SMoSE: Sparse Mixture of Shallow Experts for Interpretable Reinforcement Learning in Continuous Control Tasks

Mátyás Vincze[1,2], **Laura Ferrarotti**[2], **Leonardo Lucio Custode**[1]
Bruno Lepri[2], **Giovanni Iacca**[1]
[1] University of Trento, Italy
[2] Fondazione Bruno Kessler, Italy
{mvincze, ferrarotti, lepri}@fbk.eu, {leonardo.custode, giovanni.iacca}@unitn.it

Abstract

Continuous control tasks often involve high-dimensional, dynamic, and non-linear environments. State-of-the-art performance in these tasks is achieved through complex closed-box policies that are effective, but suffer from an inherent opacity. Interpretable policies, while generally underperforming compared to their closed-box counterparts, advantageously facilitate transparent decision-making within automated systems. Hence, their usage is often essential for diagnosing and mitigating errors, supporting ethical and legal accountability, and fostering trust among stakeholders. In this paper, we propose SMoSE, a novel method to train sparsely activated interpretable controllers, based on a *top*-1 Mixture-of-Experts architecture. SMoSE combines a set of interpretable decision-makers, trained to be experts in different basic skills, and an interpretable router that assigns tasks among the experts. The training is carried out via state-of-the-art Reinforcement Learning algorithms, exploiting load-balancing techniques to ensure fair expert usage. We then distill decision trees from the weights of the router, significantly improving the ease of interpretation. We evaluate SMoSE on six benchmark environments from MuJoCo: our method outperforms recent interpretable baselines and narrows the gap with non-interpretable state-of-the-art algorithms.

Introduction

Over the last decade, Artificial Intelligence (AI) has achieved dramatic success. However, the increasing adoption of AI systems in real-world use cases has been raising concerns related to the trustworthiness of these systems (Wexler 2017; McGough 2018; Varshney and Alemzadeh 2017; Rudin, Wang, and Coker 2019; Huang et al. 2020; Smyth et al. 2021; He 2021). One of the most promising paths toward trustworthy AI involves developing methods to enhance our understanding of AI decision-making processes. In this context, eXplainable AI (XAI) introduces AI systems enabling the generation of post-hoc explanations for their behavior (Dwivedi et al. 2023). Although such methods do provide insights into the decision-making processes, their post-hoc analysis explains just an approximation of the original closed-box behavior. This approximation cannot be trusted in high-stakes scenarios, which are frequent for instance in robot control, autonomous driving, emergency response, and public decision-making. Indeed, given that interpretable policies distilled for explanation are typically less complex than closed-box ones, they might require adopting entirely different strategies, disrupting the link between the original behavior and its interpretation (Rudin 2019). To overcome this issue, research efforts have been channeled towards Interpretable AI (IAI) (Rudin 2019; Akrour, Tateo, and Peters 2021). Interpretable AI focuses on systems whose decision-making process is inherently understandable to humans, without the need for distilled explanations (Arrieta et al. 2020). These models prioritize clarity and simplicity, enabling users to directly comprehend how inputs are transformed into outputs, thus facilitating transparency, trust, and validation ease (Rudin 2019).

Due to its sequential nature, Reinforcement Learning (RL) benefits more from directly interpretable policies than post-hoc explanations (Rudin et al. 2021). Discrepancies between a policy and its interpretable approximation can grow over time due to state distribution drift, making local explanations less meaningful (Akrour, Tateo, and Peters 2021). Moreover, post hoc XAI methods, when used in RL to tackle goal misalignment problems, might mislead observers into believing that closed-box agents select the correct actions for the appropriate reasons, even though their decision-making process may actually be misaligned (Chan, Kong, and Liang 2022; Delfosse et al. 2024). This paper addresses the challenge of Interpretable Reinforcement Learning (IRL) in continuous action spaces. In particular, the main advances presented in this work can be summarized as follows:

- We propose SMoSE, a novel, high-performing method, that exploits a sparse, interpretable Mixture-of-Experts (MoE) architecture. Our method simultaneously learns a set of simple yet effective continuous sub-policies, and an interpretable router. The sub-policies are sequentially selected by the router, one per each decision step, to control the system, based on the current state.
- We showcase the capabilities of SMoSE through results obtained on six well-known continuous control benchmarks from the MuJoCo environment (Todorov, Erez, and Tassa 2012). For all the considered environments we analyze performance both in training and in evaluation.
- We include a full interpretation for all the trained poli-

cies, demonstrating the effectiveness of SMoSE in providing extensive insight on the learned controllers. By combining router and the experts interpretation, we analyze the high-level and low-level alignment of the policy.
- We compare SMoSE with interpretable and non-interpretable state-of-the-art RL models, considering both larger model size competitors, and models with comparable size (in terms of the number of active and overall parameters). Results highlight that the proposed architecture, while retaining interpretability, improves the performance w.r.t. other interpretable methods, tightening the gap with non-interpretable approaches.

The paper is structured as follows: the next two sections summarize the relevant background and related contributions, while the Methods section details SMoSE, our proposed approach. Then, the Results section presents the experimental setup and the performance achieved by our method, including the interpretation of the learned policies. Finally, we draw the conclusions and discuss the future directions of this work.

Background

As recently surveyed in (Glanois et al. 2024), several works study IRL models. Some approaches exploit neural logic-based policies to achieve interpretability (Jiang and Luo 2019; Kimura et al. 2021; Delfosse et al. 2023; Sha et al. 2024). Older approaches (McCallum 1996; Pyeatt, Howe et al. 2001) use existing methods for decision tree (DT) induction, adapting them to the RL domain. In (Roth et al. 2019), the authors introduce a heuristic to keep the size of the trees smaller while still achieving good performance. However, these algorithms suffer from the curse of dimensionality, i.e., they do not scale well with the dimensionality of the state space. More recent approaches address this issue. In (Silva et al. 2020), the authors employ soft DTs (Irsoy, Yıldız, and Alpaydın 2012) as policies for RL agents. This simplifies training and allows the use of widely known deep RL algorithms. However, soft trees are difficult to interpret, and discretizing them into "hard" DTs, the policies obtained can suffer from a significant loss in performance.

Other approaches make use of evolutionary principles to optimize DTs. (Dhebar et al. 2020) propose a method for learning a non-linear DT from a neural network trained on the target environment. This allows for choosing the desired properties of the resulting DT (e.g., the depth). On the other hand, this methodology hinders online learning (and thus adaptation to novel scenarios). In (Custode and Iacca 2023), the authors combine evolutionary techniques with Q-Learning to produce DTs that can learn online, while still being interpretable. The DTs produced by this approach achieve performance comparable to non-interpretable state-of-the-art algorithms in a number of simple benchmarks. However, this approach has an extremely high computational cost, requiring a large number of interactions with the environment. Finally, in (Custode and Iacca 2024), the authors leverage principles from social learning to significantly improve both the computational complexity and the performance of evolutionary methods.

Related Work

IRL methods tailored to environments with continuous action space are heavily needed in a wide variety of real-world scenarios, e.g., robot manipulation and control, as showcased by many benchmarking examples (Todorov, Erez, and Tassa 2012). So far, however, only a few works have investigated the use of IRL for continuous control. A branch of research is dedicated to the learning of interpretable programs as RL policies (Verma et al. 2019, 2021; Liu et al. 2023; Kohler et al. 2024). In (Custode and Iacca 2021) the authors propose a cooperative co-evolutionary approach in order to independently evolve a population of binary DTs (generated via Grammatical Evolution) and a population of sets of actions, both optimized w.r.t. the fitness associated with the combined use of the two. (Videau et al. 2022) explore methods for constructing symbolic RL controllers, utilizing parse trees and Linear Genetic Programming (LGP) to represent the programs as a vector of integers. Additionally, a multi-armed bandit strategy distributes the computational budget across generations. LGP is also used in (Nadizar, Medvet, and Wilson 2024), along with Cartesian Genetic Programming (CGP), where programs are instead represented as directed acyclic graphs. In (Paleja et al. 2023), an IRL algorithm for continuous control is introduced, exploiting fuzzy DTs combined with nodes and leaves with differentiable crispification, that can hence be directly learned via gradient descent. As previously mentioned, our method takes advantage of an interpretable MoE architecture for the control policy. MoEs can be found in RL literature, employed to tackle different problems, such as parameter scaling in deep RL (Obando-Ceron et al. 2024), handling of multiple optimal behaviors (Ren et al. 2021), and multi-task learning (Cheng et al. 2023; Willi et al. 2024), to name a few. In the realm of interpretability, a kernel-based method employing MoE is proposed in (Akrour, Tateo, and Peters 2021). In this work, the selection of a set of centroids from trajectory data is optimized. Each state is associated with an expert policy modeled as a Gaussian distribution around a linear policy, while retaining an internal complex function approximator. According to this approach, a learned combination of experts handles the control task. This is obtained by considering fuzzy memberships to clusters and employing a learned set of cluster weights. In our method, instead, we can tune the number of experts employed at every timestep for the decision, and, for instance, force the policy to exploit only one expert, for maximum interpretability (as explained in more detail in the Methods section). Moreover, while the policies in (Akrour, Tateo, and Peters 2021) are updated via approximate policy iteration, the centroids, which must be elements of the replay buffer, require separate iterations of discrete optimization. SMoSE, instead, learns a router that distributes the control tasks among experts. This compact representation, while being fully interpretable, can be learned via backpropagation, simultaneously to the experts.

Method

We seek to solve RL problems with continuous actions structured as Markov Decision Processes (MDPs), i.e., tu-

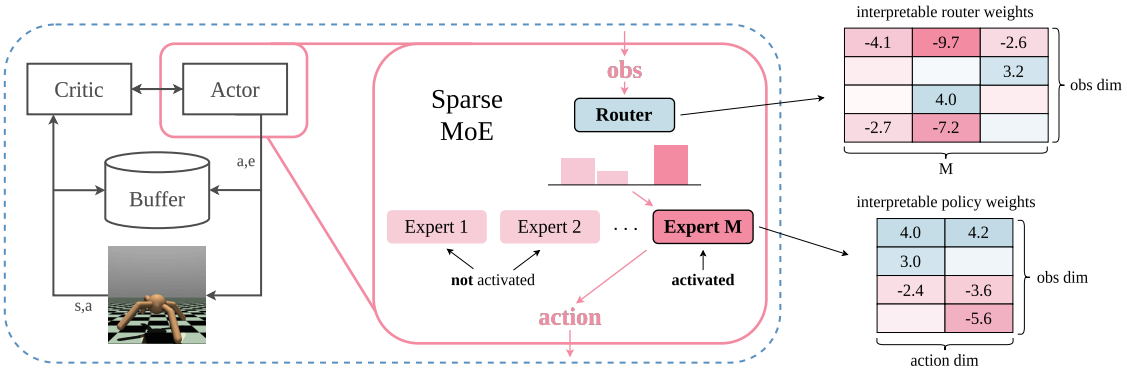

Figure 1: **SMoSE**. Schematic summary of the proposed architecture.

ples $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{S}_0)$, where $\mathcal{S}$ is the set of the states in the problem, $\mathcal{A} \in \mathbb{R}^{n_a}$ is the set of (continuous) actions, $\mathcal{P}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ associates a probability to each transition from (s, a) to each state s'; $\mathcal{R}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ assigns a reward to each triplet (s, a, s'); γ is a discount factor, used for denoting the importance of future rewards w.r.t. the current one, and $\mathcal{S}_0$ is the set of the initial states. Solving such a problem requires the design of a learning strategy to fit a policy function $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, optimizing:

$$\mathbb{E}_{a_t \sim \pi_t}\Big[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t \,\Big|\, s_0 \sim \mathcal{S}_0,\ s_{t+1} \sim \mathcal{P}_t\Big]^1$$

through interaction with the environment. Good results tackling this problem in non-trivial environments are often achieved by complex, not directly interpretable policies.

The main idea behind our method is to decompose a complex behavior by identifying a set of basic skills (or decision strategies) that are easier to interpret when combined with a policy capable of selecting which skill to employ in each state. Decomposing the decision process with this *divide-and-conquer* approach can provide insight into the interpretation of the learned controller's behavior. Thus, we structure the control policy as a composition of:

- A set $\{\pi_m(\cdot \,|\, \theta_m) : \mathcal{S} \to \mathcal{A} \mid m = 1, \dots, M\}$ of M parameterized continuous policy functions, representing M decision-makers, each trained to be expert in an individual useful basic skill. To ensure overall interpretability, we consider continuous policies π_m that are shallow and directly interpretable, such as linear ones.
- A planner, capable of assigning to each state $s \in \mathcal{S}$ an expert policy among the available set, in order to attain optimal behavior. This module is a parameterized router $g(\, s \,|\, \Theta\,) : \mathcal{S} \to [\,0\,,\,1\,]^M$ producing a one-hot encoded M-dimensional vector as output.

SMoSE, summarized in Figure 1, combines the experts and the router according to a MoE architecture of the form:

$$\pi(\, s\,) = \sum_{m=1}^{M} \big[\, g(\, s \,|\, \Theta\,)\,\big]_m \ \pi_m(\, s \,|\, \theta_m\,), \qquad (1)$$

[1] $\pi_t \doteq \pi(s_t, \cdot), \mathcal{R}_t \doteq \mathcal{R}(s_t, a_t, s_{t+1}), \mathcal{P}_t \doteq \mathcal{P}(s_t, a_t, \cdot)$

where $\big[\, g(\, s \,|\, \Theta\,)\,\big]_m$ is the m-th component of vector:

$$g(\, s \,|\, \Theta\,) = \mathrm{TOP}_1(\,\mathrm{softmax}(\,\hat{g}(\, s \,|\, \Theta\,)\,)\,). \qquad (2)$$

In the equation above, we indicate as $\hat{g}(\, s \,|\, \Theta\,) \in \mathbb{R}^M$ the inner parameterization of the router that produces as output an M-dimensional vector. The softmax function then transforms $\hat{g}(\, s \,|\, \Theta\,)$ in a preference vector, where the m-th element measures the preference in choosing the m-th expert as actor in state s. The function TOP_1 assigns value 1 to the vector component with maximum preference, and 0 to all the others, allowing for an extremely sparse MoE structure in which only one expert is activated in each state.

Remark. *The proposed method can also accommodate the use of TOP_k with $k > 1$, replacing TOP_1 in Eq. (2) and thereby enabling the selection of a combination of k experts at each timestep to make control decisions. While the resulting policy would remain interpretable, in this work we intentionally set $k = 1$ to evaluate the performance of a truly sparse MoE architecture and to maximize interpretability within this framework.*

This architecture takes inspiration from (Shazeer et al. 2017; Riquelme et al. 2021), where sparse MoE neural layers are stacked to obtain both computational efficiency in inference and parameters specialized on subsets of states. Here, such architecture is tuned to retain interpretability in continuous control, by removing the non-linearities and activation functions, employing a single-layer shallow structure, and shaping the inner router $\hat{g}$ and the experts π_m as linear functions, that is:

$$\pi_m(\, s \,|\, \theta_m\,) = \theta_m \cdot s \quad \text{and} \quad \hat{g}(\, s \,|\, \Theta\,) = \Theta \cdot s$$

with $\theta_m \in \mathbb{R}^{n_a \times n_s}$ and $\Theta \in \mathbb{R}^{M \times n_s}$.

The control policy in Eq. (1) is trained via RL. The parameters characterizing π_m and g are hence simultaneously learned. It is important to notice that composing the TOP_1 and softmax functions in the order expressed in Eq. (2) permits a larger propagation of the gradients among the router weights, if compared with the opposite ordering. At every update, indeed, each of the M components of $\mathrm{softmax}(\,\hat{g}(\, s \,|\, \Theta\,)\,)$ depends on all the weights Θ, due to the action of softmax. Hence, all the components in Θ will

be updated, not only the ones associated with the expert selected by TOP_1 (Riquelme et al. 2021). We ensure to explore the action space of each expert π_m by injecting stochasticity in the choices in training, i.e.,

$$\pi_m(\, s \,|\, \theta_m, \sigma_m \,) = \mathcal{N}(\, \theta_m \cdot s, \quad \sigma_m^2 \,).$$

Our method can be seamlessly integrated with any RL algorithm for the learning of continuous controllers. In this work, we rely for exploration on Soft Actor-Critic (SAC) (Haarnoja et al. 2018), a state-of-the-art algorithm that balances the objective function with a term promoting higher entropy policies. We structure the actor module according to the interpretable architecture in Eq. (1), while we maintain the classic neural critic introduced in (Haarnoja et al. 2018).

In order to ensure balanced workloads among the M experts, we augment the SAC actor objective function with additional penalties introduced in (Riquelme et al. 2021), weighted by a tunable parameter λ. In particular, per each mini-batch $\mathrm{S} = \{\mathrm{s_k}\} \subseteq \mathcal{S}$ of states, we consider its *importance* for $m = 1, \ldots, M$:

$$\text{Imp}_m(\mathrm{S}) = \sum_{s_\mathrm{k} \in \mathrm{S}} \text{softmax}(\pi_m(\, s_\mathrm{k} \,|\, \theta_m, \sigma_m \,))$$

and we compute the *importance loss*:

$$f_{\text{imp}}(\mathrm{S}) = \frac{1}{2}\Big(\frac{\text{std}(\text{Imp}(\mathrm{S}))}{\text{mean}(\text{Imp}(\mathrm{S}))}\Big)^2. \tag{3}$$

Then, we consider the *load* of S for $m = 1, \ldots, M$:

$$\text{Load}_m(\mathrm{S}) = \sum_{s_\mathrm{k} \in \mathrm{S}} \mathbb{P}(\epsilon_{\text{new}} \geq \tau(s_\mathrm{k}) - \pi_m(\, s_\mathrm{k} \,|\, \theta_m, \sigma_m \,))$$

with $\tau(s_\mathrm{k}) = \max_m \Big(\pi_m(\, s_\mathrm{k} \,|\, \theta_m, \sigma_m \,)\Big)$, and compute the *load-balancing loss*:

$$f_{\text{load}}(\mathrm{S}) = \frac{1}{2}\Big(\frac{\text{std}(\text{Load}(\mathrm{S}))}{\text{mean}(\text{Load}(\mathrm{S}))}\Big)^2. \tag{4}$$

When computing the load-balancing loss on each mini-batch S, noise to the inner router, i.e., $\hat{g}(\, s \,|\, \Theta \,) = \Theta \cdot s + \epsilon$, with $\epsilon \sim \mathcal{N}(\, 0, \, 1/M^2 \,)$, in order to ensure proper exploration of experts employment. Finally, once the policy training is complete, we produce a useful support for interpretation by distilling a multiclass classifier using Decision Trees (DTs), which are trained on a router-labeled replay buffer. By decomposing the original router into M binary classifiers of limited depth, we create an easily readable representation of the interpretable router's decision-making process, where each DT determines whether a specific expert should control the task for a given state. After balancing the data, we train the DTs in a supervised manner, using the CART algorithm (Timofeev 2004). More details on this can be found in the Appendix.

Results

In this section, we present the evaluation of SMoSE on six widely-known continuous-control environments from MuJoCo (Todorov, Erez, and Tassa 2012), namely Walker2d-v4, Hopper-v4, Ant-v4, HalfCheetah-v4, Reacher-v4, and Swimmer-v4 (described in Appendix), which are commonly used as benchmarks in the RL field. The following subsections detail the experimental setup, performance evaluation, and comparative analysis with interpretable and non-interpretable baselines, highlighting the effectiveness of SMoSE in delivering both interpretability and high performance. As it is common in RL literature, performance is measured considering the attained episodic rewards (ER), i.e., the accumulated rewards achieved at each timestep of an episode, in order to compare well with the state of the art.

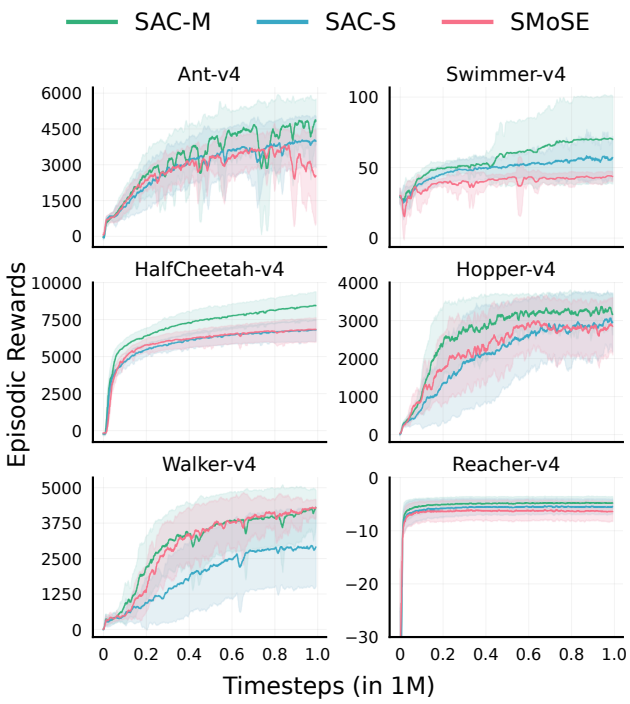

Figure 2: **Performance in training.** SMoSE compares to non-interpretable models of the same size, considering the number of overall (SAC-M) and active (SAC-S) parameters.

Training Performance

In this section, we include the results in terms of performance achieved in training by SMoSE, tuned with $M = 8$ experts, and weighing the load-balancing losses in Eq.s (3)-(4) with $\lambda = 0.1$. The value for the M parameter has been empirically tuned to strike a balance between agents' performance and easiness of interpretation, as shown in the ablation study included in Appendix. The weight λ was tuned between 0.01 and 1.0 on a three-dimensional grid, to ensure fair load and reduce expert collapse. Details on the computational setup are available in the Appendix. To achieve statistical reliability in our results, we perform $N_{\text{train}} = 10$ independent training runs, seeded from 0 to 9. Every training run consists of a sequence of episodes with a maximum horizon of $H = 1000$ interactions with the environment, which is achieved if the controller does not incur catastrophic failure.

We train for a total of one million environmental interactions (i.e., timesteps), to be comparable with the closed-box

Environment	Algorithm	avg. ER	N_{act} (N_{tot})
Walker2d-v4	CGP	1090.00* ± 59.50*	—
	LGP	1080.00* ± 14.00*	—
	Metric-40	775.00* ± 115.50*	1006
	SMoSE (ours)	4224.29 ± 25.96	360 (1872)
Hopper-v4	CGP	1150.00* ± 92.50*	—
	LGP	1120.00* ± 87.50*	—
	Metric-40	2005.00* ± 295.00*	643
	SMoSE (ours)	2816.08 ± 445.57	168 (672)
Ant-v4	CGP	1130.00* ± 222.50*	—
	LGP	1210.00* ± 390.00*	—
	Metric-40	2210.50* ± 175.50*	1488
	SMoSE (ours)	3245.43 ± 380.93	672 (3808)
HalfCheetah-v4	CGP	6375.00* ± 496.50*	—
	LGP	6388.50* ± 296.50*	—
	Metric-40	2210.50* ± 175.50*	1006
	SMoSE (ours)	7310.17 ± 131.57	360 (1872)
Reacher-v4	CGP	− 68.50* ± 43.75*	—
	LGP	− 58.50* ± 11.10*	—
	SMoSE (ours)	− 5.49 ± 2.32	360 (1872)
Swimmer-v4	CGP	280.00* ± 7.50*	—
	LGP	278.50* ± 14.00*	—
	SMoSE (ours)	45.40 ± 1.62	108 (360)

'*' = visually derived from the plots reported in the original papers
'—' = number of employed parameters not specified in literature
magenta = best score per environment

Table 1: **Performance and model size comparison**. SMoSE outperforms interpretable methods.

Environment	Algorithm	avg. ER	N_{act} (N_{tot})
Walker2d-v4	SAC-L	4358.06 ± 582.94	73484
	SAC-M	4020.51 ± 192.75	1842
	SAC-S	2967.14 ± 77.18	372
	PPO	3362.16 ± 793.40	5708
	SMoSE (ours)	4224.29 ± 25.96	360 (1872)
Hopper-v4	SAC-L	2636.49 ± 424.21	70406
	SAC-M	3224.25 ± 177.90	672
	SAC-S	3076.09 ± 178.37	186
	PPO	2311.90 ± 654.90	5126
	SMoSE (ours)	2816.08 ± 445.57	168 (672)
Ant-v4	SAC-L	5255.46 ± 1070.65	77072
	SAC-M	4894.18 ± 599.64	3800
	SAC-S	4162.97 ± 298.12	720
	PPO	2327.12 ± 871.63	6480
	SMoSE (ours)	3245.43 ± 380.93	672 (3808)
HalfCheetah-v4	SAC-L	11809.87 ± 256.10	73484
	SAC-M	8992.22 ± 80.58	1842
	SAC-S	7214.30 ± 87.29	372
	PPO	2308.29 ± 1526.87	5708
	SMoSE (ours)	7310.17 ± 131.57	360 (1872)
Reacher-v4	SAC-L	− 3.75 ± 1.50	69892
	SAC-M	− 4.02 ± 1.61	484
	SAC-S	− 4.82 ± 1.81	148
	PPO	− 6.57 ± 2.37	4930
	SMoSE (ours)	− 5.49 ± 2.32	144 (480)
Swimmer-v4	SAC-L	68.59 ± 2.87	69124
	SAC-M	71.94 ± 1.89	355
	SAC-S	59.42 ± 2.89	108
	PPO	93.26 ± 19.90	4868
	SMoSE (ours)	45.4 ± 1.62	108 (360)

magenta = best score per environment

Table 2: **Performance and model size comparison**. SMoSE narrows the gap with non-interpretable methods.

methods literature. The ER achieved by SMoSE in training over the six mentioned environments is represented in Figure 2, where we plot the mean and standard deviation of episodic returns over the environment interactions. We assign the ER to all the timesteps of the same episode, and then we plot the ER at each timestep, to make episodes with different lengths comparable. In the same figure, the ER achieved by closed-box policies is included as a reference. For a fair comparison, we consider neural policies trained with SAC (the same RL algorithm we use to learn our policy), tuned with the same set of hyperparameters, included in the Appendix. Moreover, we consider non-linear structures that exploit a comparable amount of parameters with our interpretable policy. In particular, our model has a total number N_{tot} of parameters, considering both the router and the M experts; however, for each decision, only N_{act} active parameters are used (i.e., those associated with the router and the selected expert), which alleviates the computational cost of the decision-making process. The figure includes the results in training associated with a first neural architecture, indicated as SAC-M, that exploits N_{tot} parameters for each decision. Additionally, the figure shows as well the results of a second neural architecture, indicated as SAC-S, that exploits N_{act} parameters for each decision, and hence fully comparable in inference to our policy. Both N_{tot} and N_{act} for our method are detailed in Tables 1-2. From Figure 2, we can see that SMoSE's performance is close to the one of SAC-S on three over six environments (HalfCheetah-v4, Hopper-v4, and Reacher-v4), and sensibly better than it on one over six environments (Walker2d-v4), while it is close to the result achieved by SAC-M on two over six environments (Walker2d-v4 and Reacher-v4). These evaluations are affected by the performance on Ant-v4, where we note that the initial behavior, similar again to the one of SAC-S, is worsened by a small number of final under-average training realizations (3 over 10), while higher performance is achieved in the majority of the cases.

Policy Evaluation

After training, we evaluate the performance of the learned policies in Eq. (1) once deployed, and compare it with interpretable and not-interpretable methods on the six environments. We test deterministic linear experts obtained discarding the learned standard deviations σ_m as it is often done with SAC (Haarnoja et al. 2018), while employing the coefficients $\{\, \theta_m \,|\, m = 1, \ldots, M \,\}$ in combination with the router g in Eq. (2), with the learned parameters Θ. In our evaluation campaign, per each environment, we test all the $N_{\text{train}} = 10$ policies on $N_e = 100$ independent episodes with maximum horizon $H = 1000$ (1000 episodes in total), and we measure the average ER (avg. ER) achieved by the method. Numerical results on this are included in Tables 1-2.

For comparison, we consider the results reported in (Nadizar, Medvet, and Wilson 2024) for CGP and LGP, and the ones achieved by Metric-40 in (Akrour, Tateo, and Peters 2021) (where only 4 out of the 6 environments were tested, i.e., all except Reacher-v4 and Swimmer-v4), the

best-performing policy trained in that work, characterized by a MoE architecture tuned with 40 experts. The three methods are briefly discussed in the Related Work section. Moreover, we consider as non-linear competitors the classic version of Proximal Policy Search (PPO) (Schulman et al. 2017), and three neural architectures trained with SAC (Haarnoja et al. 2018), characterized by different numbers of parameters. In particular, we consider SAC-M and SAC-S, previously described in the analysis of training performance, and we add a third, larger network, SAC-L. The architecture underlying SAC-L corresponds to the neural architecture employed in (Haarnoja et al. 2018). For PPO, we consider the performance benchmarked in (Huang et al. 2022), while, for the SAC-based methods, we train and evaluate them with the same procedure described for SMoSE.

We can see in Table 1 that SMoSE consistently outperforms its competitors in five over six environments (all except Swimmer-v4). Additionally, Table 2 underlines that on average, our method's performance is closer to the one of the non-interpretable competitors, showing how SMoSE is performing consistently better than PPO (on five environments over six, all but Swimmer-v4), and on average close to the SAC-based approaches of comparable size, namely, SAC-M and SAC-S. Table 2 shows that SAC-L often achieves the best performance. This architecture is advantaged not only by exploiting non-linearity but also by the high number of parameters at its disposal (see Table 2, last column), which, however, makes it computationally more expensive, both in training and in inference.

As mentioned, Swimmer-v4 appears to be the most difficult environment for SMoSE, being the only one in which SMoSE does not outperform any of the interpretable methods, as well as PPO. To further comment on this, we can notice in Table 2 that the performance in this environment by all the SAC-based policies (including SMoSE) is weaker.

Policy Interpretation

This section includes the interpretation of the best policy learned for the Reacher-v4 environment. We include interpretations for the other five environments in the Appendix. Figure 3 presents a graphical representation of the weights of the MoE policy, including both the router and the experts. Details on both are included in the following, but we can already notice from the figure that the controller employs almost exclusively a small subset of variables:

- coordinates of the target (T): $x_{\mathrm{T}},\, y_{\mathrm{T}}$
- coordinates' difference between the fingertip (f) and T: $\Delta_x = x_f - x_{\mathrm{T}},\ \Delta_y = y_f - y_{\mathrm{T}}$

The two control variables, the torques applied to the first and second joint, are indicated as τ_1 and τ_2, respectively. We indicate as S_m the score of the m-th expert, i.e., the "weight" computed by the corresponding column of the router.

Expert 1 ($S_1 \approx 2.7\, y_{\mathrm{T}} - 5.6\, \Delta_y$) Expert 1 is called when its score S_1 is greater than all the other scores. Its policy can be described as:

$$\begin{cases} \tau_1 \approx x_{\mathrm{T}}, \\ \tau_2 \approx y_{\mathrm{T}} - 4\, \Delta_x. \end{cases} \tag{5}$$

According to this, the first joint is strongly accelerated in a direction that is opposite to that of x_{T}, while the second joint's control signal is composed of two terms, one that moves the joint in the opposite direction of y_{T}, and the other that tries to minimize the distance on the x-axis between the fingertip and the target.

Expert 2 ($S_2 \approx -3.5\, y_{\mathrm{T}} + 8.3\, \Delta_y$) It can be noted that Expert 2's score S_2 has opposite signs w.r.t. S_1, indicating that these two experts are likely working in opposite states. Its policy can be described as follows:

$$\begin{cases} \tau_1 \approx 3.1\, y_{\mathrm{T}} - 2.7\, \Delta_x - 4.5\, \Delta_y, \\ \tau_2 \approx 2.8\, y_{\mathrm{T}} + 3.0\, \Delta_x - 3.4\, \Delta_y, \end{cases} \tag{6}$$

shows that the two joints' control signals have partially similar behaviors. Indeed, both of them have a dependency on y_{T}, which can be seen as a feed-forward control scheme combined with a feed-back control scheme (Tao, Kosut, and Aral 1994). This is suggested by the presence, in both controllers, of an amplified version of y_{T}, and a negative dependency on Δ_y (please, note that in this environment Δ can be seen as the opposite of the control error). We note, though, a strong difference between the two joints' controllers: while τ_1 has a negative dependency on Δ_x (suggesting that it tries to reach the target also on the x-axis), τ_2 shows the opposite. Interestingly, the magnitude of the two contributions is comparable, suggesting that these terms are used to balance the fingertip by simultaneously moving the two joints.

Expert 3 ($S_3 \approx 3.9\, y_{\mathrm{T}} - 5.8\, \Delta_y$) The score S_3 has similar coefficients to S_1, but with higher weight to y_{T}, meaning that Expert 3 is preferred over Expert 1 when y_{T} has a high magnitude. The policy of this expert can be summarized as:

$$\begin{cases} \tau_1 \approx 2.9\, y_{\mathrm{T}} + 4.8\, \Delta_x, \\ \tau_2 \approx -3.6\, x_{\mathrm{T}} + 2.4\, \Delta_x + 5.0\, \Delta_y. \end{cases} \tag{7}$$

Most of the terms in (7) try to move away from the target (i.e., positive dependencies on Δ_x and Δ_y, as well as a negative dependency on x_{T}. However, the positive dependency on y_{T} in τ_1, shows an attempt to strike a balance between moving towards the desired y_{T} and moving away from x_{T}. This likely builds momentum for reaching the target in the following steps, through the use of other experts.

Expert 4 ($S_4 \approx 4.1\, y_{\mathrm{T}} + 2.1\, \Delta_x - 6.0\, \Delta_y$) Expert 4 implements a simple policy, described by:

$$\begin{cases} \tau_1 \approx -3.9\, x_{\mathrm{T}}, \\ \tau_2 \approx 2.9\, \Delta_y. \end{cases} \tag{8}$$

This policy tends to move away from the target. In fact, in τ_1 we have a negative dependency on x_{T}, and in τ_2 a positive one w.r.t. $\Delta_y = y_f - y_{\mathrm{T}}$, thus a positive weight pushing y_f farther away from y_{T}.

Expert 5 ($S_5 \approx 3.5\, \Delta_x$) This expert's policy is:

$$\begin{cases} \tau_1 \approx 5.9\, y_{\mathrm{T}} - 5.2\, \Delta_y, \\ \tau_2 \approx -3.4\, x_{\mathrm{T}} + 13\, y_{\mathrm{T}}. \end{cases} \tag{9}$$

The torque τ_1 aims for y_f to reach y_{T} through combined feed-forward and feed-back control approaches, while τ_2

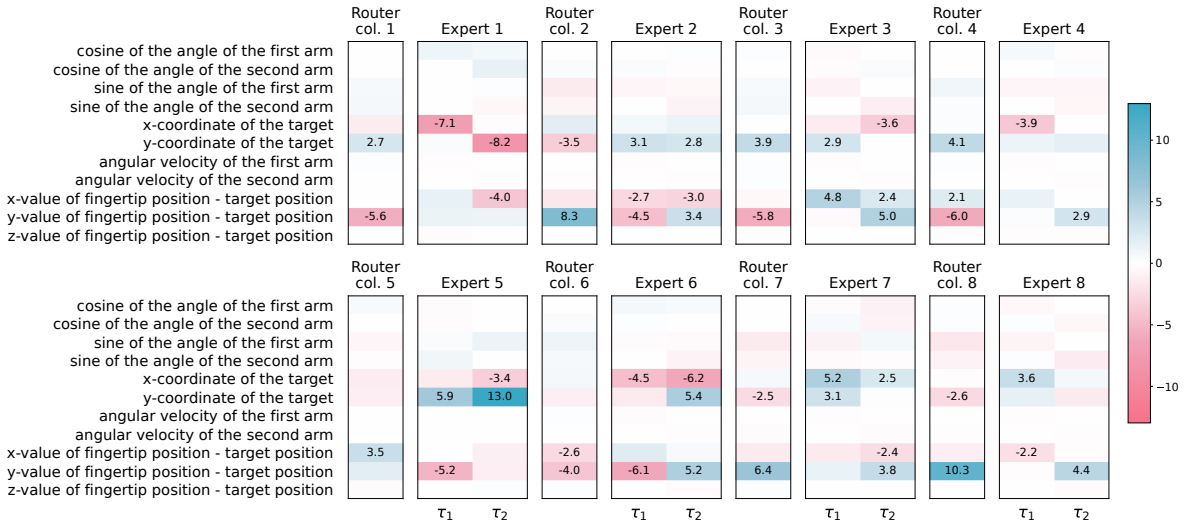

Figure 3: **Reacher-v4**. Visualization of the weights for each expert and the corresponding column of the router's weight matrix.

pushes the fingertip to reach y_T only by using feed-forward control. Moreover, τ_2 has a non-negligible negative dependency on x_T, contributing to keeping the joint away from it.

Expert 6 ($S_6 \approx -2.6\,\Delta_x - 4.0\,\Delta_y$) Score S_6, suggests that Expert 6 is called when the difference between the fingertip's and the target's coordinates have large negative values. The policy works as follows:

$$\begin{cases} \tau_1 \approx -4.5\,x_T - 6.1\,\Delta_y, \\ \tau_2 \approx -6.2\,x_T + 5.4\,y_T + 5.2\,\Delta_y. \end{cases} \tag{10}$$

While this policy is more intricate, we can interpret all the individual contributions, which will be combined at runtime (akin to the superposition principle in linear systems). In τ_1, we have a negative contribution from x_T, pushing the joint away from it, and a negative contribution w.r.t. Δ_y, moving the joint in such a way that the distance between the fingertip and the target (on the y-axis) is reduced. On τ_2 we have a negative dependency on x_T, and a positive dependency on y_T (akin to feed-forward control), moving the joint in the same direction of y_T, plus a positive dependency on Δ_y, which makes the fingertip move away from the target on the y-axis. While these two effects seem opposite, it is worth noticing that we can rework the equation as follows: $\tau_2 \approx -6.2\,x_T + 5.4\,y_T + 5.2\,\Delta_y = -6.2\,x_T + 5.4\,y_T + 5.2\,y_f - 5.2\,y_T \approx -6.2\,x_T + 5.2\,y_f$. This revised equation can be easily interpreted as: τ_2 tends to (i) move away from x_T, and (ii) move towards the positive side of the y-axis.

Expert 7 ($S_7 \approx -2.5\,y_T + 6.4\,\Delta_y$) Score S_7 suggests that Expert 7 is queried when the target is on the negative side of the y-axis, and the distance (also on the y-axis) has a large value (implying that the fingertip's y coordinate is larger than y_T). The policy is:

$$\begin{cases} \tau_1 \approx 5.2\,x_T + 3.1\,y_T, \\ \tau_2 \approx 2.5\,x_T - 2.4\,\Delta_x + 3.8.\Delta_y. \end{cases} \tag{11}$$

We can see that τ_1 uses a kind of feed-forward control to move towards both x_T and y_T. Instead, τ_2, moves joint 2 towards x_T (also exploiting Δ_x), simultaneously moving it away from y_T, through a positive contribution w.r.t. Δ_y.

Expert 8 ($S_8 \approx -2.6\,y_T + 10.3\,\Delta_y$) Score S_8, similarly to S_7, suggests that Expert 8 is called when $y_T \leq 0$ and $y_f \geq y_T$. However, the larger weights suggest that this expert is called way more often than Expert 7. Its policy is:

$$\begin{cases} \tau_1 \approx 3.6\,x_T - 2.2\,\Delta_x, \\ \tau_2 \approx 4.4\,\Delta_y. \end{cases} \tag{12}$$

Here τ_1 can be interpreted as simply performing combined feed-forward and feed-back control, while τ_2 tends to move the joint farther from the target (on the y-axis).

Conclusions

In this paper, we introduced SMoSE, a novel approach for training sparsely activated and interpretable controllers using a *top*-1 Mixture-of-Experts architecture. By integrating interpretable shallow decision-makers, each specializing in different basic skills, and an interpretable router for task allocation, SMoSE strikes a balance between performance and interpretability. The evaluation of SMoSE presented in this work demonstrates its competitive performance, outperforming existing interpretable baselines and narrowing the performance gap with non-interpretable state-of-the-art methods. The transparency of SMoSE is also showcased in this work, through an in-depth interpretation. Additionally, by distilling DTs from the learned router, we supply an additional tool to facilitate the interpretation of the trained models, making SMoSE a compelling choice for scenarios where both high performance and interpretability are required. As future work, we plan to explore the potential of SMoSE in more complex environments and extend it to multi-agent scenarios, exploiting socially-inspired reward designs to achieve interpretable cooperative and coordinated AI policies.

Acknowledgments

We acknowledge ISCRA for awarding this project access to the LEONARDO supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CINECA (Italy). This project is funded by the European Union. However, the views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. Leonardo Lucio Custode and Giovanni Iacca acknowledge funding by the European Union (project no. 101071179). Laura Ferrarotti and Bruno Lepri acknowledge funding by the European Union's Horizon Europe research and innovation program under grant agreement No. 101120237 (ELIAS) and under grant agreement No. 101120763 (TANGO).

References

Akrour, R.; Tateo, D.; and Peters, J. 2021. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10): 6795–6806.

Arrieta, A. B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, 58: 82–115.

Chan, C. S.; Kong, H.; and Liang, G. 2022. A Comparative Study of Faithfulness Metrics for Model Interpretability Methods. arXiv:2204.05514.

Cheng, G.; Dong, L.; Cai, W.; and Sun, C. 2023. Multi-task reinforcement learning with attention-based mixture of experts. *IEEE Robotics and Automation Letters*, 8(6): 3812–3819.

Custode, L. L.; and Iacca, G. 2021. A co-evolutionary approach to interpretable reinforcement learning in environments with continuous action spaces. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8. IEEE.

Custode, L. L.; and Iacca, G. 2023. Evolutionary Learning of Interpretable Decision Trees. *IEEE Access*, 11: 6169–6184.

Custode, L. L.; and Iacca, G. 2024. Social Interpretable Reinforcement Learning. *arXiv preprint arXiv: 2401.15480*.

Delfosse, Q.; Shindo, H.; Dhami, D.; and Kersting, K. 2023. Interpretable and Explainable Logical Policies via Neurally Guided Symbolic Abstraction. arXiv:2306.01439.

Delfosse, Q.; Sztwiertnia, S.; Rothermel, M.; Stammer, W.; and Kersting, K. 2024. Interpretable Concept Bottlenecks to Align Reinforcement Learning Agents. arXiv:2401.05821.

Dhebar, Y.; Deb, K.; Nageshrao, S.; Zhu, L.; and Filev, D. 2020. Interpretable-AI Policies using Evolutionary Nonlinear Decision Trees for Discrete Action Systems. *arXiv preprint arXiv:2009.09521*.

Dwivedi, R.; Dave, D.; Naik, H.; Singhal, S.; Omer, R.; Patel, P.; Qian, B.; Wen, Z.; Shah, T.; Morgan, G.; et al. 2023. Explainable AI (XAI): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9): 1–33.

Glanois, C.; Weng, P.; Zimmer, M.; Li, D.; Yang, T.; Hao, J.; and Liu, W. 2024. A survey on interpretable reinforcement learning. *Machine Learning*, 1–44.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *CoRR*, abs/1801.01290.

He, S. 2021. Who is liable for the UBER self-driving crash? Analysis of the liability allocation and the regulatory model for autonomous vehicles. *Autonomous Vehicles: Business, Technology and Law*, 93–111.

Huang, S.; Dossa, R. F. J.; Ye, C.; Braga, J.; Chakraborty, D.; Mehta, K.; and AraÃšjo, J. G. 2022. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274): 1–18.

Huang, X.; Kroening, D.; Ruan, W.; Sharp, J.; Sun, Y.; Thamo, E.; Wu, M.; and Yi, X. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37: 100270.

Irsoy, O.; Yıldız, O. T.; and Alpaydın, E. 2012. Soft decision trees. In *International Conference on Pattern Recognition*, 1819–1822. IEEE.

Jiang, Z.; and Luo, S. 2019. Neural Logic Reinforcement Learning. arXiv:1904.10729.

Kimura, D.; Ono, M.; Chaudhury, S.; Kohita, R.; Wachi, A.; Agravante, D. J.; Tatsubori, M.; Munawar, A.; and Gray, A. 2021. Neuro-Symbolic Reinforcement Learning with First-Order Logic. arXiv:2110.10963.

Kohler, H.; Delfosse, Q.; Akrour, R.; Kersting, K.; and Preux, P. 2024. Interpretable and Editable Programmatic Tree Policies for Reinforcement Learning. arXiv:2405.14956.

Liu, G.-T.; Hu, E.-P.; Cheng, P.-J.; yi Lee, H.; and Sun, S.-H. 2023. Hierarchical Programmatic Reinforcement Learning via Learning to Compose Programs. arXiv:2301.12950.

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

McGough, M. 2018. How bad is Sacramento's air, exactly? Google results appear at odds with reality, some say. *Sacramento Bee*, 7.

Nadizar, G.; Medvet, E.; and Wilson, D. G. 2024. Naturally Interpretable Control Policies via Graph-Based Genetic Programming. In *European Conference on Genetic Programming (Part of EvoStar)*, 73–89. Springer.

Obando-Ceron, J.; Sokar, G.; Willi, T.; Lyle, C.; Farebrother, J.; Foerster, J.; Dziugaite, G. K.; Precup, D.; and Castro, P. S. 2024. Mixtures of experts unlock parameter scaling for deep rl. *arXiv preprint arXiv:2402.08609*.

Paleja, R.; Chen, L.; Niu, Y.; Silva, A.; Li, Z.; Zhang, S.; Ritchie, C.; Choi, S.; Chang, K. C.; Tseng, H. E.; et al. 2023. Interpretable Reinforcement Learning for Robotics and Continuous Control. *arXiv preprint arXiv:2311.10041*.

Pyeatt, L. D.; Howe, A. E.; et al. 2001. Decision tree function approximation in reinforcement learning. In *International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*, volume 2, 70–77.

Ren, J.; Li, Y.; Ding, Z.; Pan, W.; and Dong, H. 2021. Probabilistic mixture-of-experts for efficient deep reinforcement learning. *arXiv preprint arXiv:2104.09122*.

Riquelme, C.; Puigcerver, J.; Mustafa, B.; Neumann, M.; Jenatton, R.; Pinto, A. S.; Keysers, D.; and Houlsby, N. 2021. Scaling Vision with Sparse Mixture of Experts. *CoRR*, abs/2106.05974.

Roth, A. M.; Topin, N.; Jamshidi, P.; and Veloso, M. 2019. Conservative q-improvement: Reinforcement learning for an interpretable decision-tree policy. *arXiv preprint arXiv:1907.01180*.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215.

Rudin, C.; Chen, C.; Chen, Z.; Huang, H.; Semenova, L.; and Zhong, C. 2021. Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges. *arXiv preprint arXiv:2103.11251*.

Rudin, C.; Wang, C.; and Coker, B. 2019. The age of secrecy and unfairness in recidivism prediction. *arXiv preprint arXiv:1811.00731*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

Sha, J.; Shindo, H.; Delfosse, Q.; Kersting, K.; and Dhami, D. S. 2024. EXPIL: Explanatory Predicate Invention for Learning in Games. arXiv:2406.06107.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *arXiv preprint arXiv:1701.06538*.

Silva, A.; Killian, T.; Rodriguez, I. D. J.; Son, S.-H.; and Gombolay, M. 2020. Optimization Methods for Interpretable Differentiable Decision Trees in Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*, 1855–1865. PMLR.

Smyth, J.; Ulahannan, A.; Florek, F.; Shaw, E.; and Mansfield, N. 2021. Understanding misuse of partially automated vehicles–A discussion of NTSB's findings of the 2018 mountain view Tesla crash. Technical report, Chartered Institute of Ergonomics and Human Factors (CIEHF).

Tao, K. M.; Kosut, R. L.; and Aral, G. 1994. Learning feedforward control. In *American Control Conference*, volume 3, 2575–2579. IEEE.

Timofeev, R. 2004. Classification and regression trees (CART) theory and applications. *Humboldt University, Berlin*, 54: 48.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Varshney, K. R.; and Alemzadeh, H. 2017. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big Data*, 5(3): 246–255.

Verma, A.; Le, H. M.; Yue, Y.; and Chaudhuri, S. 2021. Imitation-Projected Programmatic Reinforcement Learning. arXiv:1907.05431.

Verma, A.; Murali, V.; Singh, R.; Kohli, P.; and Chaudhuri, S. 2019. Programmatically Interpretable Reinforcement Learning. arXiv:1804.02477.

Videau, M.; Leite, A.; Teytaud, O.; and Schoenauer, M. 2022. Multi-objective genetic programming for explainable reinforcement learning. In *European Conference on Genetic Programming (Part of EvoStar)*, 278–293. Springer.

Wexler, R. 2017. When a computer program keeps you in jail. *The New York Times*, 13: 1.

Willi, T.; Obando-Ceron, J.; Foerster, J.; Dziugaite, K.; and Castro, P. S. 2024. Mixture of Experts in a Mixture of RL settings. arXiv:2406.18420.