INTERPRETABLE POLICY EXTRACTION WITH NEURO-SYMBOLIC REINFORCEMENT LEARNING

Rajdeep Dutta[1*], *Qincheng Wang*[2], *Ankur Singh*[1†], *Dhruv Kumarjiguda*[2†], *Li Xiaoli*[1], *Senthilnath Jayavelu*[1*]

[1]Institute for Infocomm Research (I2R), A*STAR, [2]Nanyang Technological University, Singapore.

ABSTRACT

This paper presents a novel RL algorithm, S-REINFORCE, designed by leveraging two types of function approximators, namely Neural Network (NN) and Symbolic Regressor (SR), to produce numerical and symbolic policies for dynamic decision-making tasks, respectively. A symbolic policy uncovers functional relations between the underlying states and action-probabilities. Further, the symbolic policy is utilized through importance sampling (IS) to improve the rewards received during the learning process. The effectiveness of S-REINFORCE has been validated on various dynamic decision-making problems involving low and high dimensional action spaces. The results obtained clearly demonstrate that by leveraging the complementary strengths of NN and SR, S-REINFORCE generates policies that exhibit both good performance and interpretability. This makes S-REINFORCE an excellent choice for real-world applications where transparency and causality play a crucial role.

Index Terms— interpretable policy, policy gradient, symbolic regression, importance sampling

1. INTRODUCTION

While reinforcement learning (RL) has gained popularity due to its effectiveness in finding optimal solutions to various sequential decision-making tasks ranging from two-player games [Silver et al.(2018)], indoor localization [Salimibeni and Mohammadi(2023)] to text semantic analysis [Gao et al.(2022)] and drug discovery [Popova et al.(2018)], the underlying function approximation remains a challenge. In the realm of RL, neural networks (NNs) are commonly used to approximate complex, nonlinear and/or unknown functions. However, NNs lack interpretability in the learned input-output mapping [Landajuela et al.(2021)], involving numerous non-linear operators and transformations that hinder the deployment of NN-generated policies in real-world applications. In contrast, symbolic regressors (SRs) offer interpretable input-output mappings by employing symbolic basis function expansions in their approximations [Soni et al.(2022)]. Unlike NNs, SRs do not require a large amount of data for fitting purposes. Symbolic policies derived from SRs are inherently interpretable, transparent, and easily reproducible due to their functional forms. Moreover, these policies offer economic deployment solutions while meeting memory or latency-related constraints [Landajuela et al.(2021)].

To obtain compact and tractable analytical policies, several offline methods exist, including symbolic value iteration, symbolic policy iteration, and direct solutions to the Bellman equation [Alibekov et al.(2016), Kubalik et al.(2021)]. However, these require access to the governing dynamics equations that are not available in complex, uncertain environments [Kubalik et al.(2021)]. When the governing state-transition laws are unavailable, alternative techniques such as model distillation and regression-based approaches can be leveraged to symbolically approximate policies [Hein et al.(2018)]. However, a conflict arises between the training objective of mimicking a pre-trained offline policy and the evaluation metric for improving the RL agent's performance. Recently, researchers have explored the use of autoregressive recurrent neural networks to symbolically approximate a distribution over a discrete sequence of tokens representing operators, input variables, and constants [Landajuela et al.(2021)]. However, their RNN-based risk-seeking policy gradient approach suffers from limited exploration ability due to early commitment and initialization bias. In this work, we utilize the exploration power of a population-based genetic programming approach for SR and transfer knowledge online from NN to SR to offer an economical solution.

Although SR produces analytical functions, the symbolic regression incurs a higher computational cost compared to training NN. This is because when functions are encoded by strings of symbols, the number of such strings grows exponentially with the string length [Udrescu and Tegmark(2020), Biggio et al.(2021)], giving rise to the combinatorial challenge of an exponentially large search space (NP-hard problem). To strike a balance between the interpretability of solutions and the associated computational costs, this study leverages the strengths of both NN and SR within an RL

[†] authors contributed equally to this work.

[*] corresponding authors

This study is supported by the Accelerated Materials Development for Manufacturing Program at A*STAR via the AME Programmatic Fund by the Agency for Science, Technology and Research under Grant No. A1898b0043.

framework. The NN component is exhaustively trained to learn a numerical probability distribution over the possible actions, while the SR component is periodically fitted to capture the functional relationships between the underlying states and the corresponding action probabilities.

To the best of our knowledge, this is the first attempt to generate symbolic policies in a cost-effective manner by combining the strengths of NN and SR in RL. The contributions of this work are highlighted as follows: (i) During the learning process, the periodic knowledge transfer from NN to SR mitigates the computational costs of symbolic regression throughout all training episodes. (ii) Both NN and SR are trained concurrently in our approach, allowing flexibility in the choice of the learned approximators, and on completion of the training, any of the two learnt approximators can be used for prediction. (iii) Experimental results demonstrate that our proposed approach outperforms its baseline counterpart in terms of interpretability and performance.

2. THE PROPOSED APPROACH

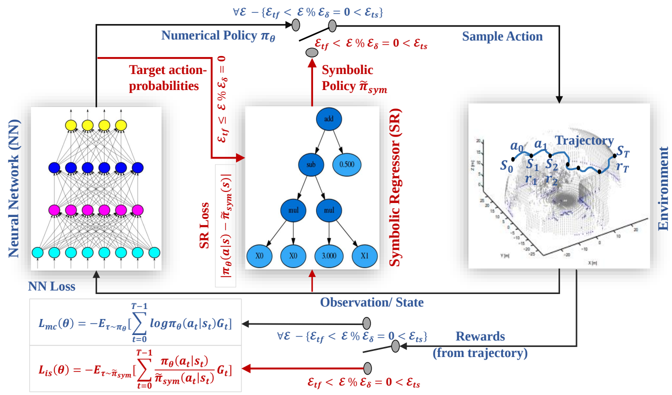
Our proposed approach for generating symbolic policies consists of three steps: (i) an NN is trained to update the numerical parameters of a policy along its gradient, (ii) an SR is fitted to extract a symbolic policy at regular intervals, concurrently while training the NN, and (iii) the extracted symbolic policy is utilized through importance sampling (IS) at regular intervals to obtain improved rewards during training.

2.1. Policy Gradient: S-REINFORCE

Here, we investigate sequential decision-making tasks modeled as Markov Decision Processes (MDPs). At time step t, the agent observes its current state s_t and takes an action a_t, which moves it to the next state s_{t+1} with a reward r_{t+1}. Typically, an MDP is defined by a tuple of information: $<\mathbf{S}, \mathbf{A}, \mathbf{P}, R, \gamma, T>$, where $\mathbf{S}$ denotes a set of states, $\mathbf{A}$ is a set of actions, $\mathbf{P} = \{p(s' = s_{t+1}, r' = r_{t+1} | s = s_t, a = a_t)\}$ represents a set of transition probabilities, R is the reward function; γ is the discount factor, and T is the horizon of a trajectory $\tau = \{s_0, a_0, s_1, r_1, ..., s_{T-1}, a_{T-1}, s_T, r_T\}$ generated by following a policy. A policy π is a mapping from $\mathbf{S}$ to a probability distribution over $\mathbf{A}$, and $\rho^\pi(s)$ is the probability of being in state s while following policy π.
The return expected from a trajectory τ (of horizon T) generated by following a parameterized policy π_θ, is given by: $J(\theta) = \mathbf{E}_{\tau\sim\pi_\theta}[R(\tau)]$; $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$. The objective function, $J(\theta)$, is maximized by updating the policy parameters along its gradient ascent direction. The policy gradient is evaluated by: $\nabla_\theta J(\theta) = \mathbf{E}_{\tau\sim\pi_\theta}[R(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)]$. In accordance with REINFORCE, the return from a trajectory $R(\tau)$ is estimated through the discounted future rewards (rewards-to-go) [Williams(1992)]; the rewards-to-go at the t^{th} transition is: $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \; \forall t \in \{0, 1, ..., T-1\}$. Then, the associated policy gradient turns into

$$\nabla_\theta J_{mc}(\theta) = \mathbf{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t] \, . \quad (1)$$

Note that the REINFORCE algorithm finds an unbiased estimate of the gradient (1) using Monte Carlo sampling, without assistance of any value function [Sutton et al.(1999)]. Let



$\forall \mathcal{E} - \{\mathcal{E}_{tf} < \mathcal{E} \,\%\, \mathcal{E}_\delta = 0 < \mathcal{E}_{ts}\}$
Numerical Policy π_θ
$\mathcal{E}_{tf} < \mathcal{E} \,\%\, \mathcal{E}_\delta = 0 < \mathcal{E}_{ts}$
Sample Action
Target action-probabilities
$\mathcal{E}_{tf} \leq \mathcal{E} \,\%\, \mathcal{E}_\delta = 0$
Symbolic Policy $\tilde{\pi}_{sym}$
Neural Network (NN)
SR Loss
$|\pi_\theta(a|s) - \tilde{\pi}_{sym}(s)|$
add
sub
0.500
mul
mul
X0
X0
3.000
X1
Symbolic Regressor (SR)
Trajectory
a_0 a_1 S_0 S_1 S_2 r_1 r_2 S_T r_T
Environment
NN Loss
Observation/ State
$L_{mc}(\theta) = -E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t)G_t]$
$\forall \mathcal{E} - \{\mathcal{E}_{tf} < \mathcal{E} \,\%\, \mathcal{E}_\delta = 0 < \mathcal{E}_{ts}\}$
Rewards (from trajectory)
$L_{is}(\theta) = -E_{\tau\sim\tilde{\pi}_{sym}}[\sum_{t=0}^{T-1} \frac{\pi_\theta(a_t|s_t)}{\tilde{\pi}_{sym}(a_t|s_t)} G_t]$
$\mathcal{E}_{tf} < \mathcal{E} \,\%\, \mathcal{E}_\delta = 0 < \mathcal{E}_{ts}$

Fig. 1: A high-level visualization of the proposed RL-framework composed of two function approximators, NN and SR. Note that $(\mathcal{E}_{tf} < \mathcal{E} \,\%\, \mathcal{E}_\delta = 0 < \mathcal{E}_{ts}) \equiv (\mathcal{E}_{tf} < \mathcal{E} < \mathcal{E}_{ts})$ and $(\mathcal{E} \,\%\, \mathcal{E}_\delta = 0)$.

us consider that data $\mathcal{D}$ contains the set of state-action pairs encountered in a trajectory sampled using the current policy [Hanna and Stone(2019)]. In this context, the Monte Carlo estimator of the policy gradient can be expressed as: $\nabla_\theta J_{mc}(\mathcal{D}) = \mathbf{E}_{s\sim\rho^{\mathcal{D}}, a\sim\pi_{\mathcal{D}}}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t]$, where $\nabla_\theta J_{mc}(\mathcal{D})$ is an unbiased estimator rendering an accurate estimate for repeatedly sampled batches of data. However, for a single batch of data, only a limited number of states are visited prior to evaluating the gradient, which leads to an inaccurate estimate. The sampling error arises because the expectation in $\nabla_\theta J_{mc}$ is taken over a deviated action distribution $\pi_{\mathcal{D}}$ instead of the actual action distribution π_θ [Hanna and Stone(2019)]. The correct state distribution is usually unknown; however, the availability π_θ can be utilized through IS to design a sampling correction [Hanna and Stone(2019)]. The gradient estimate's variance due to the stochastic nature of action selection is also reduced by such a sampling correction.

In this regard, the expected return from a trajectory is computed via IS that exploits a symbolic policy, $\tilde{\pi}_{sym}$ ($\approx \pi_{\mathcal{D}}$ when fitted), for sample collection, which is compensated by a ratio of probability distributions in the objective as: $J(\theta) = \mathbf{E}_{\tau\sim\tilde{\pi}_{sym}}[\frac{\pi_\theta(\tau)}{\tilde{\pi}_{sym}(\tau)} R(\tau)]$, where the ratio $\frac{\pi_\theta(\tau)}{\tilde{\pi}_{sym}(\tau)} = \frac{\prod_{i=0}^{T-1} \pi_\theta(a_t|s_t)}{\prod_{i=0}^{T-1} \tilde{\pi}_{sym}(a_t|s_t)}$ can be simplified by means of the likelihood ratios, using Markov's principle of causality [Jie and Abbeel(2010)]. Consequently, the policy gradient form in Eq. (1) turns into

$$\nabla_\theta J_{is}(\theta) = \mathbf{E}_{\tau\sim\tilde{\pi}_{sym}}[\sum_{t=0}^{T-1} \frac{\pi_\theta(a_t|s_t)}{\tilde{\pi}_{sym}(a_t|s_t)} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t]. \quad (2)$$

Note that the variance induced by the IS-based estimator, i.e. $Var[\frac{\pi_\theta(\tau)}{\tilde{\pi}_{sym}(\tau)}R(\tau)] = \mathbf{E}_{\tau\sim\pi_\theta}[\frac{\pi_\theta(\tau)}{\tilde{\pi}_{sym}(\tau)}R^2(\tau)] - J^2$, becomes high when π_θ and $\tilde{\pi}_{sym}$ are different and low when $\tilde{\pi}_{sym}(\tau) \propto \pi_\theta(\tau)|R(\tau)|$ is satisfied [Jie and Abbeel(2010), Tokdar and Kass(2010)], indicating IS to be applied when an RL agent receives high rewards during the training phase.

The proposed symbolic variant of the REINFORCE algorithm is dubbed as S-REINFORCE, and the related flow and training schedule are depicted in Figs. 1 and 2, respectively.

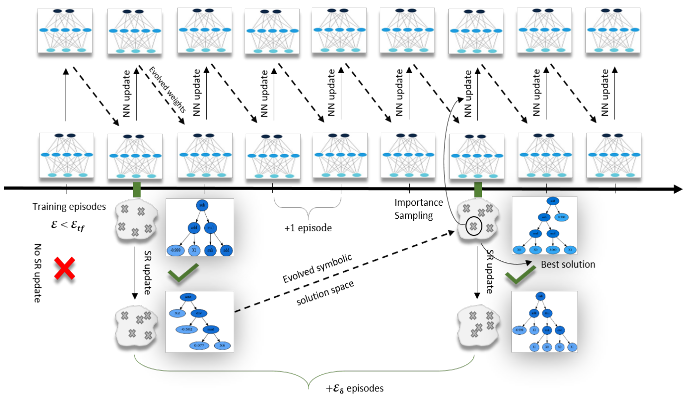


Fig. 2: Low-level visualization of two approximators trained concurrently: NN is trained throughout all episodes; SR fitting starts at episode $\mathcal{E} = \mathcal{E}_{tf}$, and thereafter continues at regular intervals of $\mathcal{E} \,\%\, \mathcal{E}_\delta = 0$. Actions are sampled from the numerical policy (π_θ) more often except for episodes $\mathcal{E}_{tf} < \mathcal{E} \,\%\, \mathcal{E}_\delta = 0 < \mathcal{E}_{ts}$, when sampled from the symbolic policy ($\tilde{\pi}_{sym}$); the respective NN loss functions are $L_{mc}(\theta)$ and $L_{is}(\theta)$.

2.2. Neuro-Symbolic Function Approximation

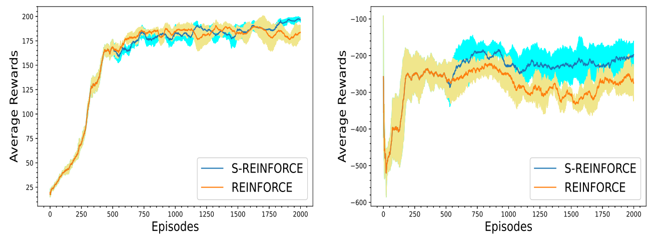
To numerically approximate the mapping from states to action probabilities, which is the solution to Equation (1), an NN is used. The generated policy evolves by updating NN weights to minimize the loss: $L_{mc}(\theta) = -\mathbf{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T-1}\log\pi_\theta(a_t|s_t)G_t]$. While an NN is being trained, we fit an SR to capture the functional forms of the evolving policy distributions. Along with the NN-training episodes, the SR is fitted starting from episode $\mathcal{E}_{tf}$, and thereafter, the fitting is continued at regular intervals of $\mathcal{E}_\delta$ episodes. For training episodes $\mathcal{E}_{tf} \leq \mathcal{E} \,\%\, \mathcal{E}_\delta = 0$, the input to the SR contains the states encountered in the preceding trajectory, i.e. $\{s_t\}_{t=0}^{T-1} \in \Re^{T\times d}$, formed using actions sampled from the NN-approximated policy, and the target is composed of the action-probabilities responsible for the state transitions in that trajectory, i.e. $\{\pi_\theta(a_t|s_t)\}_{t=0}^{T-1} \in \Re^{T\times 1}$.

An SR seeks to find the mathematical expression of a policy by minimizing an error functional, e_m, between the target, $\pi_\theta(a|s)$, and the predicted output, $\tilde{\pi}_{sym}(s)$. The solution to this functional optimization problem is the optimal function: $\tilde{\pi}^*_{sym}(s) = arg\min_{\tilde{\pi}_{sym}} e_m(\pi_\theta(a|s), \tilde{\pi}_{sym}(s))$. In practice, we use genetic programming (GP) to solve this regression, promoting an evolving population of candidate programs to attain the optimal solution that best captures the relations between the given input and the targeted output. Each candidate program is represented by a tree made up of numbers, process variables, and symbolic basis functions [Stephens(2018), Soni et al.(2022)]. During the evolutionary search, the exploitation and exploration of the solution space are taken care of by *crossover* and *mutation* operators, respectively.

3. EMPIRICAL RESULTS

We apply our proposed approach on two commonly used benchmark and one real-world environments.

In the CartPole-v0 environment with a discrete action space [Duan et al.(2016)], an SR is fitted along with training an NN for 2000 episodes (Table 1). The reward profiles in Fig. 3a reveal that the performance of S-REINFORCE improves over REINFORCE towards the end of training ($>$ 1500 episodes), leading to a higher final reward (Table 2). The extracted symbolic policies are shown in Table 2, which indicate that the action selection is independent of s_0, s_1.



(a) CartPole environment. (b) Lunar Lander environment.

Fig. 3: Performance comparison of the obtained rewards: the solid lines along with the shaded regions show (mean $\pm$ standard deviation) of the average rewards across five different random seeds.

In the LunarLanderContinuous-v2 environment with a continuous action space [Asadi et al.(2017)], along with training two NNs, four distinct SRs are fitted to find the symbolic means and variances of two Gaussian action-probability distributions (Table 1). Usually, it takes around 20,000 episodes to solve the LunarLander environment with REINFORCE [Asadi et al.(2017)]. In this study, we present a performance comparison between REINFORCE and S-REINFORCE agents up to 2000 episodes. The rewards obtained with REINFORCE increase gradually after around 500 episodes in Fig. 3b, although this increase is not consistent. On the other hand, the increase in the rewards is well maintained by S-REINFORCE and the reward profile has less fluctuations compared to REINFORCE. By maintaining a consistent increase in rewards and exhibiting reduced fluctuations, S-REINFORCE demonstrates its capability to generate more reliable and well-performing policies. The obtained expressions in Table 2 signify that the expected first and second actions depend on five and four states, respectively; however, the choice of actions is independent of s_2.

Reinforcement Learning for Structural Evolution (ReLeaSE): This environment consists of two deep neural networks, a *generator* and a *predictor*. During pre-training, the generator learns to produce chemically viable molecules and the predictor learns to evaluate the generator's performance using a supervised learning algorithm [Popova et al.(2018)].

Scenario	NN parameters	SR parameters	SR occurrence	IS
CartPole	(FC-128 with tanh)+(FC-128 with tanh)+(FC-128 with softmax) learning rate= 5×10^{-4}	population_size= 2000, tournament_size= 20, p_crossover = 0.7, p_subtree_mutation= 0.1, p_hoist_mutation= 0.05, p_point_mutation= 0.1, Basis: [add, sub, mul, div, inv, cos]	$\mathcal{E} \geq 400$ and $\mathcal{E}$ % $10 = 0$	$500 \leq \mathcal{E} \leq 1800$ and $\mathcal{E}$ % $10 = 0$
Lunar Lander	(FC-128 with tanh)+(FC-128 with tanh)+FC-128# learning rate= 5×10^{-4}	population_size= 2000, tournament_size= 20, p_crossover = 0.7, p_subtree_mutation= 0.1, p_hoist_mutation= 0.05, p_point_mutation= 0.1, Basis: [add, sub, mul, div, min, max]	$\mathcal{E} \geq 400$ and $\mathcal{E}$ % $10 = 0$	$500 \leq \mathcal{E} \leq 1800$ and $\mathcal{E}$ % $10 = 0$

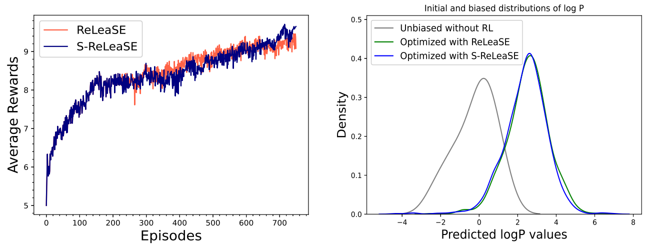
Table 1: Implementation details for benchmark environments: 'FC-128' denotes a fully-connected layer with 128 neurons; # the output layers are different for approximating the mean and variance of a Gaussian action-probability distribution.

Next, an RL-agent is leveraged to produce molecules with a desired property by fine-tuning the pre-trained generator.

Scenario	Reward$_p$	Reward$_b$	Policy Expression
CartPole	196.25 ± 2.28	183.73 ± 7.41	$\pi(a_1) = 0.49 - 2s_2 - 0.78s_3$ $\pi(a_2) = 0.51 + 2s_2 + 0.78s_3$
Lunar Lander	-274.91 ± 83.35	-294.72 ± 66.45	$\mu(a_0) = 3.61 - s_0 - 7.04s_1 - 5s_3 -$ $-2s_4 - 1.45s_5 - s_5^2$ $\sigma(a_0) = max(0.01, 0.95\frac{s_4 s_5 s_7}{s_1})$ $\mu(a_1) = 0.95\frac{s_4 s_5 s_7}{s_1}$ $\sigma(a_1) = max(0.01, 0.95\frac{s_4 s_5 s_7}{s_1})$

Table 2: Performance evaluation: Reward$_p$ and Reward$_b$, expressed as (mean $\pm$ standard deviation), denote the rewards achieved with S-REINFORCE and REINFORCE, respectively, after 2000 episodes of training across five different random seeds. The reported policy expression represents the best performing symbolic policy.

The *generator* and *predictor* architectures are adopted from an earlier study [Isayev(2018)], [Popova et al.(2018)], with parameters: number of iterations or n_iterations = 50, number of trajectories or n_policy = 15, number of batches or n_policy_replay = 10, and number of molecules generated for $\log P$ prediction = 1000. The training loss is averaged over all the batches, and the molecule properties are monitored over $50 \times 15 = 750$ episodes. Distinct SRs are employed to capture functional relations in 45 action-probabilities, with parameters: population_size= 500, tournament_size=20, p_crossover = 0.7, p_subtree_mutation= 0.1, p_hoist_mutation= 0.05, p_point_mutation= 0.1, the basis functions = [add, sub, mul, div, cos]. The SRs are fitted for (n_iterations >= 5 and %5 = 0) and (n_policy_replay %10 = 0), and IS is applied for (n_iterations > 10 and %5 = 0) and (n_policy_replay %10 = 0).



(a) ReLeaSE environment. (b) Predicted properties.

Fig. 4: Performance of the generator fine-tuned with RL. Example of two generated molecules: SMILES - Brc1ccccc1, C=C(C=CC)Sc1ccccc1 with logP values 2.45 and 3.87, respectively.

After 750 episodes of training, the maximum average reward (9.5) achieved with the symbolic ReLeaSE (S-ReLeaSE) is slightly better than the baseline (9.2), although the slopes of both the reward profiles are similar in Fig. 4a. Here, IS does not give much improvement in rewards as numerous visited states from multiple trajectories are considered in repeatedly sampled batches of data prior to updating the policies. Fig. 4b exhibits that the logP distribution over 1000 molecules predicted with the fine-tuned generator shifts into the desired zone effectively, as compared to the same predicted with the pre-trained generator. However, the logP distributions predicted with ReLeaSE and S-ReLeaSE are similar. The density curves indicate that the fine-tuned (biased) generator is able to produce much more desired molecules than the pre-trained (unbiased) one.
The log-probability expressions for the first two actions are:

$$\log \pi(a_0) = \frac{cos(s_{577})}{cos^3(s_{287}) - cos(s_{234})} \times \{cos^3(s_{572} \times s_{342} + \\ + cos^2(cos(s_{577}) + cos(s_{583} \times s_{33})) - s_{248}) + cos(s_{697})\};$$
$$\log \pi(a_1) = s_{697} - cos(s_{408}) - cos(s_{621}) - s_{936} - s_{131} - cos(s_{408}).$$

Prior to sampling actions, the above functional policies are normalized, $\sum_{i=0}^{44} \pi(a_i) = 1$. Note that $\pi(a_0), \pi(a_1)$ are uncorrelated as they do not depend on any common states. Interestingly, each policy relates to a limited number of states.
Discussion: The achieved results justify that the proposed algorithm can produce explicit policy expressions with causality, while offering a higher final reward than the baseline. IS is applied in this study to correct the policy shift using on-policy data, rather than off-policy data [Levine and Koltun(2013)]. Figs. 3a and 3b reveal noticeable dips in the reward profiles after 500 episodes due to inadequate fitting of the associated SRs, which disappear gradually as the accuracy of symbolic regression improves and the adopted IS becomes more effective. Therefore, achieving optimal performance requires accurate symbolic regression in conjunction with training NN. The choice of basis functions and hyper-parameters in SR is crucial here, in addition to the proper tuning of knowledge transfer-related parameters: $\mathcal{E}_{tf}$, $\mathcal{E}_{\delta}$ and $\mathcal{E}_{ts}$.

4. CONCLUSION

By simultaneously training numerical and symbolic approximators, the proposed approach not only improves the rewards received by the RL agent but also generates interpretable policy expressions. Our methodology has been tested in dynamic scenarios with low and high dimensional action spaces, and shown to be effective in generating appropriate policies. This work serves as a proof-of-concept for combining numerical and symbolic policy approximators into an RL framework. While this is a significant step forward, in the future, we will extend the underlying idea to more advanced RL techniques that require both value and policy approximations.

5. REFERENCES

[Alibekov et al.(2016)] Eduard Alibekov, Jiri Kubalik, and Robert Babuska. 2016. Symbolic method for deriving policy in reinforcement learning. In *IEEE 55th Conference on Decision and Control (CDC)*. IEEE, Las Vegas, NV, USA, 2789–2795.

[Asadi et al.(2017)] Kavosh Asadi, Cameron Allen, Melrose Roderick, Abdel-rahman Mohamed, George Konidaris, Michael Littman, and Brown University Amazon. 2017. Mean Actor Critic. *stat* 1050 (2017), 1.

[Biggio et al.(2021)] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. 2021. Neural symbolic regression that scales. In *International Conference on Machine Learning*. PMLR, Virtual, 936–945.

[Duan et al.(2016)] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*. PMLR, New York, 1329–1338.

[Gao et al.(2022)] Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling Laws for Reward Model Overoptimization. *arXiv* 2210.10760 (2022), 1–28.

[Hanna and Stone(2019)] Josiah Hanna and Peter Stone. 2019. Reducing sampling error in the monte carlo policy gradient estimator. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, Montreal.

[Hein et al.(2018)] Daniel Hein, Steffen Udluft, and Thomas A. Runkler. 2018. Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence* 76 (2018), 158–169.

[Isayev(2018)] Olexandr Isayev. 2018. ReLeaSE (Reinforcement Learning for Structural Evolution). `https://github.com/isayev/ReLeaSE`.

[Jie and Abbeel(2010)] Tang Jie and Pieter Abbeel. 2010. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, Vol. 23. NIPS, Vancouver.

[Kubalik et al.(2021)] Jiri Kubalik, Erik Derner, Jan Zegklitz, and Robert Babuska. 2021. Symbolic regression methods for reinforcement learning. *IEEE Access* 9 (2021), 139697–139711.

[Landajuela et al.(2021)] Mikel Landajuela, Brenden K. Petersen, Sookyung Kim, Claudio P. Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F. Pettit, and Daniel Faissol. 2021. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, Virtual, 5979–5989.

[Levine and Koltun(2013)] Sergey Levine and Vladlen Koltun. 2013. Guided policy search. In *International conference on machine learning*. PMLR, Atlanta, 1–9.

[Popova et al.(2018)] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. 2018. Deep reinforcement learning for de novo drug design. *Science Advances* 4, 7 (2018), eaap7885.

[Salimibeni and Mohammadi(2023)] Mohammad Salimibeni and Arash Mohammadi. 2023. RL-IFF: Indoor Localization via Reinforcement Learning-Based Information Fusion. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Greece, 1–5.

[Silver et al.(2018)] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, and Marc et al Lanctot. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[Soni et al.(2022)] Tejas Soni, Ashwani Sharma, Rajdeep Dutta, Annwesha Dutta, Senthilnath Jayavelu, and Saikat Sarkar. 2022. Capturing functional relations in fluid–structure interaction via machine learning. *Royal Society open science* 9, 4 (2022), 220097.

[Stephens(2018)] Trevor Stephens. 2018. gplearn: Genetic programming in python. `https://github.com/trevorstephens/gplearn`

[Sutton et al.(1999)] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, Vol. 12. Morgan Kaufmann Publishers Inc., Denver.

[Tokdar and Kass(2010)] Surya T. Tokdar and Robert E. Kass. 2010. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics* 2, 1 (2010), 54–60.

[Udrescu and Tegmark(2020)] Silviu-Marian Udrescu and Max Tegmark. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances* 6, 16 (2020), eaay2631.

[Williams(1992)] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.