Proceedings of Machine Learning Research vol vvv:1–12, 2025

Efficient Neuro-Symbolic Policy using In-Memory Computing

Tergel Molom-Ochir[†] TERGEL.MOLOM-OCHIR@DUKE.EDU
Department of ECE, Duke University, USA

Naman Saxena[†] NAMAN.SAXENA@DUKE.EDU
Department of ECE, Duke University, USA

Jiwoo Kim JIWOO.KIM@DUKE.EDU
Department of ECE, Duke University, USA

Yiran Chen YIRAN.CHEN@DUKE.EDU
Department of ECE, Duke University, USA

Miroslav Pajic MIROSLAV.PAJIC@DUKE.EDU
Department of ECE, Duke University, USA

Hai "Helen" Li HAI.LI@DUKE.EDU
Department of ECE, Duke University, USA

[†] **These authors contributed equally.**

Abstract

As AI systems grow in complexity, achieving computationally efficient and interpretable decision-making is crucial. Neuro-Symbolic AI (NeSy) offers a promising framework by integrating symbolic representation with neural learning, but its execution on traditional hardware remains inefficient due to memory bottlenecks and high computational costs. We envision a future where In-Memory Computing (IMC)-based acceleration fundamentally transforms Neuro-Symbolic policy acceleration by mapping it onto hardware-associative memory, enabling O(1) complexity decision-making with drastically reduced energy consumption and latency. Our preliminary results show that IMC-based symbolic policies achieve up to 100× speedup and six orders of magnitude better energy efficiency than CPU and GPU implementations. Furthermore, we discuss how probabilistic symbolic policies can be realized within IMC architectures, enabling AI systems to handle uncertainty while maintaining efficiency. This paper advocates for a paradigm shift in AI acceleration—moving beyond traditional von Neumann architectures toward memory-centric computation, unlocking real-time, scalable, and interpretable decision-making for next-generation AI applications.

Keywords: Neuro-Symbolic AI, In-Memory Computing, Hardware-Accelerated AI, Energy-Efficient AI

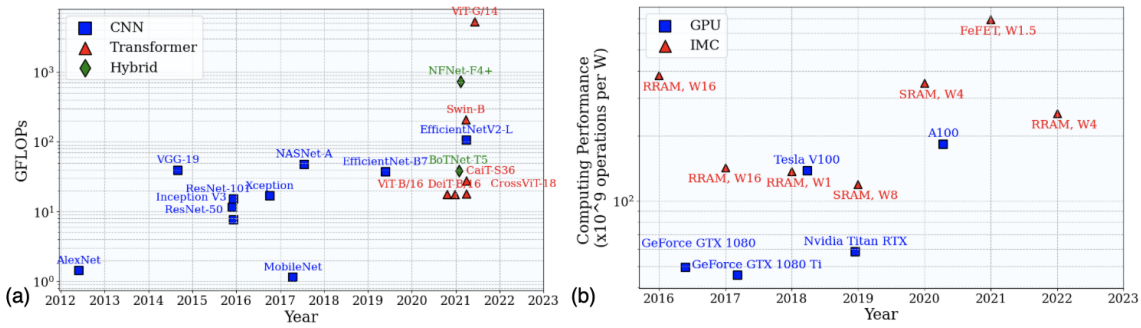

Figure 1: The figure on the left illustrates the exponential growth in the computational demands of neural network policies, highlighting an unsustainable trend in power and resource consumption. In contrast, the figure on the right showcases the steady advancements in In-Memory Computing (IMC) architectures, which are becoming increasingly capable of handling complex workloads. These trends underscore the timeliness and necessity of transitioning from purely neural network-based policies to neuro-symbolic policies, as modern IMC architectures now offer the efficiency and computational capacity required to support such hybrid models effectively.

1. Introduction

The reinforcement learning paradigm has proven effective at designing closed-loop controllers based on the feedback from the task at hand in the form of a reward. Further, the introduction of neural networks essentially gave birth to deep reinforcement learning and led to several groundbreaking works, surpassing human capabilities, such as AlphaGo (Silver et al. (2017)), and DQN (Van Hasselt et al. (2016)). Not only did deep RL-based controllers breach the ceiling of performance, but at the same time, they allowed RL-based policies to be used in a wide variety of applications. With the growing applicability, apart from the performance of the RL-based controller, several other issues have become attention, such as interpretability, safety, and energy consumption of controllers.

Most of the time, while designing the controller, the energy consumption is not considered, and it has become the default assumption that the platform used for the deployment of the controller will have enough resources to support the controller. This assumption no longer holds with the steady increment in the size of neural network (NN)-based controllers. Consequently, researchers have started focusing on Neuro-Symbolic reinforcement learning. Neuro-Symbolic (NeSy) paradigm uses interpretable structures such as finite state machines or decision trees to represent the controller. Such interpretable structures, by taking advantage of specialized hardware development, such as In-memory computing, could be used to reduce the energy consumption of controllers in general.

To alleviate the computational inefficiencies of traditional hardware for NeSy AI, we employ In-Memory Computing (IMC) architectures to accelerate controllers. Unlike traditional CPUs and GPUs, which suffer from high data movement overhead due to the von Neumann bottleneck, IMC executes computations within memory itself with reduced latency and power. By mapping NeSy policies onto IMC, we demonstrate that symbolic structure computations—such as decision-tree inference and rule-based logic execution—can be performed in *O(1)* complexity using specialized hardware. Not only is this energy efficient, but it enables real-time operation of interpretable symbolic policies. This work presents a comparative analysis that stresses the performance enhancement

Table 1: Generalized Neuro-Symbolic Architectures (Hamilton et al. (2024); Bhuyan et al. (2024))

Category	Example Architectures
Rule-Based & Symbolic AI Integration	Decision Tree-Augmented Neural Networks, Deep Symbolic Regression
Automata & Sequential Processing	Neural Turing Machines (NTM), FSM-Enhanced RNNs
Program Synthesis & Inductive Learning	Neural Program Induction, Program Synthesis Networks
Logic-Based Neural Networks	Logic Neural Networks (LNN), Logic Tensor Networks (LTN), Tensor Product Representation (TPR)
Graph & Knowledge Representation Models	Graph Neural Networks (GNNs) with Symbolic Reasoning, Recursive Neural Knowledge Networks (RNKN)
Reinforcement Learning & Decision-Making	AlphaZero, Neuro-symbolic Policy Learning
Neural-Symbolic Compilation & Theorem Proving	Neural Theorem Provers (NTP), Expression Trees from Neural Models

of IMC-based NeSy controllers over their NN-based equivalent, highlighting the scalability of IMC for low-power artificial intelligence applications.

In this paper, we propose the use of a Neuro-Symbolic decision tree controller, followed by its deployment on IMC, to demonstrate the benefits of energy consumption and execution time. In section 2, we discuss development in NeSy reinforcement learning and IMC architecture. Subsequently, in section 3, we describe the entire pipeline of our proposed method. Further, in section 4, we present a comparison between the energy consumption and throughput performance of NeSy controller and NN-based controller. Lastly, we delineate the challenges proposed by the current state of IMC architecture and discuss areas for improvement.

2. Related Works

2.1. Neuro-Symbolic Approach

NeSy approach comprises of several different methods (see Table 1). In the literature, for decision making, most of the NeSy approaches rely on using controllers based on discrete structures such as grammar. A grammar $\mathcal{G}$ is defined as $\{\mathcal{T}, \mathcal{N}, \mathcal{P}, \mathcal{S}\}$. Here, $\mathcal{T}$ denotes the set of terminal symbols. Terminal symbols are tantamount to constants and can be replaced by any other symbol while evaluating grammar. $\mathcal{N}$ represents the set of non-terminal symbol. Non-terminal symbols functions as variables and can be replace with other non-terminal or terminal symbols. Further, $\mathcal{P}$ denotes production rules that are mapping from the set of non-terminal symbols to a combined set of all the symbols. Lastly, $\mathcal{S}$ is a the state symbol. Once a grammar is assumed for a particular task, finite-state machine (Inala et al. (2020)), decision tree (Silva et al. (2020); Bastani et al. (2018); Vos and Verwer (2024)), or simple program (Verma et al. (2019, 2018)) could be obtained as the controller. Inala et al. (2020) uses a finite-state machine to obtain a hierarchical controller for tasks that require repetitive actions. It uses grammar to define the state-switching conditions and controller applicable in the state. Further, several works focus on program synthesis using grammar. Verma et al. (2019) uses a NN policy and imitation learning to perform updates to the programmatic policy. Moreover, Carvalho et al. (2024) learns programmatic policy by performing a search in the space of programmatic policy using search methods such as beam search, hill climbing, and etc. A decision tree could be seen as a special case of programmatic policy where the program only consists of

if-and-else statements. Our work uses decision tree policy, which we discuss in more detail in the next subsection.

2.2. Decision Tree-based Controller

Qiu and Zhu (2022) uses a grammar with *if-and-else* statements with affine conditions to obtain decision tree policy. The work proposes to learn the probability of introducing nested *if-and-else* conditions, increasing the decision tree's depth. Moreover, *if-and-else* conditions are also evaluated based on learned probability. Finally, the action is produced as a weighted sum of actions according to the probability of all *if-and-else* statements across all trees up to a certain depth. The previous work is an example of learning a stochastic decision tree. However, there are several works that explore deterministic trees and will be more suitable for taking advantage of the current state of IMC architecture. Vos and Verwer (2024) obtains the decision tree by using a decision tree classifier (DTC). First, the decision tree acts in the environment and produces trajectory. Later on, the actions in the trajectory are modified using the objective function of the Proximal Policy Optimization (Schulman et al. (2017)) algorithm. The next action generator becomes a target for the decision tree classifier corresponding to a particular state, and DTC is learned using supervised learning. Further, Bastani et al. (2018) proposes a method to learn decision tree policy by distilling from a NN-based policy. In this work, we focus on demonstrating how we can obtain symbolic policies with performance comparable to NN policies and show their efficiency with specialized hardware such as IMC architecture. Therefore, algorithm proposed by Bastani et al. (2018) is used as the base for our work. Next, we will discuss the development related to IMC architecture that supports symbolic policy.

2.3. Hardware Acceleration for Neuro-Symbolic AI

NeSy AI poses distinctive computational challenges that demand hardware acceleration tailored beyond general deep learning accelerators such as GPUs and TPUs. Unlike deep neural networks (DNNs), which are built on enormous matrix multiplications structured for parallel computation, NeSy models use symbolic structures incorporating rule-based decision-making, graph traversal, and logical operations. These tasks incur heavy memory access overhead and computational non-uniformity, which render them inefficient on general AI accelerators. One such strategy is IMC, which sidesteps the data transfer bottleneck by computing directly in the memory itself (Khan et al. (2024)). IMC architectures are especially suited to executing decision trees, finite-state machines, and symbolic policies, all of which are typical elements in NeSy models. New studies show that symbolic models tuned for IMC can achieve comparable performances to deep learning models at significantly lower energy consumptions, as demonstrated through experiments with decision tree classifiers on IMC accelerators (Pedretti et al. (2021); Yin et al. (2021)). Also, new hardware architectures, such as Resistive RAM (ReRAM)-based IMC, Ferroelectric Field-Effect Transistors (FeFETs), and coupled oscillator arrays, offer novel approaches to advancing the implementation of tasks related to symbolic reasoning. These hardware options provide high-density storage of memory, parallelization of logic, and symbolic computation at low energy, making them highly compatible with NeSy workloads.

A unified hybrid computing paradigm that unifies neural accelerators with special-purpose symbolic processors is essential to developing NeSy AI. Neuromorphic chips and logic-based co-processors are viable options for handling tasks with high reasoning demand, while still maintaining

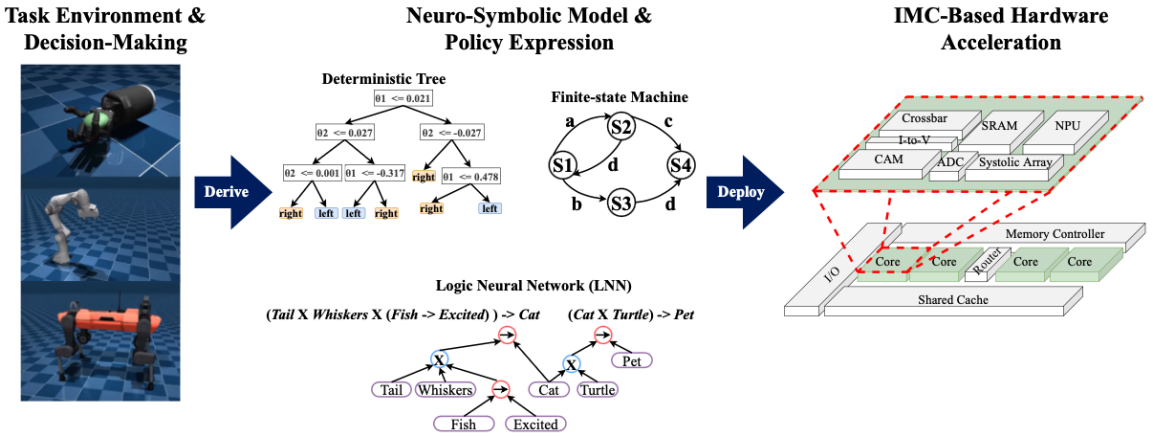

Figure 2: General workflow of accelerating Neuro-Symbolic policies using Compute-in-Memory (IMC) hardware. The left section illustrates diverse task environments requiring decision-making, such as robotics and autonomous control. The middle section represents different Neuro-Symbolic models and policy expressions, including decision trees, finite state machines (FSMs), and Logic Neural Network (LNN) structures. The right section depicts the deployment of these policies onto an IMC-based accelerator, leveraging crossbars, content-addressable memory (CAM), and specialized logic arrays for efficient symbolic reasoning and real-time execution.

low power consumption. As NeSy AI evolves, hardware-software co-design approaches will be essential for improving efficiency, scalability, and real-time execution in diverse applications such as robotics, autonomous systems, and edge AI. Future work needs to concentrate on enhancing heteregoneous architectures and developing hardware-efficient symbolic execution techniques to fully harness the power of NeSy AI. In the next section, we will talk about the symbolic policy generation pipeline used by us.

3. Proposed Methodology

The approach presented in this work focuses on the replacement of energy-hungry NN policies with symbolic policies that can be efficiently accelerated by leveraging specialized hardware. As shown in Figure. 2, a two-step process is followed: first, a NN policy is distilled into a symbolic policy using the VIPER algorithm (Bastani et al. (2018)), ensuring interpretable decision-making capabilities. Then, the symbolic policy is embedded into an IMC architecture, leveraging its parallel execution and memory-aware execution to enhance performance while reducing power consumption. This approach enables the use of NeSy models in low-power, real-time decision-making scenarios. In the following, we will discuss how to distill NN policy into symbolic policy and then deploy it on IMC.

3.1. Neural Network to Symbolic Policy

We use the VIPER algorithm proposed in Bastani et al. (2018). In this work, first, a NN policy is trained using the Deep Q-Learning algorithm (Van Hasselt et al. (2016)) for tasks with discrete action space. Subsequently, the DAGGER algorithm (Ross et al. (2011)) is used to imitate the NN

policy using the decision tree policy. Here, the decision tree policy is a classifier because of the discrete action space. During training, trajectories are generated from a mixture policy consisting of both NN policy and decision tree policy. As the training progresses, the contribution of NN policy is annealed to allow the decision tree policy to take over. This specific way of training ensures that the decision tree policy is able to perform even on the state trajectory that deviates from the trajectory generated by the NN policy. At the end of the training, we obtain a deterministic policy with branching of nodes dependent on checking a single boolean condition. The use of simple operation in the decision tree policy makes it amenable to IMC architecture.

3.2. IMC-Based Acceleration

IMC-based architectures provide a possible solution to accelerating NeSy models by eliminating data movement overhead and enabling O(1) complexity operations. Crossbar-based IMC is best suited for Type I operations such as addition and multiplication with vector-matrix multiplications for high-performance computing. CAM-enhanced IMC with O(1) complexity lookup enables fast symbolic rule lookups and Type II operations such as comparison-based logic ($<$, $>$, $=$) and hierarchical decision-making. Apart from these, Ternary Content-Addressable Memory (TCAM) goes one step further in enabling CAM's functionality with the support of multi-value symbolic matching, which finds application in probabilistic rule evaluation in NeSy AI. Moreover, Associative Processing Units (APUs) facilitate pattern-based symbolic reasoning and decrease dependency on deterministic rule tables while improving flexibility in symbolic policies (Fouda et al. (2022); Austin (1996)).

In order to effectively enhance symbolic decision policies, this study investigates the potential for deterministic trees to be executed on IMC accelerators through logic execution on Analog Content-Addressable Memory (aCAM). aCAMs support the parallel evaluation of several decision nodes, thereby condensing memory lookup latency to O(1), a key benefit for symbolic reasoning applications (Pedretti et al. (2021)). This method enables the complete execution of decision trees in memory, thereby minimizing costly data transfers and lowering energy consumption by a considerable amount (Yin et al. (2021)). At the circuit level, there is a discussion of how symbolic policies are implemented physically. Decision trees are encoded into lookup tables (LUTs) and then instantiated in content-addressable memory (CAM) architectures, thereby enabling state-based decisions to be executed quickly.

4. Case Study

The proposed IMC architecture demonstrates an order-of-magnitude improvement in latency and energy-efficiency compared to conventional CPU and GPU systems. The arrangement reduces data movement overhead and enhances real-time AI performance, making it extremely scalable to diverse NeSy models beyond the evaluated benchmarks.

4.1. Deterministic Policy Acceleration

We conducted experiments using DQN (Van Hasselt et al. (2016)) and VIPER (Bastani et al. (2018)) algorithm on four different control tasks with discrete action space: CartPole, Xor, TrafficIntersection, and Frozenlake8x8. For DQN, we use NN with 3 layers and for decision tree, we use decision tree classifier from Scikit Learn python package (Pedregosa et al. (2011)) with no limit on the depth.

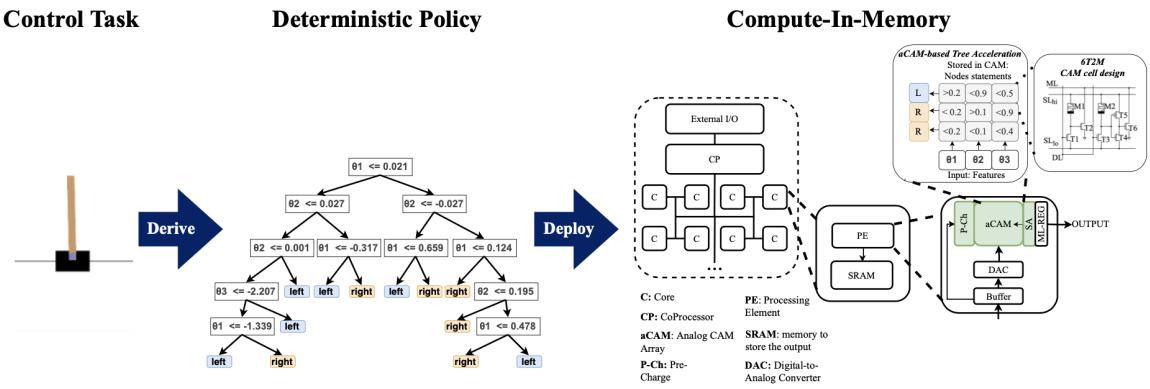

Figure 3: Illustration of the workflow from a reinforcement learning control task (CartPole) to the synthesis of a deterministic decision-tree policy, followed by its deployment onto a compute-in-memory (CIM) accelerator. The CIM architecture stores and evaluates the decision tree nodes within analog content-addressable memory (aCAM) arrays to achieve efficient inference performance.

Our decision tree policy includes Type II operations involving logical tests ($<$, $>$, $=$) and binary rule checks (see Figure. 3).

In hardware, Type II operations can be accelerated using Analog CAMs. Analog CAMs with 6T2M cells, comprising of six transistors (T1–T6) and two memristors (M1, M2), allow analog data storage, and range based comparison which support Type II operations in memory (Li et al. (2020)). We evaluate energy efficiency, latency, and throughput using HSPICE simulations. Python- and HSPICE-based simulations analyze aCAM performance, while CACTI modeling validates energy consumption at the peripheral and SRAM levels. Additionally, we synthesized key Verilog modules using Synopsys Design Compiler, obtaining realistic power estimates under standard-cell constraints. Each environment required a customized CAM array size and memory configuration, optimizing efficiency. Experimental results show that CAM-based processing achieves superior power efficiency with minimal peripheral overhead. The pre-charge, compare, and discharge operations facilitate fast symbolic rule evaluation, while memristor tuning at 1/64 precision (0.015 per step) enables 64-level storage, balancing hardware efficiency and accuracy. While real-world memristor variations may introduce minor fluctuations, our analysis assumes an ideal scenario to assess intrinsic system capabilities.

We benchmarked CPU (Intel Xeon E5-2687W v4) and GPU (NVIDIA TITAN RTX) inference performance across four environments (CartPole-v1, Xor, TrafficIntersection, FrozenLake8x8) using decision trees (`Scikit-learn`) and deep reinforcement learning models (`JAX/Flax`). Each model ran 1000 inference iterations, measuring latency and power via `nvidia-smi` (GPU) and `psutil.cpu_percent()` (CPU). Hardware specs were retrieved using `platform` and `nvidia-smi`. DQN used `jax.device_put()` for CPU/GPU inference, while decision trees ran on CPU. Trimmed mean (removing top/bottom 25%) was computed for latency and power.

Our results demonstrate that IMC-based NeSy models achieve $\sim$100$\times$ lower latency and over six orders of magnitude better energy efficiency than conventional CPU and GPU implementations (see Table. 2). IMC accelerators maintain competitive reward performance while drastically reducing power consumption, confirming their viability for high-speed, efficient decision-making in

NeSy AI. These findings highlight IMC's potential for energy-constrained applications, paving the way for NeSy AI in edge computing, robotics, and autonomous systems.

Table 2: Comparison of Execution Power, Throughput, and Energy Per Decision

Task	**Setup**	**Model**	**Process** (nm)	**Power** (mW)	**Throughput** (Dec/s)	**Energy** (nJ/Dec)
CartPole-v1	CPU	NN	14	2.936E+04	3.636E+03	8.075E+06
	CPU	NeSy	14	2.693E+04	1.783E+03	1.511E+07
	GPU	NN	12	1.074E+05	1.567E+03	6.852E+07
	IMC	NeSy	65	**1.063E+01**	**1.250E+05**	**8.503E+01**
Xor	CPU	NN	14	2.934E+04	3.636E+03	8.068E+06
	CPU	NeSy	14	3.140E+04	1.635E+03	1.933E+07
	GPU	NN	12	1.081E+05	1.593E+03	6.784E+07
	IMC	NeSy	65	**4.911E+00**	**1.250E+05**	**6.139E+01**
TrafficIntersection	CPU	NN	14	2.908E+04	3.448E+03	8.348E+06
	CPU	NeSy	14	2.928E+04	1.724E+03	1.698E+07
	GPU	NN	12	1.090E+05	1.564E+03	6.960E+07
	IMC	NeSy	65	**1.158E+01**	**1.250E+05**	**9.232E+01**
Frozenlake8x8	CPU	NN	14	2.921E+04	3.425E+03	8.473E+06
	CPU	NeSy	14	2.845E+04	1.698E+03	1.678E+07
	GPU	NN	12	1.061E+05	1.566E+03	6.774E+07
	IMC	NeSy	65	**6.457E+00**	**1.250E+05**	**8.072E+01**

4.2. Probabilistic Policy Acceleration

VIPER Bastani et al. (2018) allows us to distill NN into deterministic decision tree policy. However, for the decision tree to be applicable for more complex tasks, we need a stochastic variant as proposed in Qiu and Zhu (2022). The stochastic variant assigns probability for trees with different depth as well as to the evaluation of *if-and-else* condition. Effectively, the final output is a weighted average of different actions produced with different probabilities. Hence, we obtain a powerful policy compared to the deterministic decision tree policy. In this work, we didn't obtain the results by implementing the stochastic decision tree on IMC architecture. However, we will discuss how our work can be extended to incorporate stochastic trees.

Mapping stochastic decision trees to IMC requires specialized memory and rule-checking for probabilistic operation evaluation. Unlike deterministic trees, where decision nodes execute fixed boolean conditions, stochastic trees require sampling from probability distributions. This can be implemented using Ternary Content-Addressable Memory (TCAM), which supports multi-level state encoding, or by incorporating stochastic computing units (SCUs) within the IMC. CAM-based parallel lookup can efficiently retrieve probability-weighted actions, reducing the latency of probabilistic decision-making.

To support random sampling operations, we propose integrating low-power digital random number generators (RNGs) or memristor-based entropy sources directly within IMC. These RNGs can drive Monte Carlo sampling for probabilistic branching decisions in hardware, eliminating the need

Table 3: Comparison of Decision tree (NeSy) and DQN (NN) policies

Task	**Reward**		**Performance Drop**(%)
	NeSy (a)	NN (b)	
CartPole-v1	384.12	305.57	-25.7
Xor	979.93	980.28	0.003
TrafficIntersection	27.65	31.16	11.2
Frozenlake8x8	0.456	0.401	-13.7

Note: The performance drop is computed as $(NN\ Reward - NeSy\ Reward)/NN\ Reward \times 100$. *Negative values indicate NeSy outperforms NN.*

for external computation. Additionally, probabilistic rule tables can be preloaded into CAM arrays, enabling ultra-fast lookups with *O(1)* complexity.

Future IMC architectures could also leverage stochastic ferroelectric or phase-change memory devices, which inherently exhibit tunable probabilistic switching behaviors, making them ideal for implementing uncertainty-driven decision-making. By embedding stochastic trees within IMC, we can extend NeSy AI to handle complex, uncertain environments while maintaining energy efficiency and real-time execution.

5. Challenges and Prospects

Using VIPER (Bastani et al. (2018)) algorithm, we are able to show that we can obtain performance comparable to NN policy (see Table 3) on control task with discrete action space using deterministic decision tree and reap the benefits of computational efficiency (see Table 2) using IMC architecture. However, to obtain controllers for more complicated task, we need to use stochastic decision tree (Qiu and Zhu (2022)). The IMC architecture used in the current work doesn't support sampling from a probability distribution. Therefore, we were not able to demonstrate the performance on more complex control task. Although, we have briefly explained how to handle stochastic policies on hardware, there is still a need of concerted effort on extending the capabilities of IMC architecture to support probabilistic operations.

Further, in this paper, we discuss purely symbolic policy by distilling a NN-based policy. In order to reap the benefits of both NN policies and symbolic policies, the focus can be shifted to neuro-symbolic policies. For example, finite state-machines (FSM) with a small NN policy for each state could be used. The computational demands of FSM-based policies would be lower due to the usage of symbolic components and a small NN. However, to support the execution of FSM-based policy there is a need for hybrid hardware architecture that can support both IMC operation and the needs of small NN. Therefore, we need to focus on developments from both the algorithmic and hardware front.

As a future work from the algorithmic side, we would like to work on FSM-based policy by combining the contribution of Qiu and Zhu (2022) in utilizing small NN with symbolic policy and contribution of Inala et al. (2020) in learning an FSM. As an additional step toward accommodating both symbolic and NN models on the same platform, we envision developing a unified system-on-chip (SoC) for accelerating a variety of neuro-symbolic workloads. On the hardware side, our approach involves integrating circuits such as content-addressable memories (CAMs), crossbar arrays, and systolic arrays in a single computing substrate. CAMs can natively handle discrete lookups

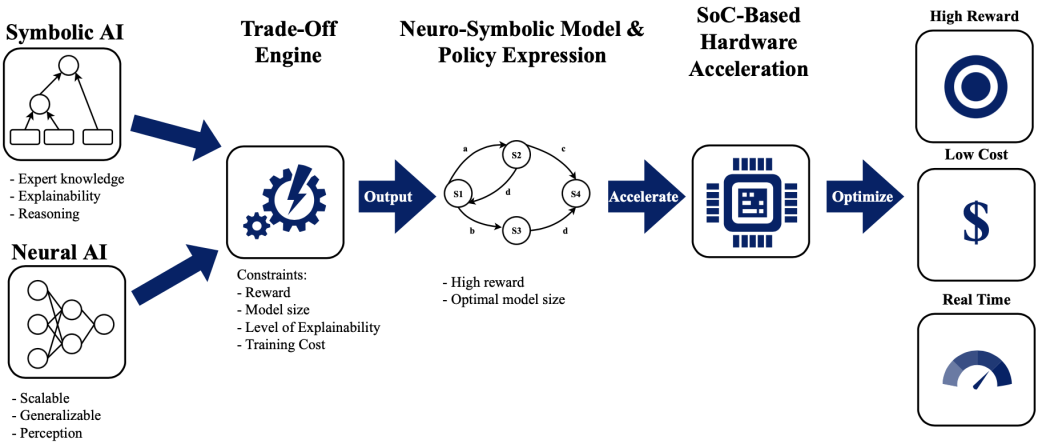

Figure 4: Trade-off between symbolic AI and neural AI components yield a neuro-symbolic policy, then mapped onto an SoC-based hardware accelerator composed of specialized modules for both logical and arithmetic operations. This hardware-software co-design targets high reward, low energy cost, and robust real-time performance in a unified framework.

for symbolic models like decision trees or state machines, while crossbars and systolic arrays can be leveraged for neural operations. By co-locating these specialized compute blocks, we aim to minimize data movement and optimize a holistic metric that balances energy cost, chip size, and reward performance. This fully integrated design would permit a range of policy expressions—from purely symbolic to hybrid neuro-symbolic models—to run efficiently under strict power or latency budgets, making it suitable for edge scenarios and real-time control.

Ultimately, our plan is to develop a complete methodology, as illustrated in Figure 4, which spans the compilation of neuro-symbolic models to hardware, the design of custom SoC modules for logic and arithmetic, and the tuning of final applications for maximum energy-efficiency and interpretability. By refining each link in this chain, we seek to deliver a robust framework that provides a trade-off between symbolic AI's inherent explainability and neural AI's adaptiveness that collectively thrive on a specialized, compact, and low-power substrate. We believe this hardware-algorithm co-design strategy will empower next-generation AI applications, enabling real-time, interpretable decision-making at scale.

6. Conclusion

This vision paper advocates for a paradigm shift toward In-Memory Computing (IMC)-based acceleration for Neuro-Symbolic AI, enabling specialized hardware to efficiently execute symbolic policies with *O(1)* complexity decision-making. Our analysis demonstrates that IMC-based symbolic policies achieve up to 100× lower latency and six orders of magnitude better energy efficiency compared to traditional CPU and GPU execution. By extending IMC architectures to support probabilistic symbolic policies, we open the door for AI systems capable of real-time reasoning under uncertainty. This work highlights the urgent need for co-designing AI models with specialized hardware, paving the way for next-generation energy-efficient, real-time, and interpretable AI that can be deployed in edge computing, robotics, and autonomous systems.

Acknowledgments

Acknowledgements text.

References

Jim Austin. Distributed associative memories for high-speed symbolic reasoning. *Fuzzy Sets and Systems*, 82(2):223–233, 1996. ISSN 0165-0114. doi: https://doi.org/10.1016/0165-0114(95)00258-8. URL https://www.sciencedirect.com/science/article/pii/0165011495002588. Connectionist and Hybrid Connectionist Systems for Approximate Reasoning.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems*, 31, 2018.

Bikram Pratim Bhuyan, Amar Ramdane-Cherif, Ravi Tomar, and T. P. Singh. Neuro-symbolic artificial intelligence: a survey. *Neural Comput. Appl.*, 36(21):12809–12844, June 2024. ISSN 0941-0643. doi: 10.1007/s00521-024-09960-z. URL https://doi.org/10.1007/s00521-024-09960-z.

Tales H Carvalho, Kenneth Tjhia, and Levi HS Lelis. Reclaiming the source of programmatic policies: Programmatic versus latent spaces. *arXiv preprint arXiv:2410.12166*, 2024.

Mohammed E. Fouda, Hasan Erdem Yantır, Ahmed M. Eltawil, and Fadi Kurdahi. In-memory associative processors: Tutorial, potential, and challenges. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(6):2641–2647, 2022. doi: 10.1109/TCSII.2022.3170468.

Kyle Hamilton, Aparna Nayak, Bojan Božić, and Luca Longo. Is neuro-symbolic ai meeting its promises in natural language processing? a structured review. *Semantic Web*, 15(4):1265–1306, October 2024. ISSN 1570-0844. doi: 10.3233/sw-223228. URL http://dx.doi.org/10.3233/SW-223228.

Jeevana Priya Inala, Osbert Bastani, Zenna Tavares, and Armando Solar-Lezama. Synthesizing programmatic policies that inductively generalize. In *8th International Conference on Learning Representations*, 2020.

Asif Ali Khan, João Paulo C. De Lima, Hamid Farzaneh, and Jeronimo Castrillon. The landscape of compute-near-memory and compute-in-memory: A research and commercial overview, 2024. URL https://arxiv.org/abs/2401.14428.

Can Li, Catherine E. Graves, Xia Sheng, Darrin Miller, Martin Foltin, Giacomo Pedretti, and John Paul Strachan. Analog content-addressable memories with memristors. *Nature Communications*, 11(1), April 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-15254-4. URL http://dx.doi.org/10.1038/s41467-020-15254-4.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Giacomo Pedretti, Catherine E. Graves, Sergey Serebryakov, Ruibin Mao, Xia Sheng, Martin Foltin, Can Li, and John Paul Strachan. Tree-based machine learning performed in-memory with memristive analog cam. *Nature Communications*, 12(1), October 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-25873-0. URL http://dx.doi.org/10.1038/s41467-021-25873-0.

Wenjie Qiu and He Zhu. Programmatic reinforcement learning without oracles. In *The Tenth International Conference on Learning Representations*, 2022.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

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

Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1855–1865. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/silva20a.html.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

Abhinav Verma, Hoang Le, Yisong Yue, and Swarat Chaudhuri. Imitation-projected programmatic reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Daniël Vos and Sicco Verwer. Optimizing interpretable decision tree policies for reinforcement learning. *arXiv preprint arXiv:2408.11632*, 2024.

Xunzhao Yin, Franz Müller, Ann Franchesca Laguna, Chao Li, Wenwen Ye, Qingrong Huang, Qinming Zhang, Zhiguo Shi, Maximilian Lederer, Nellie Laleni, Shan Deng, Zijian Zhao, Michael Niemier, Xiaobo Sharon Hu, Cheng Zhuo, Thomas Kämpfe, and Kai Ni. Deep random forest with ferroelectric analog content addressable memory, 2021. URL https://arxiv.org/abs/2110.02495.