Interpretable pipelines with evolutionarily optimized modules for reinforcement learning tasks with visual inputs

Leonardo Lucio Custode
University of Trento
Trento, Italy
leonardo.custode@unitn.it

Giovanni Iacca
University of Trento
Trento, Italy
giovanni.iacca@unitn.it

ABSTRACT

The importance of explainability in AI has become a pressing concern, for which several explainable AI (XAI) approaches have been recently proposed. However, most of the available XAI techniques are post-hoc methods, which however may be only partially reliable, as they do not reflect exactly the state of the original models. Thus, a more direct way for achieving XAI is through interpretable (also called glass-box) models. These models have been shown to obtain comparable (and, in some cases, better) performance with respect to black-boxes models in various tasks such as classification and reinforcement learning. However, they struggle when working with raw data, especially when the input dimensionality increases and the raw inputs alone do not give valuable insights on the decision-making process. Here, we propose to use end-to-end pipelines composed of multiple interpretable models co-optimized by means of evolutionary algorithms, that allows us to decompose the decision-making process into two parts: computing high-level features from raw data, and reasoning on the extracted high-level features. We test our approach in reinforcement learning environments from the Atari benchmark, where we obtain comparable results (with respect to black-box approaches) in settings without stochastic frame-skipping, while performance degrades in frame-skipping settings.

CCS CONCEPTS

• **Computing methodologies** → **Artificial life**; *Cooperation and coordination*; *Neural networks*; • **Theory of computation** → *Multi-agent learning*.

KEYWORDS

Reinforcement Learning, Interpretability, Co-evolution, Atari

ACM Reference Format:
Leonardo Lucio Custode and Giovanni Iacca. 2022. Interpretable pipelines with evolutionarily optimized modules for reinforcement learning tasks with visual inputs. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528897

1 INTRODUCTION

While the progress in AI continues to achieve new milestones, there is a growing concern on the need for understanding the decision-making process of AI models, especially in critical applications. This awareness originated the subfield of *explainable AI* (XAI), which has the goal to design tools to explain the decisions made by AI models, usually by means of post-hoc techniques. However, while such techniques seem promising, they suffer from a fundamental issue: rather than reflecting the internal state of the explained model [21, 33], they mainly focus on explaining its output, and how its behavior depends (either locally or globally) on the features of the problem at hand. For this reason, another subfield of AI has been catching on in the past few years, namely that of *interpretable AI* (IAI) [4]. Differently from XAI, IAI focuses on the development of *inherently* interpretable models (also called "explainable by design" or "glass-box" models, as opposed to the traditional black-box ones), i.e., models that are directly understandable for humans without any post-hoc explanation.

In safety-critical and high-stakes applications, explainability is an issue not only from a technical standpoint, but also from a legal and, to some extent, ethical perspective. The recent literature has proposed some seminal approaches for performing RL with interpretable models [9, 28], e.g., based on evolutionary computation [7, 8]. So far, these interpretable reinforcement learning (IRL) approaches have been mostly tested on relatively simple control RL tasks. However, these methods are not expected to work well in RL tasks with raw input data, as in these contexts each variable alone (e.g., a pixel) may not be meaningful enough to take decisions.

In this work, we introduce the concept of *interpretable pipelines* for tackling RL tasks with visual inputs. An interpretable pipeline is a set of agents where each agent is an interpretable model with well-defined responsibilities, which communicates with the other agents in the pipeline. We optimize such pipelines by means of a co-evolutionary approach, in which different evolutionary algorithms (EAs) run in parallel, each of which optimizes a single agent. We test our approach on three different Atari games, where we observe that the proposed method is able to achieve satisfactory performance in deterministic settings (i.e., without frame-skipping). On the other hand, our approach is not able to achieve satisfactory performance in environments with stochastic frame-skipping (yielding higher uncertainty about the future), which provides some hints for future work.

The paper is structured as follows. The next section makes a brief overview of the related work. Section 3 explains the methodology used in our experiments. In Section 5 we present the experimental setup and the results and, finally, in Section 6 we draw the conclusions of this work.

2 RELATED WORK

IRL has recently gained attention in the research community. Silva et al., in [28], employed differentiable decision trees trained by means of the PPO algorithm [27]. This approach gives satisfactory results when using the differentiable version of the decision trees (which have low interpretability). However, when the produced trees are discretized into decision trees with hard splits (which have high interpretability), significant losses in performance occur.

Another interesting approach to IRL has been proposed in [9]. Here, the authors introduced a methodology that optimizes decision trees with non-linear splits by using an EA. The results show that the proposed approach works well in discrete-action settings. However, the highly non-linear splits limit the interpretability of the solutions produced.

In [8], the authors proposed a methodology based on Grammatical Evolution (GE) [22] and Q-learning [32] to produce decision trees that perform online learning. In [7], this approach was further extended in order to handle RL tasks with continuous action spaces by using a co-evolutionary approach.

As mentioned earlier, the main limitation of these approaches is that while they can be effective in tasks with a small number of high-level features, they are not expected to work in environments with high-dimensional, low-level features, such as images. In fact, in the latter scenario, each input to the system does not provide significant information for the decision-making process.

Interpretability, usually intended as a binary property, is ill-defined. For this reason, some works proposed approaches to quantitatively measure interpretability.

In [30], the authors learned a metric of interpretability by training a regression model on the results of a survey.

The idea of using complexity as a proxy for interpretability was also proposed in [3], where the authors stated that the computational complexity of a model can be used as a metric of interpretability as it directly resembles the number of operations that must be interpreted by humans.

3 METHOD

To evolve interpretable pipelines for image-based RL tasks, we build on some of the aforementioned previous works from the literature [7, 8, 29]. In detail, our proposed system is an interpretable pipelines composed of two parts: a vision module, that is meant to process the input to extract a pre-defined number of features; and a decision module, whose purpose is to decide which action to take, based on the features extracted by the vision module.

A graphical representation of this kind of pipelines is shown in Figure 1. In the following subsections, we will first explain the details of the two kinds of modules, and then we will describe our co-evolutionary approach.

Similarly to the approach proposed in [29], in this work we use a vision module, whose purpose is to find the k most important patches in the image, returning their coordinates. The vision module uses k convolutional kernels, that are easy to interpret, since each kernel has the duty of recognizing one and only one entity.

The goal of the decision module, instead, is to perform "reasoning" on the coordinates of the most important patches of the input images and to take a decision on top of them.

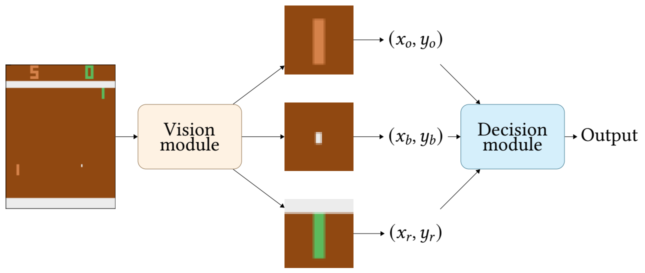


Figure 1: An example of the proposed pipelines (note that the frames are taken from the Pong environment).

To keep the interpretability of the pipelines high, we use an automatically synthesized decision tree as decision module. This decision tree takes as input the list of coordinates computed by the vision module, thus it does not use the raw data of the whole image.

3.1 Co-evolutionary process

To optimize the vision and decision modules adopted within the proposed pipelines, we employ a co-evolutionary approach [20]. In particular, we combine Covariance Matrix Adaptation Evolution Strategies (CMA-ES) [13] with Genetic Programming (GP) [16]. We use CMA-ES to evolve the parameters of the vision module, i.e., the weights of each kernel module. CMA-ES has been chosen for being one of the most robust algorithms for derivative-free optimization. On the other hand, by using Genetic Programming (more specifically, strongly typed Genetic Programming [19]), we evolve decision trees, as described below.

3.1.1 *Genetic Programming for evolving decision trees.*

To evolve decision trees, we use two types of nodes: condition nodes and leaf nodes. A condition is represented as a node with three child nodes: a comparison node and two nodes (either leaves or conditions).

A comparison node is composed of a node representing an operator (e.g., "less than", "equal to", and "greater than") that has two child nodes, encoding two expressions.

An expression node can be either a constant, a variable, or an arithmetical operation between expression nodes.

3.1.2 *Fitness evaluation.*

In order to evaluate the quality of the individuals from both populations, we pair each individual with all the individuals of the other population.

We evaluate the pair on e episodes, and we compute the average score (across episodes) for the pair $\bar{s}_{i,j}$, where i is the index of the individual from the population of vision modules and j is the index of the individual from the population of decision modules.

Then, we define the fitness of an individual as the maximum $\bar{s}$ that that individual obtained across all the pairings, i.e. $f_i = \underset{j}{max}(\bar{s}_{i,j})$ for vision modules and $f_j = \underset{i}{max}(\bar{s}_{i,j})$ for decision modules.

3.1.3 *Reducing the number of evaluations.*

The computational cost (in terms of number of evaluations, where a single evaluation relates to an instance of the proposed pipelines) for the co-evolutionary process is $O(p_c \cdot p_g \cdot g \cdot e)$, where p_c is the population size for CMA-ES, p_g is the population size for the Genetic Programming (assuming that each individual of one population is paired with all the individuals of the other population), g is the number of generations, and e is the number of episodes.

To reduce such cost, we propose a mechanism that evaluates the *behavior* (i.e., the output) of each individual in the two populations, and avoids the evaluation of individuals whose behavior is too similar. At each generation, for both the vision and decision modules in the current populations of the co-evolutionary process, we give them in input a set of samples and we store their outputs. Then, for both populations (separately), we cluster these outputs by means of the DBSCAN algorithm [26]. Once clustering has been performed, we evaluate only the centroids of each cluster (and the individuals not assigned to any cluster) and we assign, for each individual of the cluster, the same fitness.

4 EXPERIMENTAL SETUP

We test our approach in three environments from the Atari Learning Environment implemented in OpenAI Gym [5, 6, 18]. In particular, we use the Pong-v4, Bowling-v4, and Boxing-v4 environments, hereinafter simply referred to as Pong, Bowling and Boxing, respectively. For each environment, we consider both the setting with and without frame-skipping, the first one being harder than the second one. For each environment and setting, we perform 5 runs. This number of runs is enough to ensure statistical significance since, given the results shown in Table 1, the confidence interval (95%) is low enough to validate our conclusions (see the next section for the results). The confidence interval has been computed as $CI_{95\%} = \frac{t_{\{5-1,0.025\}}\sigma}{\sqrt{5}}$, where t is the critical value from the Student's t distribution, and σ is the standard deviation.

4.1 Image pre-processing

Before feeding an image to the vision module, we perform the following pre-processing steps: first, we remove the topmost 35 pixels: these pixels correspond to the part of the image that describes the "status" of the game, thus we remove it to avoid that the evolved pipelines use this information to take decisions; then, we resize the image to 96×96: this operation speeds up the vision modules' computations, without losing information about the entities present in the game; finally, we normalize the input in $[0, 1]$ by performing a min-max normalization.

5 EXPERIMENTAL RESULTS

The results obtained from the 5 available runs for each environment and setting are shown in Table 1. The results shown in the table have been obtained by testing the best evolved pipelines on 100 unseen episodes. We observe that our approach performs well in the environments without frame-skipping, but it performs poorly in settings with frame-skipping. While state-of-the-art approaches are able to achieve very good performance even in cases with frame-skipping, it is important to point out that these approaches are not interpretable, thus they do not provide any information about their inner processes. On the other hand, while our approaches do not perform well when trained in setups with frame-skipping, they are completely transparent, potentially allowing an adaptation to domains with frame-skipping.

Table 1: Summary of the results of the best pipelines evolved in 5 runs of each setting for each environment. Each pipeline has been tested on 100 unseen episodes. "FS" stands for the setting with frame-skipping, "NoFS" indicates the setting without frame-skipping, "SoTA" indicates the results from the (non-interpretable) state-of-the-art. Please note that all the papers from the literature report results only in the FS setting.

Env.	Setup	Mean	Std.	Best	Reference
Pong	FS (ours)	-6.97	8.49	7.42	
	NoFS (ours)	21.00	0.00	21.00	
	FS (SoTA)	-	-	21.00	[23, 24, 31]
Bowling	FS (ours)	189.68	5.06	196.53	
	NoFS (ours)	220.20	18.00	240.00	
	FS (SoTA)	-	-	260.00	[10, 24]
Boxing	FS (ours)	48.59	19.05	75.37	
	NoFS (ours)	92.78	3.09	98.00	
	FS (SoTA)	-	-	100.00	[2, 5, 11, 12, 14, 24, 25]

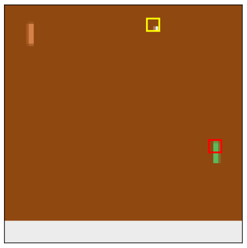
The fitness trends (not reported here for brevity) show that the clustering mechanisms allows us to save a significant amount of evaluations, increasing the efficiency of the co-evolutionary process (approximately saving 41%±12% evaluations).

The best pipelines obtained are shown graphically in Figure 2. In order to improve the readability, the decision modules have been manually simplified, deleting the conditions that always evaluate to the same truth value. It is easy to observe that the resulting policies are extremely simple to interpret. This result should encourage future research in IAI, as adding more complexity to this type of agents would still retain interpretability and, potentially, increase performance in frame-skipping settings.

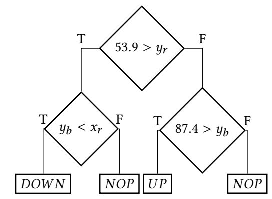
6 CONCLUSIONS AND FUTURE WORKS

Reinforcement learning (RL) has made significant progresses in recent years. However, mainstream RL methodologies, typically based on deep learning, are very hard to understand. In this paper, we proposed a novel methodology (based on a kind of divide-et-impera paradigm) for evolving interpretable systems for RL tasks with visual inputs. In particular, our approach is based on pipelines characterized by a separation of concerns between a vision module (which uses convolutional kernels) and a decision module (based on a decision tree). Our results show that our approach is able to learn how to effectively play three Atari games in simplified settings (i.e., without frame-skipping). However, when applying frame-skipping to the environments, our approach is not able to achieve satisfactory performance.

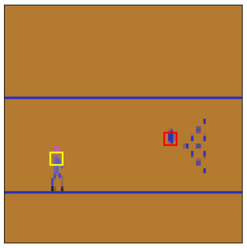
Future work should introduce ways to address the uncertainty in non-deterministic settings (i.e., with frame-skipping), in order to make this approach more robust to noise in the environment and achieve performances comparable to those of the state-of-the-art algorithms developed for these settings. In this sense, two possibilities would be to incorporate in our approach some mechanisms



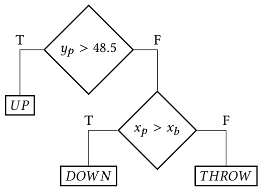
(a) Vision module - Pong



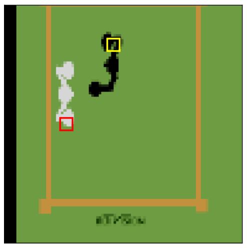
(b) Decision module - Pong



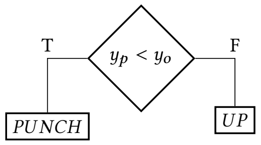
(c) Vision module - Bowling



(d) Decision module - Bowling



(e) Vision module - Boxing



(f) Decision module - Boxing

Figure 2: Best evolved pipelines for the three environments (without frame-skipping). On the top, we show the entities discovered by the vision modules, while on the bottom we show the corresponding decision modules.

used in evolutionary optimization in the presence of noise [1], or using fuzzy [15] or probabilistic [17] decision trees.

REFERENCES

[1] Dirk V Arnold and Hans-Georg Beyer. 2002. *Noisy optimization with evolution strategies*. Vol. 8. Springer, Boston, MA, USA.

[2] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. 2020. Agent57: Outperforming the Atari Human Benchmark. arXiv:2003.13350.

[3] Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux. 2020. Model Interpretability through the Lens of Computational Complexity. arXiv:2010.12265.

[4] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (June 2020), 82–115.

[5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. 2013. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research* 47 (jun 2013), 253–279.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:1606.01540.

[7] Leonardo Lucio Custode and Giovanni Iacca. 2021. A co-evolutionary approach to interpretable reinforcement learning in environments with continuous action spaces. In *IEEE Symposium Series on Computational Intelligence*. IEEE, New York, NY, USA, 1–8.

[8] Leonardo Lucio Custode and Giovanni Iacca. 2021. Evolutionary learning of interpretable decision trees. arXiv:2012.07723.

[9] Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. 2020. Interpretable-AI Policies using Evolutionary Nonlinear Decision Trees for Discrete Action Systems. arXiv:2009.09521.

[10] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2021. First return, then explore. *Nature* 590, 7847 (Feb 2021), 580–586.

[11] Jiajun Fan, Changnan Xiao, and Yue Huang. 2022. GDI: Rethinking What Makes Reinforcement Learning Different From Supervised Learning. arXiv:2106.06232.

[12] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. 2017. Noisy networks for exploration. arXiv:1706.10295.

[13] Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *IEEE International Conference on Evolutionary Computation*. IEEE, New York, NY, USA, 312–317.

[14] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. 2018. Distributed prioritized experience replay. arXiv:1803.00933.

[15] Cezary Z Janikow. 1998. Fuzzy decision trees: issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28, 1 (1998), 1–14.

[16] John R Koza and Riccardo Poli. 2005. Genetic programming. In *Search methodologies*. Springer, Boston, MA, USA, 127–164.

[17] Balaji Lakshminarayanan. 2016. *Decision trees and forests: a probabilistic perspective*. Ph.D. Dissertation. University College London.

[18] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. 2018. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research* 61 (2018), 523–562.

[19] David J Montana. 1995. Strongly typed genetic programming. *Evolutionary computation* 3, 2 (1995), 199–230.

[20] Elena Popovici, Anthony Bucci, R Paul Wiegand, and Edwin D De Jong. 2012. Coevolutionary Principles. Citeseer.

[21] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (May 2019), 206–215. Number: 5 Publisher: Nature Publishing Group.

[22] Conor Ryan, JJ Collins, and Michael O Neill. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming*, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Wolfgang Banzhaf, Riccardo Poli, Marc Schoenauer, and Terence C. Fogarty (Eds.). Vol. 1391. Springer Berlin Heidelberg, Berlin, Heidelberg, 83–96.

[23] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. arXiv:1703.03864.

[24] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, and et al. 2020. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (Dec 2020), 604–609.

[25] Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. 2021. Online and Offline Reinforcement Learning by Planning with a Learned Model. arXiv:2104.06294.

[26] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems* 42, 3 (2017), 1–21.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

[28] Andrew Silva, Taylor Killian, Ivan Dario Jimenez Rodriguez, Sung-Hyun Son, and Matthew Gombolay. 2020. Optimization Methods for Interpretable Differentiable Decision Trees in Reinforcement Learning. arXiv:1903.09338.

[29] Yujin Tang, Duong Nguyen, and David Ha. 2020. Neuroevolution of Self-Interpretable Agents. arXiv:2003.08165.

[30] Marco Virgolin, Andrea De Lorenzo, Eric Medvet, and Francesca Randone. 2020. Learning a Formula of Interpretability to Learn Interpretable Formulas. arXiv:2004.11170.

[31] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. arXiv:1511.06581.

[32] Christopher Watkins. 1989. *Learning From Delayed Rewards*. Ph.D. Dissertation. King's College.

[33] Yilun Zhou, Serena Booth, Marco Tulio Ribeiro, and Julie Shah. 2021. Do Feature Attribution Methods Correctly Attribute Features? arXiv:2104.14403.