-3657. DOI: https://doi.org/10.1016/S0933-3657(98)00062-1. URL https://www.sciencedirect.com/science/article/pii/S0933365798000621. Data Mining Techniques and Applications in Medicine.

Zachary C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, June 2018. ISSN 1542-7730. DOI: 10.1145/3236386.3241340. URL https://doi.org/10.1145/3236386.3241340.

Lirui Luo, Guoxi Zhang, Hongming Xu, Yaodong Yang, Cong Fang, and Qing Li. End-to-end neuro-symbolic reinforcement learning with textual explanations. *International Conference on Machine Learning (ICML)*, 2024.

Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/7634ea65a4e6d9041cfd3f7de18e334a-Paper.pdf.

Yishay Mansour, Michal Moshkovitz, and Cynthia Rudin. There is no accuracy-interpretability tradeoff in reinforcement learning for mazes, 2022. URL https://arxiv.org/abs/2206.04266.

David Martens, Jan Vanthienen, Wouter Verbeke, and Bart Baesens. Performance of classification models from a user perspective. *Decision Support Systems*, 51(4):782–793, 2011. ISSN 0167-9236. DOI: https://doi.org/10.1016/j.dss.2011.01.013. URL https://www.sciencedirect.com/science/article/pii/S016792361100042X. Recent Advances in Data, Text, and Media Mining & Information Issues in Supply Chain and in Service System Design.

Sascha Marton, Tim Grams, Florian Vogt, Stefan Lüdtke, Christian Bartelt, and Heiner Stuckenschmidt. Mitigating information loss in tree-based reinforcement learning via direct optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=qpXctF2aLZ.

Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Comput. Surv.*, 56(7), April 2024. ISSN 0360-0300. DOI: 10.1145/3616864. URL https://doi.org/10.1145/3616864.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Sreerama K Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 1994.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Wenjie Qiu and He Zhu. Programmatic reinforcement learning without oracles. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=6Tk2noBdvxt.

Antonin Raffin. Rl baselines3 zoo, 2020.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: reliable reinforcement learning implementations. *J. Mach. Learn. Res.*, 22(1), January 2021. ISSN 1532-4435.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots andSystems (IROS)*, pp. 5026–5033. IEEE, 2012.

Nicholay Topin, Stephanie Milani, Fei Fang, and Manuela Veloso. Iterative bounding mdps: Learning interpretable policies via non-interpretable methods. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Dweep Trivedi, Jesse Zhang, Shao-Hua Sun, and Joseph J Lim. Learning to synthesize programs as interpretable and generalizable policies. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=wP9twkexC3V.

Wouter Verbeke, David Martens, Christophe Mues, and Bart Baesens. Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, 38(3):2354–2364, 2011. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2010.08.023. URL https://www.sciencedirect.com/science/article/pii/S0957417410008067.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International conference on machine learning*, pp. 5045–5054. PMLR, 2018.

Maxime Wabartha and Joelle Pineau. Piecewise linear parametrization of policies: Towards interpretable deep reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hOMVq57Ce0.

Haoze Wu, Omri Isac, Aleksandar Zeljić, Teruhiro Tagomori, Matthew Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, Pei Huang, Ori Lahav, Min Wu, Min Zhang, Ekaterina Komendantskaya, Guy Katz, and Clark Barrett. Marabou 2.0: A versatile formal analyzer of neural networks, 2024. URL https://arxiv.org/abs/2401.14461.

A Experimental details

In this section we give all the experimental details necessary to reproduce our results.

Classic	**MuJoCo**	**OCAtari**
CartPole (4, 2, **490**)	Swimmer (8, 2, **300**)	Breakout (452, 4, **30**)
LunarLander (8, 4, **200**)	Walker2d (17, 6, **2000**)	Pong (20, 6, **14**)
LunarLanderContinuous (8, 2, **200**)	HalfCheetah (17, 6, **3000**)	SpaceInvaders (188, 6, **680**)
BipedalWalker (24, 4, **250**)	Hopper (11, 3, **2000**)	Seaquest (180, 18, **2000**)
MountainCar (2, 3, **90**)		
MountainCarContinuous (2, 1, **-110**)		
Acrobot (6, 3, **-100**)		
Pendulum (3, 1, **-400**)		

Table 3: Summary of considered environments (dimensions of states and number or dimensions of actions, **reward thresholds**). The rewards thresholds are obtained from gymnasium (Towers et al., 2024). For OCAtari environments, we choose the thresholds as the minimum between the DQN expert from Raffin (2020) and the human scores. We also adapt subjectively some thresholds that we find too restrictive especially for MuJoCo (for example, the PPO expert from Raffin (2020) has 2200 reward on Hopper while the default threshold was 3800).

Envs	BC 50K	BC 100K	DAgger 50K	DAgger 100K	Q 50K	Q-DAgger 100K
Classic	50 (PPO, DQN)	50 (PPO, DQN)	50 (PPO, DQN)	50 (PPO, DQN)	50 (DQN)	50 (DQN)
OCAtari	0	0	0	5 (DQN)	0	5 (DQN)
Mujoco	10 (SAC)	10 (SAC)	10 (SAC)	10 (SAC)	0	0

Table 4: Repetitions of each imitation learning algorithm on each environment. We specify which deep reinforcement learning agent from the zoo (Raffin, 2020) uses as experts in parentheses.

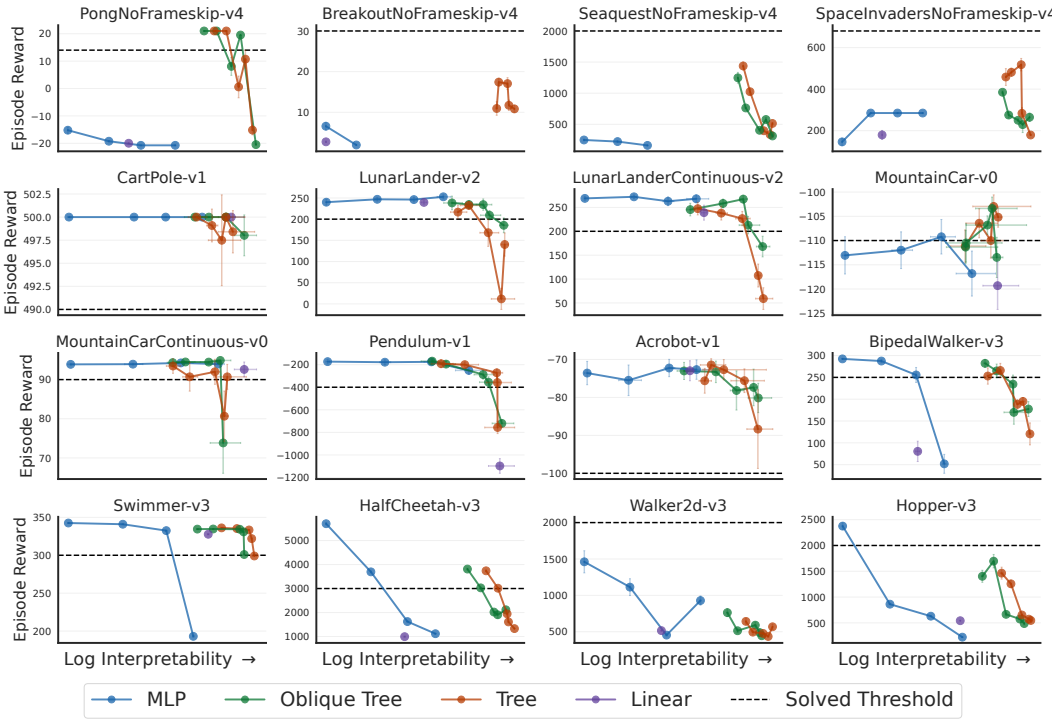


Figure 9: Trade-off Cumulative Reward vs. Step Inference Time

B All interpretability-performance trade-offs

In this appendix we provide the interpretability-performance trade-offs of all the tested environments. All the measures come from the experiment from Section 4.2.

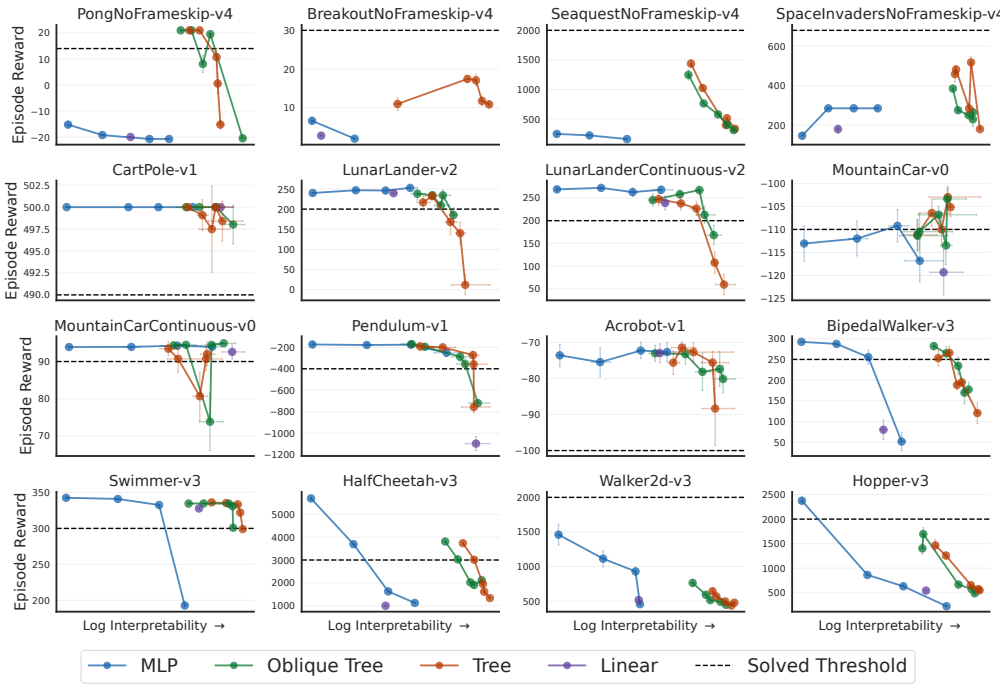


Figure 10: Trade-off Cumulative Reward vs. Episode Inference Time

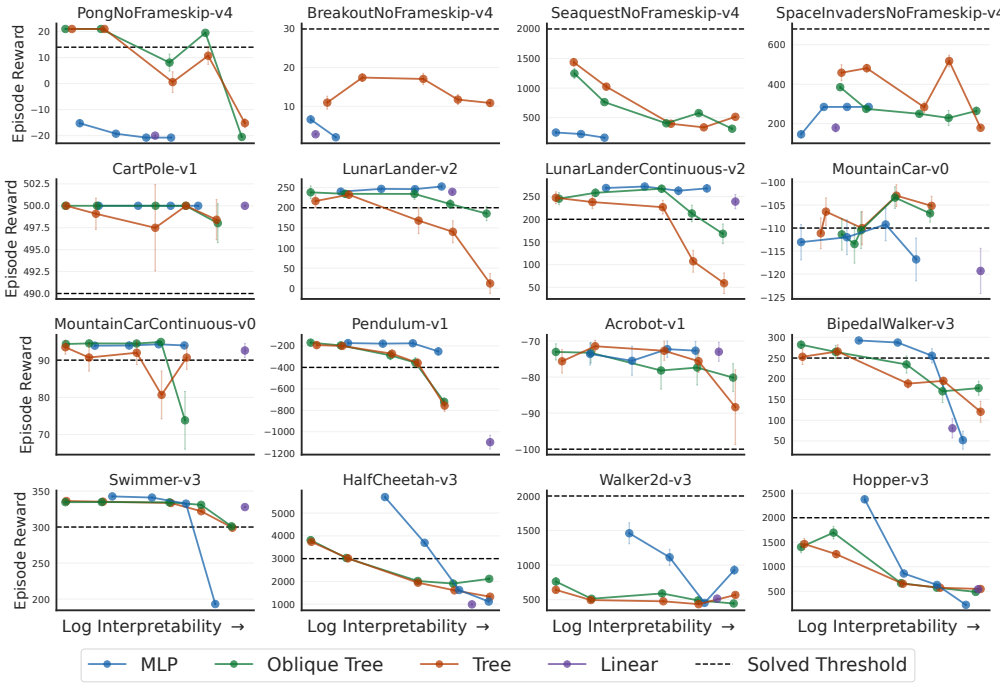


Figure 11: Trade-off Cumulative Reward vs. Policy Size