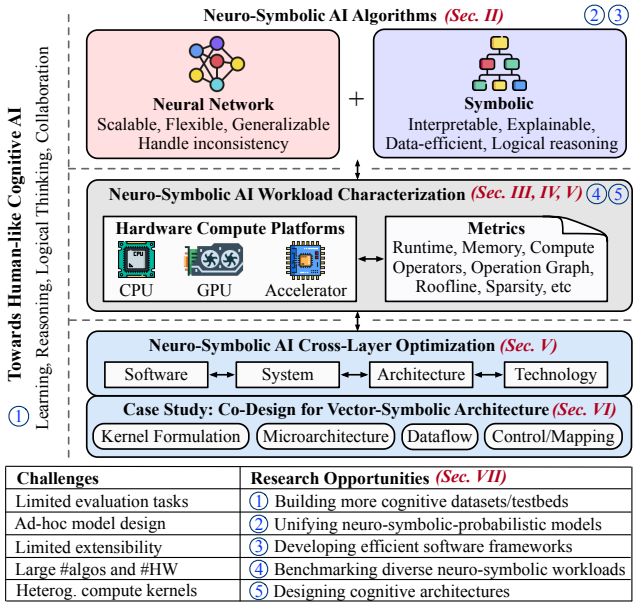
Towards Efficient Neuro-Symbolic AI: From Workload Characterization to Hardware Architecture

Zishen Wan[1], Che-Kai Liu[1], Hanchen Yang[1], Ritik Raj[1], Chaojian Li[1], Haoran You[1], Yonggan Fu[1], Cheng Wan[1], Sixu Li[1], Youbin Kim[2], Ananda Samajdar[3], Yingyan (Celine) Lin[1], Mohamed Ibrahim[1,2], Jan M. Rabaey[2], Tushar Krishna[1], and Arijit Raychowdhury[1]

[1]*Georgia Institute of Technology* [2]*University of California, Berkeley* [3]*IBM Research*

***Abstract*—The remarkable advancements in artificial intelligence (AI), primarily driven by deep neural networks, are facing challenges surrounding unsustainable computational trajectories, limited robustness, and a lack of explainability. To develop next-generation cognitive AI systems, neuro-symbolic AI emerges as a promising paradigm, fusing neural and symbolic approaches to enhance interpretability, robustness, and trustworthiness, while facilitating learning from much less data. Recent neuro-symbolic systems have demonstrated great potential in collaborative human-AI scenarios with reasoning and cognitive capabilities. In this paper, we aim to understand the workload characteristics and potential architectures for neuro-symbolic AI. We first systematically categorize neuro-symbolic AI algorithms, and then experimentally evaluate and analyze them in terms of runtime, memory, computational operators, sparsity, and system characteristics on CPUs, GPUs, and edge SoCs. Our studies reveal that neuro-symbolic models suffer from inefficiencies on off-the-shelf hardware, due to the memory-bound nature of vector-symbolic and logical operations, complex flow control, data dependencies, sparsity variations, and limited scalability. Based on profiling insights, we suggest cross-layer optimization solutions and present a hardware acceleration case study for vector-symbolic architecture to improve the performance, efficiency, and scalability of neuro-symbolic computing. Finally, we discuss the challenges and potential future directions of neuro-symbolic AI from both system and architectural perspectives.**

***Index Terms*—cognitive AI, neuro-symbolic AI, workload characterization, performance analysis, domain-specific architecture**



Challenges	Research Opportunities *(Sec. VII)*
Limited evaluation tasks	① Building more cognitive datasets/testbeds
Ad-hoc model design	② Unifying neuro-symbolic-probabilistic models
Limited extensibility	③ Developing efficient software frameworks
Large #algos and #HW	④ Benchmarking diverse neuro-symbolic workloads
Heterog. compute kernels	⑤ Designing cognitive architectures

Fig. 1: Overview of neuro-symbolic AI systems, workload characterizations, optimization solutions, challenges, and research opportunities in improving the performance of next-generation cognitive AI.

I. Introduction

The remarkable advancements in AI have had a profound impact on our society. These advancements are primarily driven by deep neural networks and a virtuous cycle involving large networks, extensive datasets, and augmented computing power. As we reap the benefits of this success, there is growing evidence that continuing our current trajectory may not be viable for realizing AI's full potential. First, the escalating computational requirements and energy consumption associated with AI are on an unsustainable trajectory [1], threatening to reach a level that could stifle innovation by restricting it to fewer organizations. Second, the lack of robustness and explainability remains a significant challenge, likely due to inherent limitations in current learning methodologies [2], [3]. Third, contemporary AI systems often operate in isolation with limited collaboration among humans and other AI agents. Hence, it is imperative to develop next-generation AI paradigms that address the growing demand for enhanced efficiency, explainability, and trust in AI systems.

Neuro-symbolic AI [4] represents an emerging AI paradigm that integrates the neural and symbolic approaches with probabilistic representations to enhance explainability, robustness and facilitates learning from much less data in AI (Fig. 1). Neural methods are highly effective in extracting complex features from data for vision and language tasks. On the other hand, symbolic methods enhance explainability and reduce the dependence on extensive training data by incorporating established models of the physical world, and probabilistic representations enable cognitive systems to more effectively handle uncertainty, resulting in improved robustness under unstructured conditions. The synergistic fusion of neural and symbolic methods positions neuro-symbolic AI as a promising paradigm capable of ushering in the third wave of AI [5], [6].

Neuro-symbolic AI promises possibilities for systems that acquire human-like communication and reasoning capabilities, enabling them to recognize, classify, and adapt to new situations autonomously. For example, neuro-vector-symbolic architecture [7] is able to reach 98.8% accuracy on spatial-temporal reasoning tasks, greatly surpassing human performance (84.4%), neuro-only ResNet (53.4%) and GPT-4 performance (89.0%). In addition to its superior performance in vision and language [8], neuro-symbolic AI holds significant potential for enhancing explainability and trustworthiness of collaborative human-AI applications [9]. These applications include collaborative robotics, mixed-reality systems, and

human-AI interactions, where robots can seamlessly interact with humans in environments, agents can reason and make decisions in an explainable manner, and intelligence is pervasively embedded and untethered from the cloud.

Despite the promising algorithmic performance, the higher memory intensity, greater kernel heterogeneity, and access pattern irregularity of neuro-symbolic computing lead to an increasing divergence from the current hardware roadmap that largely optimizes for matrix multiplication and convolution [10]–[14] and leads to severe inefficiencies and underutilization of hardware. Therefore, understanding its computational and memory demands is essential for efficient processing on both general-purpose and custom hardware.

Our goal in this work is to quantify the workload characteristics and potential system architecture for neuro-symbolic AI. Built on our work [4], [15], we first conduct a systematic review and categorize state-of-the-art neuro-symbolic AI workloads in a structured manner (Sec. II). We then characterize seven representative neuro-symbolic workloads on general-purpose and edge platforms, analyzing their runtime, memory, compute operators, operation graph, hardware utilization, and sparsity characteristics (Secs. III, IV, V). Our workload characterization reveals several key observations and insights, including the following:

- Neuro-symbolic AI models typically exhibit high latency compared to neural models, prohibiting them from real-time applications.
- The neural components mainly consist of MatMul and Convs, while the symbolic components are dominated by vector/element-wise and logical operations. The low ALU utilization, low cache hit rates, and high volume of data movement of symbolic operations make them inefficient on CPUs/GPUs and may result in system bottlenecks.
- The neural workloads are compute-bounded while the symbolic workloads are typically memory-bounded and face potential scalability issues.
- The symbolic operations may depend on neural results or need to compile into the neural structure, thus lying on the critical path of end-to-end neuro-symbolic systems.
- Some neural and vector-symbolic components demonstrate a high level of unstructured sparsity with variations under different task scenarios and attributes.

Inspired by our workload profiling insights, we recommend several cross-layer software and hardware optimization solutions to improve the efficiency and scalability of neuro-symbolic systems (Sec. V). Specifically, we leverage vector-symbolic architecture as a case study and present a hardware acceleration methodology, including kernel formulation, microarchitecture, dataflow, and control schemes (Sec. VI). Finally, we explore the research opportunities in neuro-symbolic computing and share our outlook on the road ahead (Sec. VII).

To the best of our knowledge, this is one of the *first* works to characterize neuro-symbolic computing from both system and architectural perspectives, and enable its efficient and scalable execution. We aim to inspire the design of next-generation cognitive computing systems through synergistic advancements in neuro-symbolic algorithms, systems, architecture, and algorithm-hardware co-design.

II. Neuro-Symbolic AI Algorithms

In this section, we systematically review and categorize the recent research progress in neuro-symbolic AI algorithms.

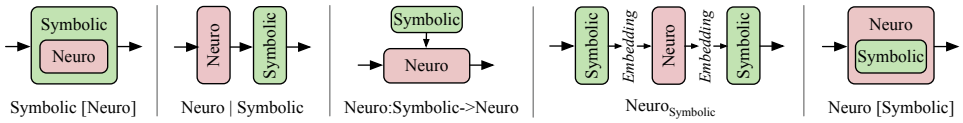
Overview. Neuro-symbolic AI represents an interdisciplinary approach that synergistically combines symbolic reasoning with neural network (NN) learning to create intelligent systems, leveraging the complementary strengths of both to enhance the accuracy and interpretability of the resulting models. Given that neuro-symbolic algorithms incorporate symbolic and neural components, various paradigms can be categorized based on how these components are integrated into a cohesive system. Inspired by Henry Kautz's taxonomy [31], we systematically categorize these algorithms into five paradigms (Tab. I). We elaborate on each of these paradigms below. Additionally, Tab. II provides examples of several underlying operations based on the categorization in Tab. I.

Symbolic[Neuro] refers to an intelligent system that empowers symbolic reasoning with the statistical learning capabilities of NNs. These systems typically consist of a comprehensive symbolic problem solver that includes loosely-coupled neural subroutines for statistical learning. Examples include DeepMind's AlphaGo [16] and AlphaZero [32], which use Monte-Carlo Tree Search (MCTS) as the symbolic solver and NN state estimators for learning statistical patterns.

Neuro|Symbolic refers to a hybrid system that combines neural and symbolic components in a pipeline, where each component typically specializes in complementary tasks. To the best of our knowledge, the majority of neuro-symbolic algorithms fall into this category. For example, IBM's neuro-vector-symbolic architecture (NVSA) [7] uses an NN as the perception frontend for semantic parsing and a symbolic reasoner as the backend for probabilistic abductive reasoning on the RAVEN [33] and I-RAVEN [34] datasets. Probabilistic abduction and execution (PrAE) learner [22] adopts a similar approach where the difference lies in features are first projected to high-dimensional vectors in NVSA, whereas PrAE utilizes the original features directly as the NN's input. Other examples include vector symbolic architecture-based image-to-image translation (VSAIT) [21], neuro-probabilistic soft logic (NeuPSL) [17], neural probabilistic logic programming (DeepProbLog) [35], neuro-answer set programming (NeurASP) [18], neural symbolic dynamic reasoning [36], neural symbolic concept learner (NSCL) [8], abductive learning (ABL) [19], and neuro-symbolic visual question answering (NSVQA) [20] on the CLEVRER dataset [36].

Neuro:Symbolic→Neuro approach incorporates symbolic rules into NNs to guide the learning process, where symbolic knowledge is compiled into the structure of neural models for enhancing the model interpretability. For instance, logical NNs (LNNs) [23] encode knowledge or domain expertise as symbolic rules (first-order logic or fuzzy logic) that act as constraints on the NN output. Other examples include the application of deep learning for symbolic mathematics [24] and differentiable inductive logic programming (ILP) [25].

Neuro$_{\text{Symbolic}}$ is a type of hybrid approach that combines symbolic logic rules with NNs. It involves mapping symbolic logic rules onto embeddings that serve as soft constraints or



Category	Category Description	Neuro-Symbolic Algorithm	Underlying Operation	If Vector
Symbolic[Neuro]	End-to-end **symbolic** system that uses **neural** models internally as a subroutine	**AlphaGo** [16]	NN, MCTS	Vector
Neuro|Symbolic	Pipelined system that integrates **neural** and **symbolic** components where each component specializes in complementary tasks within the whole system	**NVSA** [7]	NN, mul, add, circular conv.	Vector
		NeuPSL [17]	NN, fuzzy logic	Vector
		NSCL [8]	NN, add, mul, div, log	Vector
		NeurASP [18]	NN, logic rules	Non-Vector
		ABL [19]	NN, logic rules	Non-Vector
		NSVQA [20]	NN, pre-defined objects	Non-Vector
		VSAIT [21]	NN, binding/unbinding	Vector
		PrAE [22]	NN, logic rules, prob. abduction	Vector
Neuro:Symbolic→Neuro	End-to-end **neural** system that compiles **symbolic** knowledge externally	**LNN** [23]	NN, fuzzy logic	Vector
		Symbolic Math [24]	NN	Vector
		Differentiable ILP [25]	NN, fuzzy logic	Vector
Neuro$_{\text{Symbolic}}$	Pipelined system that maps **symbolic** first-order logic onto embeddings serving as soft constraints or regularizers for **neural** model	**LTN** [26]	NN, fuzzy logic	Vector
		DON [27]	NN	Vector
Neuro[Symbolic]	End-to-end **neural** system that uses **symbolic** models internally as a subroutine	**GNN+attention** [28]	NN, SpMM, SDDMM	Vector
		ZeroC [29]	NN (energy-based model, graph)	Vector
		NLM [30]	NN, permutation	Vector

TABLE I: Review of recent neuro-symbolic AI algorithms into five categories, with their underlying operations and vector formats.

TABLE II: Enumeration of the underlying operations based on Tab. I.

Underlying Operations	Examples
Fuzzy logic (LTN)	$F = \forall x(isCarnivor(s)) \rightarrow (isMammal(x))$ $\{isCarnivor(s)\text{:}[0,1],\ isMammal(x)\text{:}[1,0]\} \rightarrow F = [1,0]$
Mul, Add, and Circular Conv. (NVSA)	$X_i \in \{+1,-1\}^d \rightarrow (X_i \cdot X_j)/(X_i + X_j)$
Logic rules (ABL)	Domain: $animal(dog).carnivore(dog).mammal(dog)$ Logical formula: $mammal(x) \wedge carnivore(x)$ ABL: $hypos(x) : -animal(x), mammal(x), carnivore(x)$
Pre-defined objects (NSVQA)	`equal_color`: $(entry, entry) \rightarrow Boolean$ `equal_integer`: $(number, number) \rightarrow Boolean$

regularizers on the NN's loss function. Logical tensor networks (LTNs) [26], for instance, use logical formulas to define constraints on the tensor representations, which have proven successful in knowledge graph completion tasks. These tasks aim to predict missing facts or relationships between entities. Other examples of this approach include deep ontology networks (DONs) [27] and tensorization methods [37]. As inference is still governed by NNs, it remains a research question whether this approach will compromise interpretability.

Neuro[Symbolic] refers to a system that empowers NNs with the explainability and robustness of symbolic reasoning. Unlike **Symbolic[Neuro]**, where symbolic reasoning is used to guide the neural model learning process, in **Neuro[Symbolic]**, the neural model incorporates symbolic reasoning by paying attention to a specific symbolic at certain conditions. For instance, graph neural networks (GNNs) are adopted for representing symbolic expressions when endowed with attention mechanisms [28]. In particular, this attention mechanism can be leveraged to incorporate symbolic rules into GNN models, enabling selective attention to pertinent symbolic information in the graph. Other examples include neural logic machines (NLM) [30] and Zero-shot concept recognition and acquisition (ZeroC) [29]. ZeroC leverages the graph representation where the constituent concept models are represented as nodes and their relations are represented by edges.

Each neuro-symbolic category reflects different kernel operators and data dependencies. *Therefore, this paper takes one of the first steps towards understanding its computing characteristics and aims to serve as a cornerstone for the design and deployment of future neuro-symbolic systems.*

III. Representative Neuro-Symbolic Models

This section presents selected widely-used neuro-symbolic AI workloads as representative ones for our analysis. We consider them representative because they are diverse in terms of applications, model structures, and computational patterns.

A. Model Overview.

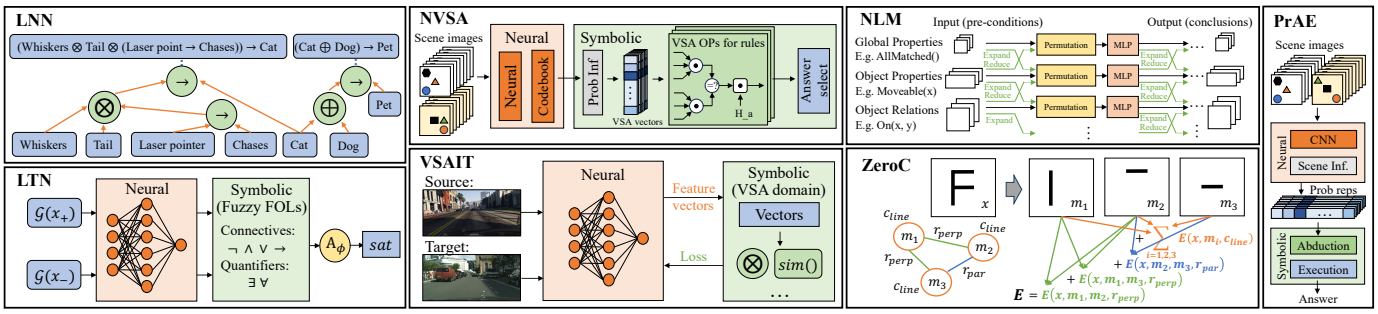
We select seven neuro-symbolic AI models for profiling analysis (Tab. III): LNN on logic program tasks [23], LTN on querying and reasoning tasks [26], NVSA [7] on the Raven's Progressive Matrices task [33], NLM on relational reasoning and decision making tasks [30], VSAIT on unpaired image-to-image translation tasks [21], ZeroC on cross-domain classification and detection tasks [29], and PrAE on spatial-temporal reasoning tasks [22]. These selected workloads represent Neuro:Symbolic→Neuro, Neuro$_{\text{Symbolic}}$, Neuro|Symbolic, and Neuro[Symbolic] systems (Sec. II), respectively. Interested readers could refer to their references for more details.

B. Logical Neural Network (LNN)

LNN is a neuro-symbolic framework that integrates neural learning with symbolic logic, enabling direct interpretability, domain knowledge utilization, and robust problem-solving [23]. LNNs map neurons to logical formula elements, using parameterized functions to represent logical connectives (e.g., $\wedge, \vee$) with constraints to preserve logical behavior. By combining facts and rules within a neural framework, LNNs use weighted real-valued logics via Łukasiewicz logic [26]. Compared to neural models, LNNs offer superior logical expressivity, tolerance to incomplete knowledge, and general task applicability, excelling in theorem proving with compositional, modular structures.

C. Logical Tensor Network (LTN)

LTN is a neuro-symbolic framework for querying, learning, and reasoning with data and abstract knowledge using fuzzy first-order logic (FOL) [26]. LTN grounds FOL elements in data using neural graphs and fuzzy logic, transforming



Representative Neuro-Symbolic AI Workloads		Logic Neural Network [23]	Logic Tensor Network [26]	Neuro-Vector-Symbolic Architecture [7]	Neural Logic Machine [30]	Vector Symbolic Architecture Image2Image Translation [21]	Zero-shot Concept Recognition and Acquisition [29]	Probabilistic Abduction and Execution [22]
Abbreviation		LNN	LTN	NVSA	NLM	VSAIT	ZeroC	PrAE
Neuro-Symbolic Category		Neuro:Symbolic→Neuro	$\text{Neuro}_{\text{Symbolic}}$	Neuro|Symbolic	Neuro[Symbolic]	Neuro|Symbolic	Neuro[Symbolic]	Neuro|Symbolic
Learning Approach		Supervised	Supervised/Unsupervised	Supervised/Unsupervised	Supervised/Unsupervised	Supervised	Supervised	Supervised/Unsupervised
Deployment Scenario	Application	Learning and reasoning, Full theorem prover	Querying, learning, reasoning (relational and embedding learning, query answering)	Fluid intelligence, Abstract reasoning	Relational reasoning, Decision making	Unpaired image-to-image translation	Cross-domain classification and detection, Concept acquisition	Fluid intelligence, Spatial-temporal reasoning
	Advantage vs. Neural Model	Higher interoperability, resilience to incomplete knowledge, generalization	Higher data efficiency, comprehensibility, out-of-distribution generalization	Higher joint representations efficiency, abstract reasoning capability, transparency	Higher generalization, logic reasoning, deduction, explainability capability	Address semantic flipping and hallucinations issue in unpaired image translation tasks	Higher generalization, concept acquisition and recognition, compositionality capability	Higher generalization, transparency, interpretability, and robustness
	Dataset	LUBM benchmark [38], TPTP benchmark [39]	UCI [40], Leptograpsus crabs [41], DeepProbLog [42]	RAVEN [33], I-RAVEN [34], PGM [43]	Family graph reasoning, sorting, path finding [44]	GTA [45], Cityscapes [46], Google Maps dataset [47]	Abstraction reasoning [48], Hierarchical-concept corpus [49]	RAVEN [33], I-RAVEN [34], PGM [43]
Computation Pattern	Datatype	FP32	FP32	FP32	FP32	FP32	INT64	FP32
	Neural	Graph	MLP	ConvNet	Sequential tensor	ConvNet	Energy-based network	ConvNet
	Symbolic	FOL/Logical operation	FOL/Logical operation	VSA/Vector operation	FOL/Logical operation	VSA/Vector operation	Graph, vector operation	VSA/Vector operation

TABLE III: Selected neuro-symbolic AI workloads for analysis, representing a diverse of categories, applications, and computational patterns.

connectives into real values and interpreting quantifiers via approximate aggregations [26]. The network computes truth degrees using embedded tensor representations. Compared to neural models, LTN enhances explainability, data efficiency, and out-of-distribution generalization by expressing knowledge through logical axioms over data.

D. Neuro-Vector-Symbolic Architecture (NVSA)

NVSA is a neuro-symbolic architecture for abstract reasoning, combining neural visual perception and vector-symbolic probabilistic reasoning to improve abduction reasoning efficiency [7]. NVSA uses holographic distributed representations to co-design visual perception and probabilistic reasoning, enabling perceptual representations and symbolic rule processing for accurate Raven's progressive matrices (RPM) [50], [51] test performance. Compared to neural models, NVSA overcomes the binding problem and superposition catastrophe, achieving superior accuracy in RPM tests and even surpassing human performance.

E. Neural Logic Machine (NLM)

NLM is a neuro-symbolic architecture for inductive learning and logical reasoning, combining neural networks as function approximators with logic programming for symbolic processing [30]. NLM approximates logic operations using neural networks and implements logic quantifiers through neural module wiring. Its multi-layer structure deduces object relations, forming higher abstractions with increased layers. Compared to neural models, NLM excels in relational reasoning and decision-making, generalizing well from small-scale to large-scale tasks, outperforming traditional neural networks and logic programming.

F. Vector Symbolic Architecture-Based Image-to-Image Translation (VSAIT)

VSAIT addresses semantic flipping in image translation between domains with large distribution gaps, leveraging vector-symbolic architecture for photorealism and robustness [21]. VSAIT learns invertible mappings in hypervector space, ensuring consistency between source and translated images while encoding features into random vector-symbolic hyperspace. Compared to neural models, VSAIT ensures robustness to semantic flipping and significantly reduces image hallucinations observed for unpaired image translation between domains with large gaps.

G. Zero-Shot Concept Recognition and Acquisition (ZeroC)

ZeroC is a neuro-symbolic architecture that recognizes and acquires novel concepts in a zero-shot manner by leveraging symbolic graph structures [29]. ZeroC uses graphs and energy-based models to represent concepts and relations, allowing hierarchical concept models to generalize across domains during inference. Compared to neural models, ZeroC excels in zero-shot concept recognition, surpassing neural models in tasks requiring novel concept learning without extensive examples.

H. Probabilistic Abduction and Execution (PrAE) Learner

PrAE is a neuro-symbolic learner for spatial-temporal cognitive reasoning, centered on probabilistic abduction and execution of scene representations [22]. PrAE combines neural visual perception with symbolic reasoning to predict object attributes and generate probabilistic scene representations, inferring hidden rules for systematic generalization. Compared to neural models, PrAE outperforms them in spatial-temporal reasoning, offering transparency, interpretability, and human-level generalizability.

IV. Workload Characterization Methodology

This section presents our neuro-symbolic AI workload profiling methodology (Sec. IV-A) and operator characterization taxonomy (Sec. IV-B) that will be leveraged in Sec. V.

A. Workload Profiling Methodology

We first conduct function-level profiling to capture statistics such as runtime, memory, invocation counts, tensor sizes, and sparsity of each model, by leveraging the built-in PyTorch Profiler. We also perform post-processing to partition the characterization results into various operation categories. The experiments are conducted on a system with Intel Xeon Silver 4114 CPU and Nvidia RTX 2080 Ti GPU (250W), as well as edge SoCs such as Xavier NX (20W) and Jetson TX2 (15W).

B. Workload Characterization Taxonomy

On top of function-level profiling, we further conduct compute operator-level profiling for further analysis. We classify each neural and symbolic workload of the LNN, LTN, NVSA, NLM, VSAIT, ZeroC, and PrAE neuro-symbolic models into six operator categories: convolution, matrix multiplication (MatMul), vector/element-wise tensor operation, data transformation, data movement, and others [52].

Convolution: refers to operations involving overlaying a matrix (kernel) onto another matrix (input) and computing the sum of element-wise products. This process is slid across the entire matrix and transforms the data. Convolution is common in neural networks and leads to high operational intensity.

Matrix Multiplication: refers to general matrix multiplication (GEMM) with two matrices, either dense or sparse. Fully-connected layers in neural networks use GEMM as their primary mathematical operation. Multiplication of large, dense matrices is typically computationally intensive but highly parallelizable. There is typically a trade-off between the generality of the sparsity and the overhead of hardware optimization. Sparse matrix multiplication requires efficient mechanisms to perform lookups into the tables of non-zero values.

Vector/Element-wise Tensor Operation: refers to operations performed element-wise on tensors (generalized matrices, vectors, and higher-dimensional arrays), including addition, subtraction, multiplication, and division, applied between two tensors element by element, as well as activation, normalization, and relational operations in neuron models.

Data Transformation: refers to operations that reshape or subsample data, including matrix transposes, tensor reordering, masked selection, and coalescing which is a process in which duplicate entries for the same coordinates in a sparse matrix are eliminated by summing their associated values.

Data Movement: refers to data transferring from memory-to-compute, host-to-device, and device-to-host, as well as operations such as tensor duplication and assignment.

Others: refers to operations such as fuzzy first of logic and logical rules that are utilized in some symbolic AI workloads.

V. Workload Characterization Results

This section analyzes the performance characteristics of representative neuro-symbolic workloads and discusses their runtime and scalability (Sec. V-A), compute operators (Sec. V-B), memory usage (Sec. V-C), operation graph (Sec. V-D), hardware utilization (Sec. V-E), and sparsity (Sec. V-F).

A. Compute Latency Analysis

End-to-end latency breakdown. We first characterize the end-to-end latency of representative neuro-symbolic AI workloads (Fig. 2). We can observe that (1) Compared to neural workloads, symbolic workloads are not negligible in computing latency and may become a system bottleneck. For example, the neural (symbolic) workloads account for 54.6% (45.4%), 48.0% (52.0%), 7.9% (92.1%), 39.4% (60.6%), 16.3% (83.7%), 73.2% (26.8%), and 19.5% (80.5%) runtime of LNN, LTN, NVSA, NLM, VSAIT, ZeroC, and PrAE models, respectively (Fig. 2a). Notably, the symbolic workload dominates the NVSA's runtime, predominately due to the sequential and computational-intensive rule detection during the involved reasoning procedure. (2) The real-time performance cannot be satisfied, e.g., RTX 2080Ti GPU takes 380 s and TX2 takes 7507 s for RPM task in NVSA (Fig. 2b). Even if more computing resources are available to reduce neural inference time, the significant overhead of vector-symbolic-based reasoning still prohibits real-time execution. (3) The symbolic operations may not be well accelerated by GPU. For example, symbolic counts for 92.1% of total NVSA inference time while its floating-point operations (FLOPS) count for only 19% of total FLOPS, indicating inefficient computation.

Takeaway 1: *Neuro-symbolic AI models typically exhibit high latency compared to neural models, prohibiting them from real-time applications. Symbolic operations are processed inefficiently on CPU/GPUs and may result in system bottlenecks.*

End-to-end latency scalability. We evaluate the end-to-end runtime across various task sizes and complexities, as shown in Fig. 2c of RPM task for NVSA. We can observe that (1) The neural vs. symbolic runtime proportion remains relatively stable across various task sizes. For example, when task size increases from 2×2 to 3×3, the symbolic runtime slightly changes from 91.59% to 87.35%. (2) The total runtime increases quadratically with task size evolving. For example, the total runtime increases 5.02× in the above case, indicating the potential scalability bottleneck of neuro-symbolic models.

Takeaway 2: *The neural and symbolic components runtime ratio remains relatively stable while total latency explodes with the task complexity evolving. The potential scalability bottleneck calls for highly scalable and efficient architecture.*

Recommendation 1: *Optimization on neuro-symbolic workloads from algorithm-system-hardware cross-layer perspectives is highly desirable for achieving real-time, efficient and scalable cognitive systems.*

B. Compute Operator Analysis

Fig. 3a partitions the neural and symbolic workloads of the LNN, LTN, NVSA, NLM, VSAIT, ZeroC, and PrAE workloads into six operator categories (Sec. IV-B) with runtime latency breakdown. We make the following observations:

Neural Workload Analysis. The neural workload is dominated by the MatMul and activation operations. LTN (neuro) is dominated by MatMul due to its heavy MLP components, while NVSA, VSAIT, and PrAE's (neuro) majority runtime is on MatMul and convolution because they adopt the neural network as the perception backbone for feature extraction. By

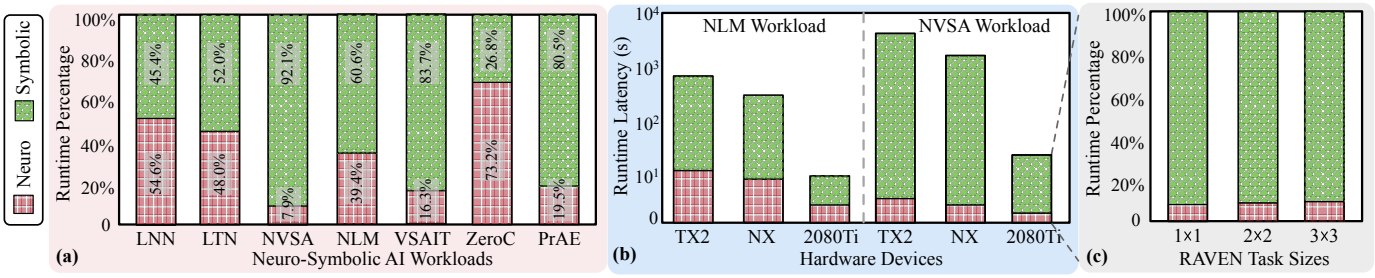


Fig. 2: Neural and symbolic runtime latency characterization. **(a)** Benchmark seven representative neuro-symbolic workloads (LNN, LTN, NVSA, NLM, VSAIT, ZeroC, PrAE) on the CPU+GPU system, showing symbolic may serve as system bottleneck. **(b)** Benchmark NVSA and NLM workloads on Jetson TX2, Xavier NX, and RTX GPU, showing that real-time performance cannot be satisfied. **(c)** Benchmark NVSA workload on various RPM task sizes on RTX GPU, indicating the potential scalability problem and consistent symbolic bottleneck.

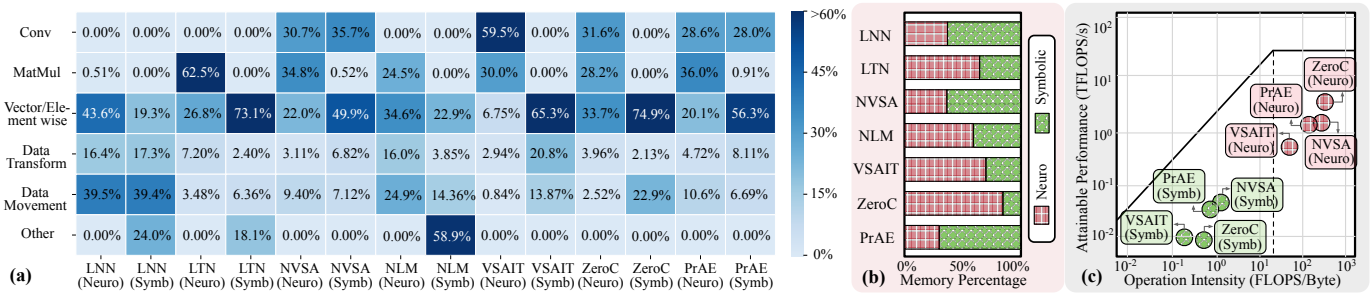


Fig. 3: Compute operators, memory and roofline characterization. **(a)** Compute operator runtime ratio of representative neuro-symbolic workloads, indicating neural operations mainly consisting of MatMul and Conv, while symbolic operations with vector/tensors. **(b)** Benchmark memory usage during computation and **(c)** roofline analysis on RTX 2080Ti GPU, showing typically neural operations are compute-bounded and symbolic operations are memory-bounded.

contrast, a large portion of LNN and NLM's (neuro) runtime is on vector and element-wise tensor operations due to the sparse syntax tree structure composed of proposition logic and the sequential logic deduction computations on multi-group architecture. Notably, data movement also takes up a significant amount of LNN (neuro) runtime because of its unique bidirectional dataflow during reasoning inference.

Symbolic Workload Analysis. The symbolic workload is dominated by vector and scalar operations that exhibit low operational intensities and complex control flows. Both LNN, LTN, and NLM's (symbolic) have a large number of logic operations, posing parallelism optimization opportunities in their database queries and arithmetic operations, especially for larger symbolic models. Meanwhile, LNN (symbolic) is severally data movement-bounded due to its sparse and irregular memory accesses and bidirectional inference, where model-aware dataflow architecture would likely be beneficial for alleviating this bottleneck. NVSA, VSAIT, and PrAE's (symbolic) are composed of vectors for vector-symbolic operations. Notably, these operations usually stem from high-dimensional distributed vector computations (e.g., binding, bundling) for symbolic representation, which are difficult to process efficiently on GPUs. Therefore, the challenges of accelerating these computations will become increasingly important as the task and feature complexities further grow. We leverage VSA kernels as a case study and present a cross-layer optimization solution in Sec. VI to improve system efficiency.

Takeaway 3: *The neural components mainly consist of MatMul and Convs, while the symbolic components are dominated by vector/element-wise tensor and logical operations. The data transfer overhead arising from the separate neural and symbolic execution on GPUs and CPUs poses efficient hardware design challenges.*

Recommendation 2: *From the architecture level, custom processing units can be built for efficient symbolic operations (e.g., high-dimensional distributed vectors, logical operation, graph, etc). For non-overlap neural and symbolic components, reconfigurable processing units supporting both neural and symbolic operations are recommended.*

C. Memory and System Analysis

Memory Usage Analysis. Fig. 3b characterizes the memory usage of the LNN, LTN, NVSA, NLM, VSAIT, ZeroC, and PrAE workloads during computation. We can observe that (1) PrAE (symbolic) consumes a high ratio of memory due to its large number of vector operations depending on intermediate results and exhaustive symbolic search. NVSA (symbolic) slightly alleviates the vector-symbolic operation memory by leveraging probabilistic abduction reasoning. ZeroC (neuro) contains energy-based models and process images in a large ensemble thus taking much memory. (2) In terms of storage footprint, neural weights and symbolic codebooks typically consume more storage. For example, neural network and holographic vector-inspired codebook account for >90% memory footprint in NVSA, because NVSA neural frontend enables the expression of more object combinations than vector space dimensions, requiring the codebook to be large enough to contain all object combinations and ensure quasi-orthogonality.

System Roofline Analysis. Fig. 3c employs the roofline model to quantify the memory boundedness of RTX 2080Ti

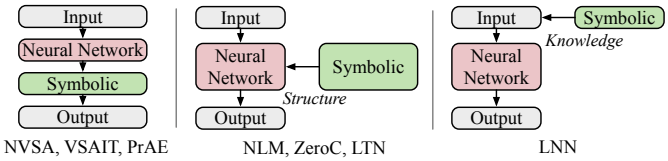


Fig. 4: Operator graph analysis. Symbolic operation depends on neural results or needs to compile in neural structure as the critical path. Complex control and symbolic-only phase operation result in inefficiency and low hardware resource utilization.

GPU versions of the selected workloads. We observe that the symbolic components are in the memory-bound area while neural components are in the compute-bound area. For example, NVSA and PrAE symbolic operations require streaming vector elements to circular convolution computing units, increasing the memory bandwidth pressure. Optimizing the compute dataflow and leveraging the scalable and reconfigurable processing element can help provide this bandwidth.

Takeaway 4: *Symbolic operations are memory-bounded due to large element streaming for vector-symbolic operations. Neural operations are compute-bounded due to computational-intensive MatMul/Convs. Neural weights and vector codebooks typically account for most storage while symbolic components require large intermediate caching during computation.*

Recommendation 3: *From the algorithm level, model compression (e.g., quantization and pruning) and efficient factorization of neural and symbolic components can be used to reduce memory and data movement overhead without sacrificing cognitive reasoning accuracy.*

Recommendation 4: *From the technology level, emerging memories and in/near-memory computing can alleviate the memory-bounded symbolic operations and improve scalability, performance, and efficiency of neuro-symbolic systems.*

D. Operation and Dataflow

Fig. 4 analyzes the operation dependency in representative neuro-symbolic workloads. We can observe that the reasoning computation of NVSA, VSAIT, and PrAE depends on the result of the frontend neural workload and thus lies on the critical path during inference. LNN, LTN, NLM, and ZeroC need to compile the symbolic knowledge in neural representation or input embeddings. The complex control results in inefficiency in CPU and GPU, and the vector-symbolic computation period results in low hardware utilization. There are opportunities for data pre-processing, parallel rule query, and heterogeneous and reconfigurable hardware design to reduce this bottleneck.

Takeaway 5: *The symbolic operations depend on the neural module results or need to compile into the neural structure, thus lying on the critical path of end-to-end neuro-symbolic systems. The vector-symbolic computation phase and complex control of neuro-symbolic components bring low hardware resource utilization and inefficiency in CPU/GPU.*

Recommendation 5: *From the system level, adaptive workload scheduling with parallelism processing of neural and symbolic components can be leveraged to alleviate resource underutilization and improve runtime efficiency.*

TABLE IV: Hardware inefficiency analysis. The compute, memory, and communication characteristics of representative neural and symbolic kernels in NVSA workload executed on CPU/GPU platform.

	Neural Kernel		Symbolic Kernel	
	sgemm_nn	relu_nn	vectorized_elem	elementwise
Compute Throughput (%)	95.1	92.9	3.0	2.3
ALU Utilization (%)	90.1	48.3	5.9	4.5
L1 Cache Throughput (%)	79.7	82.6	28.4	10.8
L2 Cache Throughput (%)	19.2	17.5	29.8	22.8
L1 Cache Hit Rate (%)	1.6	51.6	29.5	33.3
L2 Cache Hit Rate (%)	86.8	65.5	48.6	34.3
DRAM BW Utilization (%)	14.9	24.2	90.9	78.4

E. Hardware Inefficiency Analysis

The hardware inefficiencies of executing neuro-symbolic workloads mainly come from ALU underutilization, low cache hit rate, and massive data transfer. We leverage Nsight Systems/Compute tools to further characterize the GPU behavior of executing selected neuro-symbolic workloads. Tab. IV lists the compute, memory, and data movement characteristics of representative neural and symbolic kernels in NVSA as an example. We observe that typically in symbolic operations, the ALU utilization is $<$10%, the L1 cache hit rate is around 20%, the L2 cache hit rate is around 40%, and DRAM bandwidth utilization is around 90% with several memory-bounded. The data transfer memory operations account for around 50% of total latency, where $>$80% is from host CPU to GPU. Additionally, the synchronization overhead and waiting for GPU operations to complete results in CPU underutilization.

Takeaway 6: *While neural kernels exhibit high compute utilization and memory efficiency in GPUs, symbolic operations suffer from low ALU utilization, low L1 cache hit rates, and high memory transactions, resulting in low efficiency.*

Recommendation 6: *From the architecture level, heterogeneous or reconfigurable neural/symbolic architecture with efficient vector-symbolic units and high-bandwidth NoC can be optimized to improve ALU utilization and reduce data movement, thus improving system performance.*

F. Sparsity Analysis

Neuro-symbolic workloads also exhibit sparsity features. For example, Fig. 5 characterizes the sparsity of NVSA symbolic modules, including probabilistic mass function (PMF)-to-VSA transform, probability computation, and VSA-to-PMF transform, under different reasoning rule attributes. We can observe that NVSA has a high sparsity ratio ($>$95%) with variations for specific attributes and unstructured patterns. Similarly, ZeroC and LNN also demonstrate $>$90% sparsity ratio, while LTN features a dense computation pattern.

Takeaway 7: *Some neural and vector-symbolic components demonstrate a high level of unstructured sparsity with variations under different task scenarios and attributes.*

Recommendation 7: *From the algorithm and architecture level, sparsity-aware neural and symbolic algorithm and architecture design can benefit memory footprint, communication overhead, and computation FLOPS reduction.*

G. Uniqueness of Neuro-Symbolic vs. Neural Networks

To summarize, based on above analysis, neuro-symbolic AI workloads differ from neural networks mainly in three aspects:

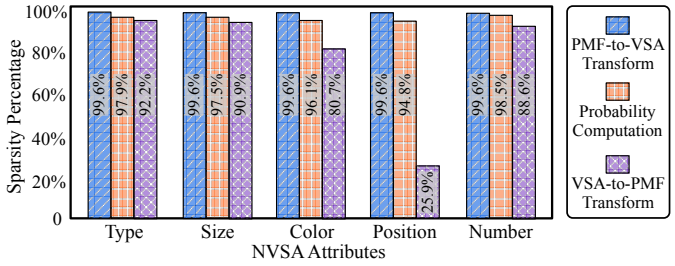


Fig. 5: Sparsity analysis. The Sparsity ratio of NVSA symbolic operations, shows a high degree of sparsity with variations in attributes.

Compute kernels. Neuro-symbolic workloads consist of heterogeneous neural and symbolic kernels. The symbolic operators (e.g., vector, graph, logic) are processed inefficiently on off-the-shelf CPUs/GPUs with low hardware utilization and cache hit and may result in runtime latency bottleneck.

Memory. Symbolic operations are memory-bounded due to large element streaming for vector-symbolic operations. Symbolic codebooks typically account for large memory footprints and require large intermediate caching during computation.

Dataflow and scalability. Neuro-symbolic workloads exhibit more complex control than NNs. Symbolic operations either critically depend on or compile in neural kernels. Their irregular dataflow, data dependency, and sequential processing bring low parallelism scalability and inefficiency in CPU/GPU.

VI. Case Study: Hardware Acceleration of Vector-Symbolic Architecture

This section presents a cross-layer acceleration case study for vector-symbolic architecture (VSA), which is a powerful model in many neuro-symbolic tasks [7], [21], [53], [54]. We develop a design method consisting of accelerated vector-symbolic kernel formulation (Sec. VI-A, VI-B), architecture and dataflow (Sec. VI-C), and programming method (Sec. VI-D), that overcomes computational inefficiencies from executing VSA components on CPUs and GPUs (Sec. VI-E).

Our proposed hardware design is inspired by neuro-symbolic workload insights from the characterization study in Sec. V. Specifically, as shown in Tab. V, it features (1) an energy-efficient dataflow with heterogeneous arithmetic units that can flexibly execute key vector-symbolic operations, (2) a distributed memory system employing near-memory computing to enhance scalability and memory performance, (3) compressed storage of symbolic operators to reduce the memory footprint of vector codebooks, and (4) a tiled design for vector-symbolic units to minimize data movement and optimize computational efficiency. These features collectively enable a highly efficient and scalable vector-symbolic hardware accelerator that significantly outperforms traditional platforms.

TABLE V: Design Features. Features of the proposed VSA processor and their association with design recommendations from Sec. V.

VSA Processor Feature	Fulfilled Recommendation
Compressed Storage of Symbols	Recommendation 3 (Sec. V-C)
Distributed Memory System	Recommendation 4 (Sec. V-C)
SIMD Multi-Tile Dataflow	Recommendation 5 (Sec. V-D)
Heterogeneous Arithmetic Processing	Recommendation 6 (Sec. V-E)

A. Vector-Symbolic Operations

In the vector-symbolic kernel, computational elements, such as scalars and objects, are represented with hypervectors which can be manipulated by a set of algebraic operations [15], [55], specifically, (1) binding, or element-wise multiplication, which creates a new hypervector that is quasi-orthogonal (dissimilar) to its constituents; (2) bundling, or element-wise addition, which combines hypervectors using element-wise majority count; (3) permutation, which rearranges the elements of a hypervector to preserve its order within a sequence; (4) scalar multiplication, which scales hypervector elements with a scalar weight The similarity between vectors is measured using a variety of distance metrics, such as the dot product, Hamming distance, L1, and L2 [56], [57]. These operations collectively form a mathematical framework for implementing various cognitive functions tailored for VSA operations [58].

B. Vector-Symbolic Kernel Formulation

We present a description of operations and programmability features of our proposed hardware accelerator using a formal representation, i.e., kernel function. We express this kernel function as $O := F(y, s)$, where $F(\cdot)$ integrates an array of kernel sub-functions f_i that together cover the whole domain of accelerator operations, and $y = \{y_1, y_2, \ldots\}$ represents an array combining all item and prototype vectors used in computation. The argument s is defined by a group of conditional variables $s = (s_1, s_2, \ldots)$, which together are used to draw the sub-domains associated with the sub-functions f_i.

The kernel functionality integrates computations for encoding and decoding, memory, and reasoning. Next, we formulate sub-functions f_i to describe these computations.

Encoding and Decoding Kernel. To facilitate the encoding and decoding, the kernel function needs to allow for flexible configuration of hypervector operations (binding, bundling, permutation). We take into account that binding can be distributed over bundling [59], and propose the kernel function:

$$a(y,(s_1,s_2)) := \begin{cases} b(y,(s_2)); & s_1 = 0 \\ \sum_i \left[b(y_i,(s_2))\right]; & s_1 = 1 \end{cases} \quad \forall\{i,j\} \subset \mathbb{N}$$

$$b(y,(s_2)) := \begin{cases} y; & s_2 = 0 \\ \otimes_j (y_j); & s_2 = 1 \\ \rho_j(y_j); & s_2 = 2 \\ \otimes_j \rho_{(j-1)}(y_j); & s_2 = 3 \end{cases} \quad \forall\{i,j\} \subset \mathbb{N}$$

where ρ_j means that the permutation operation (ρ) is repeated j times, i.e., $\rho_3(x) = \rho(\rho(\rho(x)))$. Likewise, when $j = 3$, the term $\otimes_j(x_j)$ becomes equivalent to $(x_1 \otimes x_2 \otimes x_3)$, and also $\otimes_j \rho_{(j-1)}(x_j)$ becomes equivalent to $(x_1 \otimes \rho(x_2) \otimes \rho(\rho(x_3)))$.

Resonator-Network Kernel. This is a template VSA kernel for reasoning functions. Specifically, it takes as input a composed vector (which may represent a visual scene involving multiple objects as in the RPM problem) and seeks to factorize the vector into its constituent factors. The operation of the resonator network involves iterative steps for similarity evaluation and projection [54]. The kernel function used for projection can be defined as follows: $c(y) := \sum_i [n_i \times y_i]; \forall i \in \mathbb{N};\ n_i \in \mathbb{Z}$. Here, $c(y)$ calculates a weighted sum of the vectors in y.

Task	Algorithm	Description	Kernel (s_1, s_2, s_3)
Reactive behavior learning and recall [53]	(1) $s_j \leftarrow \sum_i [obs_i]$	• Superpose state information	$(1,0,0)$
	(2) $m_j \leftarrow \sum_k [a_k \otimes v_k^l]$	• Binding motor value with ID (a_k)	$(1,1,0)$
	(3) $b_j \leftarrow \sum_l [lc_l]$	• Gather environment data (labels)	$(1,0,0)$
	(4) $x \leftarrow \sum_j (s_j \otimes m_j \otimes b_j)$	• Learning reactive behavior model	$(1,1,0)$
	(5) $\widehat{v_x} \leftarrow x \otimes (s_j \otimes b_j \otimes a_k)$	• Decoding of a motor value	$(0,1,0)$
	(6) $\text{argmax}_l\, d(v_k^l, \widehat{v_x})$	• Clean-up memory	$(-,-,2)$
Factoring – Single iteration [46]	(1) $x \leftarrow s \otimes (\hat{b} \otimes \hat{c} \otimes \hat{d})$	• Decoding step for factor a	$(0,1,0)$
	(2) $\hat{a} \leftarrow \sum_i [d(a_i, x) \times a_i]$	• Similarity against codebook of "a" and weighted bundling of vectors	$(1,0,1)$
	(3) $\text{argmax}_i\, d(a_i, \hat{a})$	• Finding the right item for factor "a" (following the last iteration)	$(-,-,2)$

Algorithm Legend: *Encoding*; *Decoding*; *Clean-up/Associative Memory*; *Resonator Network*

Fig. 6: Compact VSA Kernel Formulation. Illustration of how the VSA kernel is programmed to implement workloads.

The weight n_i is given by a function $d(y_i, \bar{y})$, which measures the similarity between items y_i and an estimate vector $\bar{y} \in y$.

Nearest-Neighbor Search Kernel. The similarity function $d(y_i, \bar{y})$ serves as the basis for identifying the closest vector to a query $\bar{y} \in y$ among an array of vectors $y = \{y_1, y_2, \cdots\}$. The array y represents item vectors when performing a clean-up memory search, and prototype vectors when performing an associative memory search. To describe this operation, we use a kernel function defined as follows: $e(y) := \text{argmax}_i\, d(y_i, \bar{y})$.

Kernel Support for Extended Vector Dimensions via Time-Multiplexing: A particular advantage of element-wise vector operations (binding, bundling, and permutation) is that they can process full-scale vectors and time-multiplexed folds similarly. In contrast, similarity operations in $d(y_i, \bar{y})$ require the time-multiplexed folds to be collapsed into a single vector representation. When a similarity quantity is computed using only a single fold, it represents a partial quantity. Therefore, to obtain the total similarity value, $d(y_i, \bar{y})$ needs to aggregate these partial quantities. We express this condition as follows: $d(y_i, \bar{y}) := \sum_k \left(y_{ik} \cdot \bar{y}_k\right)\ \forall\ i \in \mathbb{N};\ k \in \{1, 2, \ldots, L\}$. Here, L is the number of folds, and $\bar{y}_k$ and y_{ik} are the k-th folds of the vectors $\bar{y}$ and y_i, respectively. The dot product measures the similarity between these folds, and the sum over all k aggregates the similarities computed for all the folds.

Compact Kernel Formalism. Considering the information presented above, we present a compact and formal description of VSA hardware accelerator's kernel functionality as follows:

$$F\big(y, (s_1, s_2, s_3)\big) := \begin{cases} a(y, (s_1, s_2)); & s_3 = 0 \\ c(y); & s_3 = 1 \\ e(y); & s_3 = 2 \end{cases}$$

In this definition, the control variables (s_1, s_2, s_3) are used to dynamically adjust the behavior of the kernel during runtime. Fig. 6 demonstrates how the kernel is adjusted to execute VSA workloads. Performance results based on the mapping of these workloads and others are shown in Sec. VI-E.

C. Hardware Architecture and Dataflow

We present a method for constructing architecture dataflow informed by the derived VSA kernels. Fig. 7 shows the overall architecture, consisting of three subsystems: (1) memory and codebook-generation subsystem (MCG), (2) vector-symbolic operations subsystem (VOP), and (3) distance computation subsystem (DC). A control unit is used to decode instructions and determine control configurations. A description of these subsystems and their internal operations are presented below.

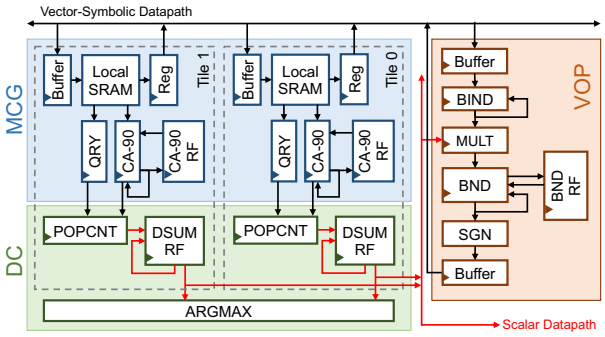


Fig. 7: Hardware Architecture and Dataflow. The proposed multi-tile architecture for VSA consists of MCG, DC, and VOP subsystems.

MCG Subsystem. This subsystem is distributed across multiple tiles, with each tile comprising four functional modules: a local memory (SRAM), a logic unit implementing cellular automata with rule 90 (CA-90) [60], a register file (CA-90 RF), and a query register (QRY). Vectors loaded from the local SRAM are processed exclusively within the tile's logic to leverage near-memory computing. The SRAMs are initialized with randomly generated atomic vectors (i.e., codebooks) used for symbolic encoding. The dimension of these vectors is constrained by the size of the physical datapth; therefore, we utilize a folding mechanism to support extended vector dimensions. CA-90 is integral to this mechanism, utilizing XOR and shift operations to generate new random vectors on-the-fly [60]. This design significantly reduces the memory footprint, as only seed folds need to be stored in the local SRAMs. CA-90 RF is a register file that temporarily stores newly generated folds to minimize redundant activations of CA-90. The QRY register holds query data required for similarity computation, an essential component of VSA.

VOP Subsystem. The VOP subsystem implements key VSA operations, used to construct distributed perceptual representations and perform symbolic reasoning computations. It consists of five logic units: a binding unit (BIND), a multiplying unit (MULT), a bundling unit (BND), a register file (BND RF), and a sign unit (SGN). BIND connects to a local buffer storing vectors and is to used to execute the binding operations over these vectors. The superposition of binded vectors is implemented in BND through element-wise addition (bundling). BIND and BND utilize different data representations, with BIND using binary and BND using integer formats. MULT manages the conversion from binary to integer formats and also performs element-wise scalar multiplication, an essential operation for neuro-symbolic encoding. Integer folds outputted from BND can be temporarily stored in BND RF for continuous superposition or converted to binary through SGN for transfer over the global vector-symbolic datapath.

DC Subsystem. This subsystem handles operations for distance computation and nearest neighbor search, and it comprises three critical logic units: POPCNT, DSUM RF, and ARGMAX. POPCNT evaluates the popcount of the difference between two vectors; it executes element-wise XOR operations followed by an addition operation to compute the difference between the number of 1's and the number of 0's in the difference vector. As POPCNT operates on partial vectors due

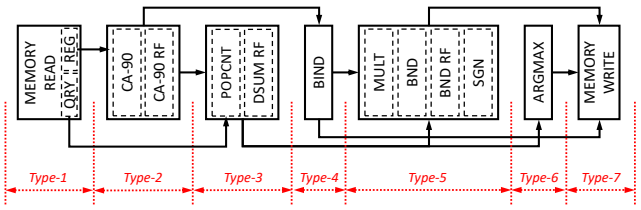


Fig. 8: VSA accelerator's pipeline stages and operation types.

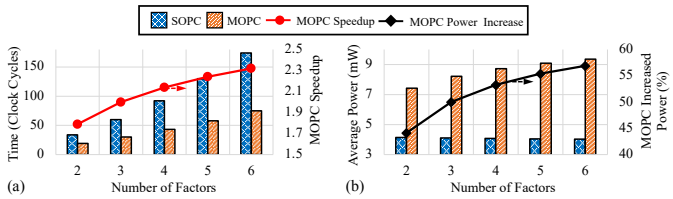


Fig. 9: Accelerator Control Methods Comparison. Runtime and power consumption results under two control methods (SOPC and MOPC) when executing the resonator network VSA algorithm.

OP_PARAM (57 bits)	Type_7 (3 bits)	Type_6 (3 bits)	Type_5 (3 bits)	Type_4 (2 bits)	Type_3 (3 bits)	Type_2 (3 bits)	Type_1 (2 bits)

Fig. 10: The *Instruction Word* format adopted in the proposed design.

to vector folding, its output also represents a partial distance quantity. Hence, DSUM RF facilitates distance accumulation over multiple partial vectors, distributing distance computations across multiple independently controllable registers. The resulting distance data is then communicated to ARGMAX, which manages the search for the nearest neighbor vector based on the transferred distance values.

Parameterized Multi-Tile Architecture. The integration of the above modules leads to a "single-tile" architecture, which includes a single instance of MCG and DC. We also propose a "multi-tile" architecture, which allows memory-bounded vector loading and similarity computations to be distributed across multiple tiles and exploits a SIMD implementation to speed up the execution. This approach, therefore, enables parallel, near-memory processing of symbolic computations, and hence improves the utilization of compute units. A multi-tile architecture also extends the storage capabilities, providing a means to accommodate larger models. Tiles are also equipped with configuration registers, which allow tiles to be selectively activated (or deactivated) before issuing instructions.

D. Accelerator Control Methods

The configuration of the different modules as described above exhibits a pipelined architecture that consists of seven pipeline stages, with each stage associated with a certain type of operation (Fig. 8). Such a pipelined configuration motivates a streamlined integration of dataflow and control-flow primitives, allowing different control methods to be applied without hazard. To perform this study, we particularly examine two control methods for this accelerator: *single-operation-per-cycle* (SOPC) and *multiple-operations-per-cycle* (MOPC).

SOPC and MOPC. SOPC simplifies programming and reduces power consumption since only one pipeline stage switches during each cycle. However, this approach increases runtime, making it unsuitable for high-throughput applications. Conversely, MOPC enables pipeline stages to perform operations simultaneously, thus increasing the number of operations per cycle. However, MOPC leads to increased power consumption and requires a complex mapping framework to analyze program dependencies and optimize control activities. MOPC is better suited for high-throughput applications that require a balance between runtime and power consumption.

Control Methods Comparison. We compare SOPC and MOPC by implementing factorization using the resonator network kernel. Fig. 9 compares the runtime and power consumption of SOPC and MOPC when executing at various complexity levels (number of factors). We observe that MOPC achieves lower runtime in comparison with SOPC, and that the speed-up gained by using MOPC increases from 1.8 to 2.3. However, using MOPC also increases power consumption by 44% to 57% as the complexity increases. We adopt MOPC in our design because its better speedup capability is especially important when multiple heterogeneous tasks need to be executed simultaneously. Moreover, the speed-up gain of MOPC can be flexibly configured based on power-consumption constraints for low-power purposes.

Accelerator Instruction Format. To realize the MOPC control method, we design an instruction-set architecture that employs a wide-word macro format, referred to as *Instruction Word*. Similar to a Very-Large Instruction Word (VLIW), a single *Word* consists of multiple operations, except that these operations are sequential in the pipelined dataflow and not parallel like VLIW architectures. As shown in Fig. 10, the *Word* format consists of seven `Type` fields, used to specify the operations to be executed in seven pipelined stages, and an `OP_PARAM` field, used to configure `Type` operations. This approach offers a high degree of flexibility and is commonly used with domain-specific processors. Details on the instruction fields and compiler optimization are omitted due to space.

E. Evaluation Results

Experimental Setup. The design was implemented in SystemVerilog and synthesized with Synopsys Design Compiler using foundry 28nm library. Tab. VI lists the architectural parameters. The energy is measured using Synopsys PrimeTime PX. VSA workloads are also simulated on NVIDIA V100 GPU as the baseline, and GPU power was measured using the `nvidia-smi` utility. The algorithms listed in Tab. VII are used for evaluation, facilitating a comprehensive assessment of multi-layer cognition systems.

Latency. We first evaluate the impact of varying VSA accelerator (Acc) size on latency. Fig. 11a shows that Acc_4 provides speed-up of 1.3-1.8× compared to Acc_2, highlighting resource underprovisioning in Acc_2. However, we observe that the benefits of scaling up the design from Acc_4 to Acc_8 are not equally realized by all algorithms. Specifically, only 1.16× speed-up is achieved by MULT. This is because MULT typically performs VOP-intensive computations for sequence encoding and thus its response to further increase in design size is minimal. On the other hand, REACT achieves 1.69× speed-up when Acc_8 is used. This result is attributed to the fact that REACT performs extensive clean-up memory operations, which can be efficiently distributed across all tiles.

TABLE VI: Hardware Setup. VSA accelerator (Acc) configurations.

Instance Name	Bus Width (*W*)	#Tiles (*K*)	#CA-90-RF register (*R*)	#BND-RF register (*B*)	#DSUM register (*D*)	Distance bit -width (*C*)	BND bit-width (*H*)	Memory Capacity
Acc_2	512	2	2	2	2	12	8	128 KB
Acc_4	512	4	4	4	4	12	8	256 KB
Acc_8	512	8	8	8	8	12	8	512 KB

TABLE VII: Algorithm Setup. VSA workloads used in evaluation.

Workload	Layer	Application	Problem Size (Complexity)
MULT	Perception	Multi-modal learning and Inference [61]	300 samples, 120 item vectors, 16 prototype vectors (classes), 100 queries
TREE	Reasoning	Tree encoding and search [53]	70 tree structures, 9 items, 400 queries
FACT	Reasoning	Factorization of data sets [54]	60 iterations, 120 item vectors, 13 prototype vectors
REACT	Control	Motor learning and recall [62]	500 samples, 55 item vectors, 160 recalls

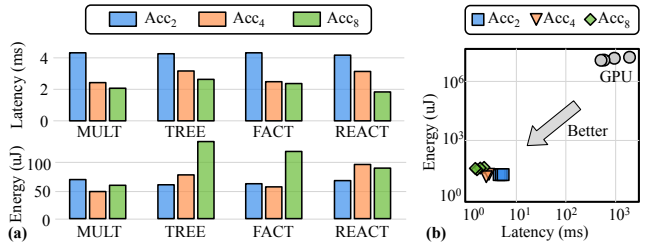


Fig. 11: VSA Accelerator (Acc) Efficiency. (a) Comparison between Acc_2, Acc_4, and Acc_8 in terms of latency and energy consumption across workloads. **(b)** Comparison between Acc and GPU (baseline).

Energy Consumption. Fig. 11a shows that the energy efficiency does not exhibit systematic behavior as the accelerator size varies. The reasons are twofold: (1) The leakage power becomes increasingly significant when Acc size is increased. Our analysis shows that the leakage power increases from 1.7 mW to 5.2 mW (i.e., 3× increase) when the design is scaled up from Acc_2 to Acc_8. (2) Each of the instructions has a unique effect on energy consumption, especially because instructions trigger circuit activity at different hardware modules.

Comparison with GPU. We also compare all VSA accelerator instances with GPU in terms of latency and energy consumption. Fig. 11b shows that Acc is up to three orders of magnitude faster in executing VSA workloads than GPU, despite using batch processing in our GPU implementation. This result consolidates suggests the GPU-memory interface is not optimized for VSA data transfer. In addition, Acc operation is up to six orders of magnitude more energy efficient than GPU processing. This performance gap is attributed to GPU's scalar architecture, which relies on complex SIMD arithmetic units to perform simple vector operations.

VII. Outlook and Research Opportunities

In this section, we discuss the challenges and opportunities for neuro-symbolic systems, and outline our vision for the future, focusing on the system and architecture perspectives.

Building ImageNet-like neuro-symbolic datasets. Neuro-symbolic systems hold great potential in achieving human-like performance [63]. However, their current applications are still limited to basic decision-making and reasoning problems, falling short of the broader vision of human cognitive abilities, such as deductive reasoning, compositionality, and counterfactual thinking. It is still an open question of how perception learned from other domains can be transferred to abstract reasoning tasks. To significantly advance the metacognitive capabilities of neuro-symbolic systems, more challenging and suitable datasets are highly desirable to unleash its potential.

Unifying neuro-symbolic models. Integrating neural, symbolic, and probabilistic approaches offers promise to improve AI models' explainability and robustness. However, the current attempts to combine these complementary approaches are still in a nascent manner - how to integrate them in a principled manner remains an open challenge. Particularly, symbolic components can be combined with Large Language Models (LLMs) to improve their planning and reasoning capabilities [64]. We envision a unified framework to design algorithms that opportunistically combine neural and symbolic with probabilistic representations, and for quantifying scaling laws for neuro-symbolic inference versus large neural models.

Developing efficient software frameworks. Neuro-symbolic AI systems typically utilize underlying logic, such as fuzzy logic, parameterization, and differentiable structures, to support learning and reasoning capabilities. However, most system implementations create custom software for deduction for the particular logic, which limits modularity and extensibility. Thus, new software frameworks are needed that can encompass a broad set of reasoning logical capabilities and provide practical syntactic and semantic extensions while being fast and memory-efficient. Moreover, new programming models and compilers that can facilitate the ease and efficient realization of the neuro-symbolic models are of significance to realize the full promise of neuro-symbolic AI paradigms.

Benchmarking diverse neuro-symbolic workloads. Given the proliferation of neuro-symbolic algorithms and the rapid hardware advancements, it is crucial to benchmark neuro-symbolic AI systems in a comparable and validated manner. To achieve this, from the system aspect, we need representative benchmarks that capture the essential workload characteristics (e.g., compute kernels, access patterns, and sparsity) of neural and symbolic models, and that can be quantitatively tested in human-AI applications. From an architectural and hardware perspective, we need modeling-simulation frameworks to enable the development of novel architectures for these workloads and build optimized modular blocks as libraries by leveraging workload characteristics. Benchmarking neuro-symbolic computing will guide ML researchers and system architects in investigating the trade-offs in accuracy, performance, and efficiency of various neuro-symbolic algorithms, and in implementing systems in a performance-portable way.

Designing cognitive hardware architectures. Neuro-symbolic workloads that combine neural, symbolic, and probabilistic methods feature much greater heterogeneity in compute kernels, sparsity, irregularity in access patterns, and higher memory intensity than DNNs. This leads to an increasing divergence with the current hardware roadmap that largely focuses on matrix multiplication and regular dataflow. Therefore, we need novel architectures with dedicated processing units, memory hierarchies, and NoCs that can handle the additional complexities in computations and communications. Additionally, the architecture needs to provide flexibility with both configurable interconnects and full addressable memories to keep pace with neuro-symbolic AI algorithmic innovations.

VIII. Conclusion

Neuro-symbolic AI is an emerging paradigm for developing efficient, robust, explainable, and cognitively advanced AI systems. This paper provides a systematic characterization of neuro-symbolic system performance and analyzes their operational components. Leveraging insights from profiling, we propose cross-layer optimization techniques and present a case study of a hardware architecture designed to enhance their performance and efficiency. We believe this research will address key challenges and highlight opportunities essential for advancing next-generation neuro-symbolic AI systems.

Acknowledgements

This work was supported in part by CoCoSys, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai, *et al.*, "Sustainable ai: Environmental implications, challenges and opportunities," *Proceedings of Machine Learning and Systems (MLSys)*, vol. 4, pp. 795–813, 2022.

[2] Z. Wan, A. Anwar, Y.-S. Hsiao, T. Jia, V. J. Reddi, and A. Raychowdhury, "Analyzing and improving fault tolerance of learning-based navigation systems," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 841–846, IEEE, 2021.

[3] E. Debenedetti, Z. Wan, M. Andriushchenko, V. Sehwag, K. Bhardwaj, and B. Kailkhura, "Scaling compute is not all you need for adversarial robustness," *arXiv preprint arXiv:2312.13131*, 2023.

[4] Z. Wan, C.-K. Liu, R. Raj, C. Li, H. You, Y. Fu, C. Wan, A. Samajdar, C. Lin, T. Krishna, and A. Raychowdhury, "Towards cognitive ai systems: Workload and characterization of neuro-symbolic ai," in *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2024.

[5] A. d. Garcez and L. C. Lamb, "Neurosymbolic ai: The 3 rd wave," *Artificial Intelligence Review*, pp. 1–20, 2023.

[6] Z. Wan, C.-K. Liu, H. Yang, C. Li, H. You, Y. Fu, C. Wan, T. Krishna, Y. Lin, and A. Raychowdhury, "Towards cognitive ai systems: a survey and prospective on neuro-symbolic ai," *arXiv preprint arXiv:2401.01040*, 2024.

[7] M. Hersche, M. Zeqiri, L. Benini, A. Sebastian, and A. Rahimi *Nature Machine Intelligence*, pp. 1–13, 2023.

[8] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," in *International Conference on Learning Representations (ICLR)*, 2019.

[9] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, *et al.*, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," *arXiv preprint arXiv:2309.16650*, 2023.

[10] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of dnn accelerators using scale-sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 58–68, IEEE, 2020.

[11] H. Kwon, L. Lai, M. Pellauer, T. Krishna, Y.-H. Chen, and V. Chandra, "Heterogeneous dataflow accelerators for multi-dnn workloads," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 71–83, IEEE, 2021.

[12] Y. N. Wu, P.-A. Tsai, S. Muralidharan, A. Parashar, V. Sze, and J. Emer, "Highlight: Efficient and flexible dnn acceleration with hierarchical structured sparsity," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1106–1120, 2023.

[13] A. Ramachandran, Z. Wan, G. Jeong, J. Gustafson, and T. Krishna, "Algorithm-hardware co-design of distribution-aware logarithmic-posit encodings for efficient dnn inference," *arXiv preprint arXiv:2403.05465*, 2024.

[14] Z. Fan, Z. Wan, C.-K. Liu, A. Lu, K. Bhardwaj, and A. Raychowdhury, "Benchmarking test-time dnn adaptation at edge with compute-in-memory," *Journal on Autonomous Transportation Systems*, 2024.

[15] M. Ibrahim, Y. Kim, and J. M. Rabaey, "Efficient design of a hyperdimensional processing unit for multi-layer cognition," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2024.

[16] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[17] C. Pryor, C. Dickens, E. Augustine, A. Albalak, W. Wang, and L. Getoor, "Neupsl: Neural probabilistic soft logic," *arXiv preprint arXiv:2205.14268*, 2022.

[18] Z. Yang, A. Ishay, and J. Lee, "Neurasp: Embracing neural networks into answer set programming," in *29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[19] W.-Z. Dai, Q. Xu, Y. Yu, and Z.-H. Zhou, "Bridging machine learning and logical reasoning by abductive learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[20] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[21] J. Theiss, J. Leverett, D. Kim, and A. Prakash, "Unpaired image translation via vector symbolic architectures," in *European Conference on Computer Vision (ECCV)*, pp. 17–32, Springer, 2022.

[22] C. Zhang, B. Jia, S.-C. Zhu, and Y. Zhu, "Abstract spatial-temporal reasoning via probabilistic abduction and execution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9736–9746, 2021.

[23] R. Riegel, A. Gray, F. Luus, N. Khan, N. Makondo, I. Y. Akhalwaya, H. Qian, R. Fagin, F. Barahona, U. Sharma, *et al.*, "Logical neural networks," *arXiv preprint arXiv:2006.13155*, 2020.

[24] G. Lample and F. Charton, "Deep learning for symbolic mathematics," in *International Conference on Learning Representations (ICLR)*, 2019.

[25] R. Evans and E. Grefenstette, "Learning explanatory rules from noisy data," *Journal of Artificial Intelligence Research*, vol. 61, pp. 1–64, 2018.

[26] S. Badreddine, A. d. Garcez, L. Serafini, and M. Spranger, "Logic tensor networks," *Artificial Intelligence*, vol. 303, p. 103649, 2022.

[27] P. Hohenecker and T. Lukas, "Ontology reasoning with deep neural networks," *Journal of Artificial Intelligence Research*, vol. 68, pp. 503–540, 2020.

[28] L. C. Lamb, A. Garcez, M. Gori, M. Prates, P. Avelar, and M. Vardi, "Graph neural networks meet neural-symbolic computing: A survey and perspective," in *IJCAI 2020-29th International Joint Conference on Artificial Intelligence*, 2020.

[29] T. Wu, M. Tjandrasuwita, Z. Wu, X. Yang, K. Liu, R. Sosic, and J. Leskovec, "Zeroc: A neuro-symbolic model for zero-shot concept recognition and acquisition at inference time," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022.

[30] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou, "Neural logic machines," in *International Conference on Learning Representations (ICLR)*, 2019.

[31] H. Kaut, "Robert s. engelmore memorial lecture at aaai 2020," *https://roc-hci.com/announcements/the-third-ai-summer/*, 2020.

[32] H. Zhang and T. Yu, "Alphazero," *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 391–415, 2020.

[33] C. Zhang, F. Gao, B. Jia, Y. Zhu, and S.-C. Zhu, "Raven: A dataset for relational and analogical visual reasoning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5317–5327, 2019.

[34] S. Hu, Y. Ma, X. Liu, Y. Wei, and S. Bai, "Stratified rule-aware network for abstract visual reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, pp. 1567–1574, 2021.

[35] R. Manhaeve, S. Dumančić, A. Kimmig, T. Demeester, and L. De Raedt, "Neural probabilistic logic programming in deepproblog," *Artificial Intelligence*, vol. 298, p. 103504, 2021.

[36] K. Yi, C. Gan, Y. Li, P. Kohli, J. Wu, A. Torralba, and J. B. Tenenbaum, "Clevrer: Collision events for video representation and reasoning," in *International Conference on Learning Representations (ICLR)*, 2020.

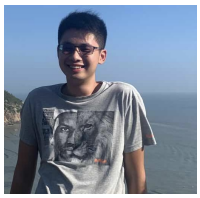
[37] A. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran, "Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning," *arXiv preprint arXiv:1905.06088*, 2019.

[38] Y. Guo, Z. Pan, and J. Heflin, "Lubm: A benchmark for owl knowledge base systems," *Journal of Web Semantics*, vol. 3, no. 2-3, 2005.

[39] G. Sutcliffe, "The tptp problem library and associated infrastructure," *Journal of Automated Reasoning*, vol. 59, no. 4, pp. 483–502, 2017.
[40] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.
[41] Ö. GENCER, "The research about morphometric characteristics on leptograpsus crabs," 2023.
[42] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt, "Deepproblog: Neural probabilistic logic programming," *Advances in neural information processing systems (NeurIPS)*, vol. 31, 2018.
[43] D. Barrett, F. Hill, A. Santoro, A. Morcos, and T. Lillicrap, "Measuring abstract reasoning in neural networks," in *International conference on machine learning (ICML)*, pp. 511–520, PMLR, 2018.
[44] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.
[45] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, pp. 102–118, Springer, 2016.
[46] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.
[47] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.
[48] F. Chollet, "On the measure of intelligence," *arXiv preprint arXiv:1911.01547*, 2019.
[49] M. Shanahan, K. Nikiforou, A. Creswell, C. Kaplanis, D. Barrett, and M. Garnelo, "An explicitly relational neural network architecture," in *International Conference on Machine Learning (ICML)*, PMLR, 2020.
[50] C. Zhang, F. Gao, B. Jia, Y. Zhu, and S.-C. Zhu, "Raven: A dataset for relational and analogical visual reasoning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 5317–5327, 2019.
[51] S. Hu, Y. Ma, X. Liu, Y. Wei, and S. Bai, "Stratified rule-aware network for abstract visual reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, pp. 1567–1574, 2021.
[52] Z. Susskind, B. Arden, L. K. John, P. Stockton, and E. B. John, "Neuro-symbolic ai: An emerging class of ai workloads and their characterization," *arXiv preprint arXiv:2109.06133*, 2021.
[53] D. Kleyko *et al.*, "Vector Symbolic Architectures as a Computing Framework for Emerging Hardware," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1538–1571, 2022.
[54] E. P. Frady *et al.*, "Resonator Networks, 1: An Efficient Solution for Factoring High-Dimensional, Distributed Representations of Data Structures," *Neural Computation*, vol. 32, no. 12, pp. 2311–2331, 2020.
[55] Z. Wan, C.-K. Liu, M. Ibrahim, H. Yang, S. Spetalnick, T. Krishna, and A. Raychowdhury, "H3dfact: Heterogeneous 3d integrated cim for factorization with holographic perceptual representations," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2024.
[56] Z. Xu, C.-K. Liu, C. Li, R. Mao, J. Yang, T. Kämpfe, M. Imani, C. Li, C. Zhuo, and X. Yin, "Ferex: A reconfigurable design of multi-bit ferroelectric compute-in-memory for nearest neighbor search," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2024*, IEEE, 2024.
[57] S. Shou, C.-K. Liu, S. Yun, Z. Wan, K. Ni, M. Imani, X. S. Hu, J. Yang, C. Zhuo, and X. Yin, "See-mcam: Scalable multi-bit fefet content addressable memories for energy efficient associative search," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9, IEEE, 2023.
[58] D. Kleyko, M. Davies, E. P. Frady, P. Kanerva, S. J. Kent, B. A. Olshausen, E. Osipov, J. M. Rabaey, D. A. Rachkovskij, A. Rahimi, *et al.*, "Vector symbolic architectures as a computing framework for emerging hardware," *Proceedings of the IEEE*, vol. 110, no. 10, 2022.
[59] P. Kanerva, "Prototypes and Mapping in Concept Space," in *Proceedings of AAAI Fall Symposium: Quantum Informatics for Cognitive Social and Semantic Processes*, pp. 2–6, 2010.
[60] D. Kleyko, E. P. Frady, and F. T. Sommer, "Cellular automata can reduce memory requirements of collective-state computing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, 2021.
[61] S. Datta *et al.*, "A Programmable Hyper-Dimensional Processor Architecture for Human-Centric IoT," *IEEE JETCAS*, vol. 9, no. 3, 2019.
[62] A. Menon *et al.*, "Shared Control of Assistive Robots through User-intent Prediction and Hyperdimensional Recall of Reactive Behavior," in *Proc. IEEE ICRA*, pp. 12638–12644, 2023.
[63] G. Booch, F. Fabiano, L. Horesh, K. Kate, J. Lenchner, N. Linck, A. Loreggia, K. Murgesan, N. Mattei, F. Rossi, *et al.*, "Thinking fast and slow in ai," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, pp. 15042–15046, 2021.
[64] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy, "Llms can't plan, but can help planning in llm-modulo frameworks," *arXiv preprint arXiv:2402.01817*, 2024.

Zishen Wan (Student Member, IEEE) received the B.E. degree in electrical engineering and automation from Harbin Institute of Technology, Harbin, China, in 2018, and the M.S. degree in electrical engineering from Harvard University, Cambridge, MA, USA, in 2020. Currently, he is pursuing Ph.D. degree with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include computer architecture, VLSI, and embedded systems, with a focus on designing efficient and reliable hardware and systems for autonomous machines and cognitive intelligence.

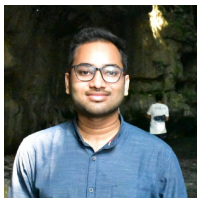


Che-Kai Liu (Student Member, IEEE) received the B.Eng. degree in Electronic Engineering from Zhejiang University, Hangzhou, China, in 2023. He is currently pursuing Ph.D. degree at the Integrated Circuits and Systems Research Lab (ICSRL), Georgia Institute of Technology, USA, under the supervision of Dr. Arijit Raychowdhury. He currently holds a research intern position at Corporate Research, TSMC, San Jose, USA.

His research interest includes SoC prototype for next-generation AI applications.

Hanchen Yang (Student Member, IEEE) received the B.S. degree in applied physics from Beijing University of Posts and Telecommunications, Beijing, China, in 2018, and the M.S. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2020. Currently, he is pursuing Ph.D. degree with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include computer architecture, neural-symbolic AI, and hardware-software co-design for novel ML applications.

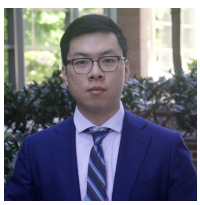


Ritik Raj (Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering from Indian Institute of Technology, Roorkee, India, in 2023. Currently, he is pursuing Ph.D. degree with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include computer architecture, specifically AI and domain-specific accelerators, FPGA, and architecture simulator design.

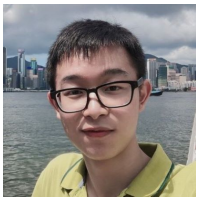
Chaojian Li (Student Member, IEEE) received the B.E. degree in precision instrument from Tsinghua University, Beijing, China, in 2019. Currently, he is pursuing Ph.D. degree with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include deep learning and computer architecture, with a focus on 3D reconstruction and rendering in an algorithm-hardware co-design approach and deep learning on edge devices.



Haoran You (Student Member, IEEE) received the B.E. degree in electronic information and communication from Huazhong University of Science and Technology, Wuhan, China, in 2019. Currently, he is pursuing Ph.D. degree with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests are efficient and automated ML/AI systems through algorithm-hardware co-design.



Yonggan Fu (Student Member, IEEE) received the B.E. degree in applied physics and computer science from University of Science and Technology of China, Hefei, China, in 2019. Currently, he is pursuing Ph.D. degree with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests are developing efficient and robust AI algorithms and co-designing the corresponding hardware accelerators towards a triple-win in accuracy, efficiency, and robustness.

Cheng Wan (Student Member, IEEE) received the B.E. degree in computer science at Shanghai Jiao Tong University, Shanghai, China, in 2018. Currently, he is pursuing Ph.D. degree with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include algorithm-system co-design for machine learning systems, with a special focus on distributed training.

Sixu Li (Student Member, IEEE) received the B.E. degree in communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2021. Currently, he is pursuing Ph.D. degree with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include computer architecture, digital circuits and systems, with a focus on accelerator design for neural rendering.

Youbin Kim (Student Member, IEEE) received the B.A. degree in physics from Harvard University, Cambridge, MA, USA, in 2018. Currently, he is pursuing Ph.D. degree with the School of Electrical Engineering, University of California, Berkeley, Berkeley, CA, USA.

His research interests include digital integrated circuits, computer architecture, and the design of hardware accelerators for high-dimensional computing and machine learning algorithms.

Ananda Samajdar (Member, IEEE) received the Ph.D. degree in electrical and computer engineering, Georgia Institute of Technology, Atlanta, GA, USA, in 2022. Currently, he is the research staff member at IBM T.J. Watson Research Center, Yorktown Heights, NY, USA.

His research interests include computer archtitecture, VLSI design, computer systems design, machine learning algorithm development.

Yingyan (Celine) Lin (Member, IEEE) received her Ph.D. degree in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, in 2017. Currently, she is an Associate Professor at the School of Computer Science and also serves as the Co-Director of the Center for Advancing Responsible Computing (CARE) at the Georgia Institute of Technology, Atlanta, GA, USA.

Dr. Lin's research focuses on developing efficient machine learning solutions through cross-layer innovations from efficient AI algorithms to AI accelerator and chip design in order to enable ubiquitous on-device intelligence and promote green AI. Her work has won first place in both the University Demonstration at DAC 2022 and the ACM/IEEE TinyML Design Contest at ICCAD 2022, and was selected as an IEEE Micro Top Pick of 2023. She has received the NSF CAREER Award, the IBM Faculty Award, the Meta Faculty Research Award, the ACM SIGDA Outstanding Young Faculty Award, and recently the SRC Young Faculty Award. She is currently serving as the Program Co-Chair for the eighth annual Machine Learning and Systems conference (MLSys 2025) and on the Technical Program Committees of several first-tier conferences, such as DAC and MICRO.

Mohamed Ibrahim (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Duke University, Durham, NC, in 2018. He was a postdoctoral researcher at the University of California at Berkeley, Berkeley, CA, USA. Currently, he is a senior research engineer and research faculty member with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

His research is broadly at the intersection of energy-efficient AI, brain-inspired computing, and human-centered interactive systems.

Jan M. Rabaey (Life Fellow, IEEE) is a professor with Graduate School, Electrical Engineering and Computer Science Department, the University of California at Berkeley, Berkeley, CA, USA, after being the holder of the Donald O. Pederson Distinguished Professorship at the same institute for over 30 years. He is a founding director of the Berkeley Wireless Research Center (BWRC) and the Berkeley Ubiquitous SwarmLab, and has served as the Electrical Engineering Division Chair at Berkeley twice. In 2019, he also became the CTO of the System-Technology Co-Optimization (STCO) Division of IMEC, Belgium.

Dr. Rabaey has made high-impact contributions to a number of fields, including low power integrated circuits, advanced wireless systems, mobile devices, sensor networks, and ubiquitous computing. Some of the systems he helped envision include the infoPad (a forerunner of the iPad), PicoNets and PicoRadios (IoT avant-la-lettre), the Swarm (IoT on steroids), Brain-Machine interfaces and the Human Intranet. His current interests include the conception of the next-generation distributed systems, as well as the exploration of the interaction between the cyber and the biological worlds. He is the primary author of the influential *Digital Integrated Circuits: A Design Perspective* textbook that has served to educate hundreds of thousands of students all over the world. He is the recipient of numerous awards, is a Life Fellow of the IEEE, and has been involved in a broad variety of start-up ventures.

Tushar Krishna (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2014. Currently, he is an Associate Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

Dr. Krishna's research spans computer architecture, interconnection networks, networks-on-chip, and AI/ML accelerator systems – with a focus on optimizing data movement in modern computing platforms. His research is funded via multiple awards from NSF, DARPA, IARPA, SRC (including JUMP2.0), Department of Energy, Intel, Google, Meta/Facebook, Qualcomm and TSMC. Three of his papers have been selected for IEEE Micro's Top Picks from Computer Architecture, one more received an honorable mention, and four have won best paper awards. Dr. Krishna was inducted into the HPCA Hall of Fame in 2022. He has been honored by the Roger P. Webb Outstanding Junior Faculty Award in 2021, the Richard M. Bass/Eta Kappa Nu Outstanding Junior Teacher Award in 2023, and the Roger P. Webb Outstanding Mid-career Faculty Award in 2024.

Arijit Raychowdhury (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2007. He is currently the Steve W Chaddick Chair and a Professor with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. He is also the Director of the Center for the Co-Design of Cognitive Systems (CoCoSys), a Joint University Microelectronics Program 2.0. His research interests include low-power digital and mixed-signal circuit design, signal processors, and exploring interactions of circuits with device technologies.

Dr. Raychowdhury is currently a Distinguished Lecturer of the IEEE Solid State Circuits Society (SSCS). He serves on the Technical Program Committee of key circuits and design conferences, including ISSCC, VLSI Symposium, DAC, and CICC. He is the winner of several awards, including the SRC Technical Excellence Award in 2021, the Qualcomm Faculty Award in 2021 and 2020, the IEEE/ACM Innovator under 40 Award, and the NSF CISE Research Initiation Initiative Award (CRII) in 2015.