Neuro-Symbolic Architecture Meets Large Language Models: A Memory-Centric Perspective

Mohamed Ibrahim[1], Zishen Wan[1], Haitong Li[2], Priyadarshini Panda[3], Tushar Krishna[1], Pentti Kanerva[4], Yiran Chen[5], and Arijit Raychowdhury[1]

[1]*Georgia Institute of Technology* [2]*Purdue University* [3]*Yale University*
[4]*University of California, Berkeley* [5]*Duke University*

***Abstract*—Large language models (LLMs) have significantly transformed the landscape of artificial intelligence, demonstrating exceptional capabilities in natural language understanding and generation. Recently, the integration of LLMs with neuro-symbolic architectures has gained traction to enhance contextual awareness and planning capabilities. However, this integration faces computational challenges that hinder scalability and efficiency, especially in edge computing environments. This paper provides an in-depth analysis of these challenges and explores state-of-the-art solutions, focusing on memory-centric computing principles at both algorithmic and hardware levels. Our exploration is centered around the key computational elements of the Transformer, the foundation of all LLMs, and vector-symbolic architecture, the leading neuro-symbolic model for edge applications. Additionally, we propose potential research directions for further investigation. By examining these aspects, this paper aims to bridge critical gaps in the path toward effective artificial general intelligence at the edge.**

I. Introduction

Advances in large language models (LLMs) have revolutionized the field of artificial intelligence (AI) with their powerful capabilities for understanding and generating human language. LLMs, such as OpenAI's GPT series [1], [2] and Google's BERT [3], leverage vast amounts of data and deep Transformer architectures to achieve remarkable performance in a wide range of tasks, from machine translation and sentiment analysis to content generation and question answering [4], [5]. By capturing intricate patterns in text, LLMs can emulate cognitive processes, providing responses that exhibit a high degree of fluency and coherence. Their ability to process and generate human-like text has opened new avenues for AI-driven innovation in various domains, including robotics, healthcare, and many conversational services [6]–[8].

Research into the integration of LLMs with neuro-symbolic (NeSy) systems has gained considerable momentum in recent years. NeSy models merge the adaptive learning capabilities of neural networks with the structured, rule-based reasoning of symbolic systems [9]. A prominent example of such a model is the vector-symbolic architecture (VSA), also known as hyperdimensional computing. This brain-inspired paradigm encodes and processes neuro-symbolic information using high-dimensional vectors, enabling efficient representation and processing of complex data structures for learning and reasoning [10], [11]. By integrating LLMs with NeSy models like VSA, *researchers aim to enhance the contextual understanding and reasoning capabilities of LLM-powered systems*. Currently, these systems often struggle to maintain context and exhibit depth in long interactions, leading to responses that may be relevant yet lack strategic insight [12]–[14]. Additionally, training an LLM like GPT-3 can consume substantial amounts of energy and produce significant carbon emissions, with estimates showing 1,287 MWh of electricity consumption and approximately 552 metric tons of CO_2 emissions [15]. Integrating NeSy models, which utilize sparse activation and improved contextual understanding, can potentially reduce these costs by up to 10 times, *significantly lowering both the energy consumption and the carbon footprint without sacrificing accuracy*. Hence, addressing these limitations through the integration of LLMs and NeSy models holds the promise of advancing the development of effective and energy-efficient AI systems, particularly in the realm of embodied cognition [16].

Despite the impressive advancements of this hybrid approach, current developments still face critical challenges that hinder their broader adoption and effectiveness. One major challenge is the "memory-wall" problem, which arises from the growing disparity between processor speeds and memory-access speeds. As processors become faster, memory access has not kept pace, creating bottlenecks that limit overall system performance [17]. This issue is especially pronounced in models like LLMs and VSA, where efficient data flow is crucial [18]. Another challenge is the diverse computational characteristics within these models. Components within LLMs and VSA may exhibit varying levels of some important attributes, such as noise tolerance, intermediate caching needs, and data sparsity [19]. Addressing these varying demands necessitates advanced memory systems capable of optimizing performance across various computational needs.

This paper provides an in-depth analysis of the aforementioned challenges and investigates memory-centric design and optimization strategies at both algorithmic and hardware levels (see Fig. 1). On the algorithmic front, we explore techniques such as model compression, mixture of experts, federated learning, and computation-in-superposition, evaluating their potential to enhance computational efficiency and model performance in both LLMs and NeSy systems. At the hardware level, we address the transformative shift towards memory-centric computing (MCC), focusing on compute-in-memory

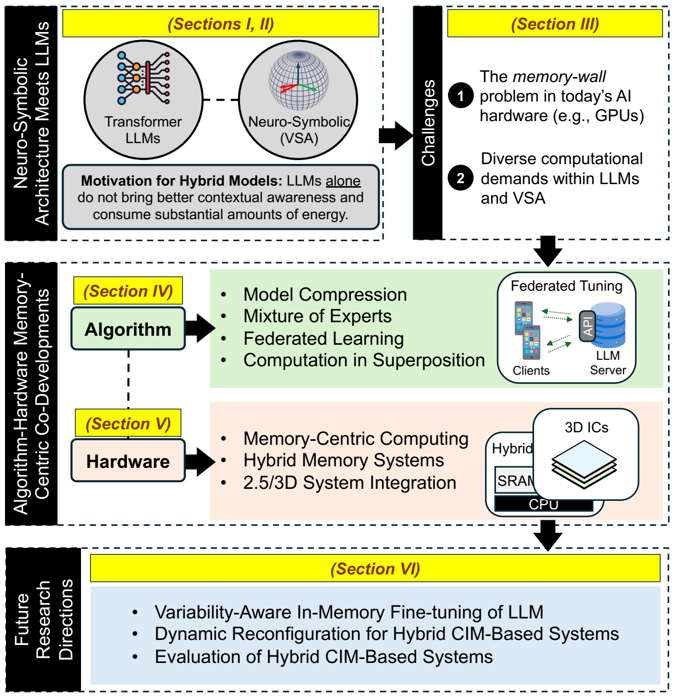

Neuro-Symbolic Architecture Meets LLMs
(Sections I, II)
Transformer LLMs
Neuro-Symbolic (VSA)
Motivation for Hybrid Models: LLMs alone do not bring better contextual awareness and consume substantial amounts of energy.
Challenges
(Section III)
1 The memory-wall problem in today's AI hardware (e.g., GPUs)
2 Diverse computational demands within LLMs and VSA
Algorithm-Hardware Memory-Centric Co-Developments
(Section IV)
Algorithm
Model Compression
Mixture of Experts
Federated Learning
Computation in Superposition
Federated Tuning
API
Clients
LLM Server
(Section V)
Hardware
Memory-Centric Computing
Hybrid Memory Systems
2.5/3D System Integration
Hybrid
SRAM
CPU
3D ICs
Future Research Directions
(Section VI)
Variability-Aware In-Memory Fine-tuning of LLM
Dynamic Reconfiguration for Hybrid CIM-Based Systems
Evaluation of Hybrid CIM-Based Systems

Fig. 1. **Overview of Research Topics.** This paper aims to identify and understand the system challenges, algorithm and hardware optimization solutions, and research opportunities of LLM-powered neuro-symbolic systems.

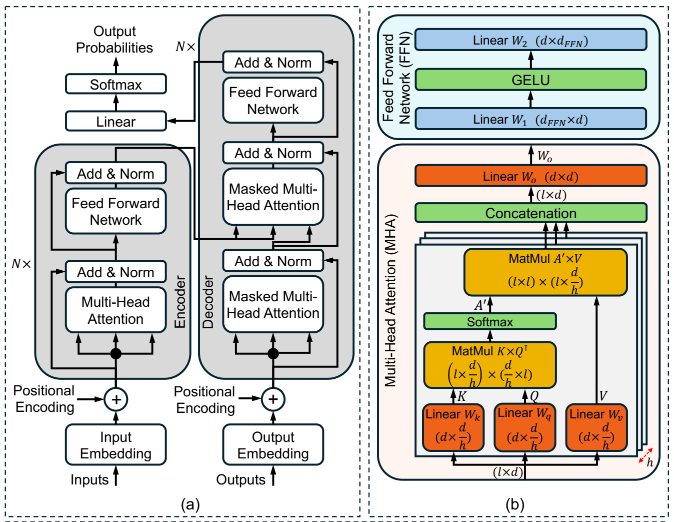
and compute-near-memory approaches that offer significant improvements in speed and energy efficiency. We also discuss the efficient integration of memory systems with logic circuits through 2.5D/3D stacked architectures. Additionally, we propose potential research directions for MCC to further explore its benefits. We anticipate that this work will serve as a "call to action" for interdisciplinary research aimed at enhancing the efficiency of hybrid cognitive models and promoting their adoption in resource-constrained AI systems.

The remainder of the paper is organized as follows: Section II provides the necessary background information. Section III addresses the key challenges encountered by the hybrid cognitive approach. Memory-centric solutions are detailed in Sections IV and V, with Section IV focusing on algorithmic optimizations and Section V covering advancements in hardware. Section VI outlines research directions for future exploration, while Section VII offers the concluding remarks.

II. Background

A. Transformer Architecture

The success of LLMs and foundation models in general can be largely attributed to the Transformer architecture, introduced by Vaswani et al. in 2017 [5]. This architecture has gained popularity due to its self-attention mechanism, which enables capturing long-range dependencies in input features without the need for complicated recurrent layers. A typical Transformer architecture is composed of a layered encoder-decoder structure, shown in Fig. 2(a). Each layer comprises a multi-head attention (MHA) mechanism followed by a feed-forward neural network (FFN), with residual connections and


Output Probabilities
Softmax
Linear
N×
Add & Norm
Feed Forward Network
Add & Norm
Multi-Head Attention
Encoder
Positional Encoding
Input Embedding
Inputs
Add & Norm
Feed Forward Network
Add & Norm
Masked Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Decoder
Positional Encoding
Output Embedding
Outputs
(a)
Feed Forward Network (FFN)
Linear W_2 $(d \times d_{FFN})$
GELU
Linear W_1 $(d_{FFN} \times d)$
Multi-Head Attention (MHA)
W_o
Linear W_o $(d \times d)$
$(l \times d)$
Concatenation
MatMul $A' \times V$
$(l \times l) \times (l \times \frac{d}{h})$
A'
Softmax
MatMul $K \times Q^T$
$(l \times \frac{d}{h}) \times (\frac{d}{h} \times l)$
K
Q
V
Linear W_k $(d \times \frac{d}{h})$
Linear W_q $(d \times \frac{d}{h})$
Linear W_v $(d \times \frac{d}{h})$
h
$(l \times d)$
(b)

Symbol	Parameter	BERT-Large	GPT-2	GPT-3	Megatron-Turing NLG	PaLM 540B
N	# Layers	24	12	96	105	118
d	Model dimension	1024	768	12288	2048	18432
h	# Heads	16	12	96	128	48
d_{FFN}	FFN dimension	4096	3072	49152	8192	73728

(c)

Fig. 2. **Transformer Background.** (a) The encoder-decoder structure [5]. (b) The operators of MHA and FFN. (c) Example parameter configurations.

layer normalization applied to each sub-layer. The decoder has an additional cross-attention mechanism to attend to the encoder's output. The input to the Transformer is first embedded into a continuous vector space and then augmented with positional encodings to preserve the order of the sequence. LLMs are usually pre-trained with input sequence lengths of around 500 to 2,000 tokens.

The computational workload of the Transformer is known to be dominated by the operators in the MHA and FFN blocks. The operators of these blocks are illustrated in Fig. 2(b). MHA consists of six linear operators, four of which are identical weight-to-activation matrix multiplications (labeled as Linear), and the remaining two of which are activation-to-activation matrix multiplications (labeled as MatMul). FFN consists of two Linear matrix operations with a non-linear layer (e.g., GELU) placed between them. LLMs may adopt various configurations of these operators, leading to differences in model capabilities and performance. Fig. 2(c) shows the configurations of these operators in commonly used LLMs.

B. Benefits of Combining NeSy Models with LLMs

There are several forms of NeSy models, each with unique approaches, as detailed in state-of-the-art taxonomies [20]. Combining NeSy models with LLMs offers significant benefits. First, hybrid models reduce computational resources and data (or label) storage needs; for instance, [21] shows that such hybrid models handle learning and reasoning tasks efficiently with significantly less data labeling and processing. Second, they create more effective AI for adversarial conditions, as described in [22]. An integrated NeSy-LLM model enables complex task-solving through unambiguous intent specification, task decomposition into subtasks solvable by individual LLMs, program synthesis for composing LLMs, and NeSy inference for scheduling and combining the results of different LLMs. Third, integrating NeSy models with LLMs enhances planning and scheduling; [23] demonstrates that NeSy models overcome LLMs' limitations by providing structured and context-aware planning capabilities. These advantages, and several others, motivate further investigation into the benefits and challenges of such a hybrid approach.

C. *Vector-Symbolic Architecture (VSA)*

For brevity, we use VSA as the example NeSy model throughout this paper. VSA is a computational framework designed to represent, manipulate, and reason about information using high-dimensional vectors [11]. The computations of VSA, shown in Fig. 3(a), include encoding, where information is transformed into high-dimensional data using an embedding matrix. This matrix can consist of either randomly generated vectors, often referred to as item vectors, or engineered vectors derived from neural networks [24]. Using generated vectors, VSA also employs an algebraic program, which builds data structures and algorithms through operations like binding, bundling, and permutation [10]. These operations enable the combination and aggregation of information in meaningful ways. Finally, associative memory search, shown in Fig. 3(b), is used to find the nearest vector to a query, facilitating efficient retrieval of symbolic information [25].

A key example of a VSA program for reasoning is the resonator network, which aims to factorize a compound vector representation into its atomic constituents [26]. This process, shown in Fig. 3(c), involves iteratively adjusting candidate vectors to minimize the difference between their combined representation and the target compound vector. By leveraging unbinding and dot-product operations, the resonator network efficiently decomposes complex data structures, revealing the underlying components. This approach demonstrates how VSA can manage intricate geometric relationships within data, offering robust solutions for tasks like visual reasoning [27].

While VSA excels at manipulating and reasoning with symbolic information, it typically assumes that the input data is intrinsically structured and symbolic in nature. This limitation prevents VSA from capturing the rich semantics and contextual nuances of unstructured data, such as sensory data. Unlike VSA, LLMs are designed to learn and extract meaning from vast amounts of unstructured text. Therefore, integrating VSA and LLMs can leverage the strengths of both approaches, leading to more robust and versatile AI systems.

D. *Memory Technologies*

Memory-centric platforms are developed using various memory technologies, including both volatile and nonvolatile devices. In this paper, we explore three examples of widely used random-access memory (RAM) technologies, namely static RAM (SRAM), dynamic RAM (DRAM), and resistive RAM (RRAM). Readers can refer to the following papers for a comprehensive treatment of memory technologies [28], [29].

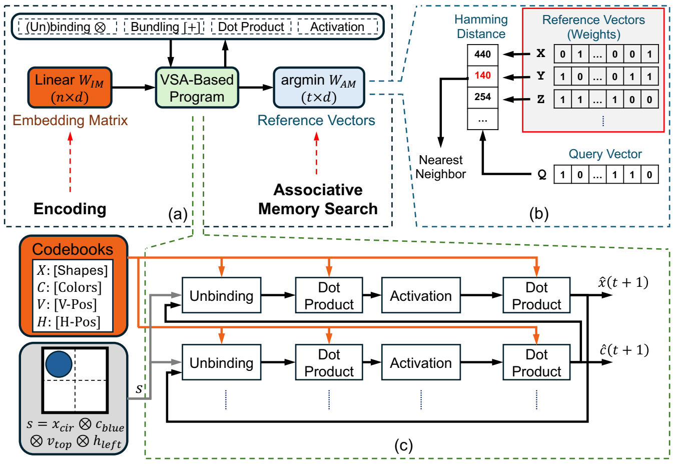


Fig. 3. **VSA Background.** (a) A flow diagram describing computations in VSA. (b) Associative search finds the nearest vector to a query. (c) A VSA-based reasoning program for factoring an object's vector [26].

SRAM. It refers to a memory technology that employs bistable latching circuitry to store each bit. An SRAM cell typically consists of six transistors (6T), arranged as a pair of cross-coupled inverters to hold a single bit of data. Although the 6T cell configuration occupies a considerable area and is limited to read and write operations, it offers advantages due to its relatively low write energy and high endurance [30]. These characteristics make SRAM an excellent choice for caching and storing intermediate activations that are frequently updated during operation. Moreover, advancements in SRAM platforms can further extend their function, aiming to enable logic computations within the cells or at the periphery [31].

DRAM. The DRAM technology stores information using capacitors. A typical DRAM cell consists of a capacitor and an access transistor, arranged in a one-transistor-one-capacitor (1T1C) configuration. DRAM cells are organized into arrays, which are further grouped into memory banks. Each bank operates independently, enabling parallel access and efficient data management, thereby enhancing overall memory throughput. This hierarchical structure of cells, arrays, and banks maximizes both density and performance, making DRAM a suitable choice for main system memory where large amounts of data need to be stored. Advancements in DRAM, such as higher bandwidth interfaces, reduced latency, and integration with processing elements, can significantly boost its potential for DRAM-based acceleration in various high-performance computing applications [32].

RRAM. It is a non-volatile memory technology that stores data by altering the resistance across a dielectric solid-state material. A RRAM cell is composed of a metal-insulator-metal structure, where the insulator is typically a thin film of a metal oxide or other resistive switching material. The cell operates by modulating the resistance state of the insulator, storing binary data as high resistance (binary 0) and low resistance (binary 1). RRAM cells are organized in a crossbar architecture, which maps the elements of a weight matrix into its resistive cells. For in-RRAM vector-matrix multiplication,

TABLE I
COMPARISON OF MEMORY TECHNOLOGIES. DATA OBTAINED FROM [29], [30]. ENDURANCE UNIT: # VALID WRITE/ERASE CYCLES BEFORE FAILURE.

Metric	Memory Technology		
	SRAM	DRAM	RRAM
Cell Structure	6T	1T1C	1T1R
Volatility	Yes	Yes	No
Write Voltage	<1 V	<1 V	<3 V
Write Energy	$\sim$ fJ	$\sim$ 10 fJ	$\sim$ 1 pJ
Write Speed	$\sim$ ns	$\sim$ 10 ns	$\sim$ 10 ns
Read Speed	$\sim$ ns	$\sim$ 3 ns	$\sim$ 10 ns
Endurance	10^{16}	10^{16}	$>10^{7}$

an input vector is encoded as voltages applied to the word lines (rows) of the crossbar. This configuration generates the output vector as currents along the bit lines (columns), which can be processed efficiently for various computational tasks [33].

Overall, different memory technologies, each with its unique properties, play crucial roles in advancing LLMs and NeSy systems. A comparison between the above technologies in terms of volatility, speed, energy, and endurance is provided in Table I. This table highlights the trade-offs between these technologies, helping to inform optimal memory choices and architectural organization for cognitive applications.

III. KEY COMPUTATIONAL CHALLENGES OF LLM-POWERED HYBRID MODELS

Integrating LLMs with NeSy models like VSA offers promising advancements in AI by combining the strengths of both approaches. However, this paradigm introduces several computational challenges that must be addressed to realize its full potential. This section describes two key challenges: the memory-wall problem (Section III-A), which hampers scalability and efficiency, and the diverse computational characteristics inherent in LLMs and VSA (Section III-B). Understanding and overcoming these challenges is essential for developing robust and efficient hybrid models.

A. The Memory-Wall Problem

The memory-wall problem, a term coined by Wulf and McKee in [17], refers to the bottleneck created by the growing disparity between processor speed and memory-access speed, depicted in Fig. 4. Over the past 20 years, the peak performance of server hardware has been scaling at a rate of 3.0× every 2 years, while DRAM and interconnect bandwidth have only scaled at 1.6 and 1.4 times every 2 years, respectively [18]. This disparity has shifted the main performance bottleneck from compute to memory bandwidth, particularly in AI applications involving LLMs. This increasing pressure on memory bandwidth also applies to major vector-symbolic algorithms, which rely heavily on rapid memory access to efficiently process and search through a database of high-dimensional data [34]. Therefore, reducing the impact of this memory wall—primarily by minimizing memory latency and reducing data transfer—is crucial for the sustainable advancement of AI technologies.

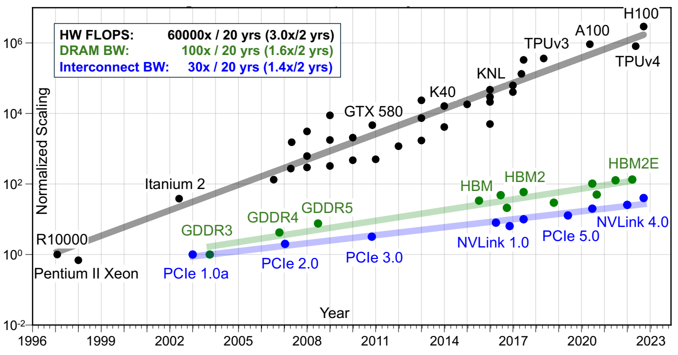


Fig. 4. **The Memory-Wall Problem.** The scaling of the memory and interconnect bandwidth, as well as peak processor FLOPS (floating-point operations per second). As can be seen, there is growing disparity between processor speed and memory/interconnect bandwith. Data collected from [18].

B. Diverse Computational Demands

Another notable challenge is the diverse computational characteristics within LLMs and VSA. This diversity is evident in attributes such as noise tolerance, intermediate caching needs, and data sparsity. For instance, certain LLM operations, like MatMul in MHA (Fig. 2(b)), require high precision and caching to maintain context accuracy [35], [36]. In contrast, tasks like associative search (Fig. 3(b)) can tolerate lower precision, allowing for faster processing with reduced memory overhead [25]. Similarly, VSA involves both sparse and dense computations: sparse operations manage high-dimensional data efficiently with minimal active elements, while dense computations require substantial resources to process fully populated matrices [37]. These varying demands necessitate adaptable design and memory systems that optimize performance across different computational tasks, balancing precision, caching requirements, and data sparsity.

IV. MEMORY-CENTRIC ALGORITHMIC OPTIMIZATIONS

This section discusses four algorithmic optimizations for LLMs and NeSy systems, namely model compression (Section IV-A), mixture of experts (Section IV-B), federated learning (Section IV-C), and computation in superposition (Section IV-D). Each method addresses critical aspects of memory and computational efficiency, aiming to alleviate the challenges posed earlier. By reducing model size, improving resource allocation, and superposing computations, these optimizations contribute to more scalable and efficient systems.

A. Model Compression

Considerable research efforts have been devoted to developing effective compression techniques, aiming to reduce the substantial memory footprint and computational demands of the Transformer model [38]. The primary focus has been on post-training quantization (PTQ), which involves reducing the precision of the weights in MHA and FFN from 32-bit or 16-bit floating-point to lower bit-width representations (e.g., 8-

bit integers), performed post the LLM's training phase[1] [40]. However, weight-only PTQ often leads to large accuracy degradation and may become ineffective, especially as the length of the input sequence is significantly increased [41].

To overcome such limitations, quantizing both weights and dynamic activations (e.g., MatMuls) has become a promising approach. The core idea of this approach is to provide special treatment to *salient* weights using insights from their respective activation channels [42]. Evaluations based on Llama 2-70B show that activation-aware PTQ can reduce the memory requirements by up to 4-8× and inference latency by 2-4× compared to full-precision models, providing opportunities for efficient implementation of LLMs on the edge [43].

Quantization in NeSy systems serves a distinct purpose compared to its role in LLMs, especially as symbolic architectures typically work with explicitly defined discrete quantities such as rules and attributes. Perhaps the most common form of quantization in these systems is concept quantization [44]. In VSA, this involves discretizing the neural network's output by mapping it to the nearest neighbor from a set of pre-defined values or symbols [24]. In essence, quantization in NeSy systems can be understood as a function whose performance is influenced by parameters in the symbolic space, such as the length and number of vector-symbolic representations.

B. Mixture of Experts (MoE)

MoE is a neural network architecture that aims to increase computational efficiency by effectively decoupling the parameter count of the learning model and the computation's floating-point operations required for training and inference [45]. Unlike traditional dense models where all parameters are used for every input, MoE models use a gating function to dynamically select the most relevant "experts" (sub-networks) based on the input data. Fig. 5(a) shows the embedding of MoE layers, which typically replace FFN layers in a Transformer model. During operation, the gating network determines which experts are activated for a given input, ensuring that only a fraction of the total parameters are utilized at any time. This selective activation helps in reducing the computational load per inference step, despite the drastic increase in the total number of parameters [46].

Large MoE models have been shown to improve performance on various tasks, often surpassing traditional dense LLMs. For instance, large MoE models like ST-MoE and Mixtral have demonstrated superior performance while using significantly fewer operations compared to dense counterparts. Specifically, ST-MoE, using 20× and 40× fewer operations in training and inference, surpasses PaLM 540B in performance [47]. Mixtral 8x7B, while only actively using 13B parameters during inference, performs on par with Llama 2-70B models across various evaluation benchmarks [48].

Besides, MoE models excel in NeSy reasoning problems by efficiently managing complex symbolic structures such as trees and graphs. For instance, the integration of MoE with advanced NeSy models like the Differentiable Tree Machine [49], which learns tree operations using a combination of Transformers and distributed symbolic representations, demonstrates its ability to handle intricate tree operations while employing a reduced number of computations [50]. Overall, advancements offered by MoE not only reduce computational costs but also enhance the model's reasoning capabilities by allowing for more specialized and efficient processing of complex symbolic information.

[1]Quantization can also be applied during the training phase, i.e., quantization-aware training (QAT) [39]. However, QAT becomes impractical for models with billions of parameters due to excessive training costs.

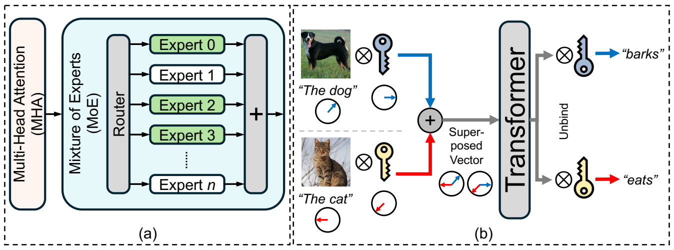


Fig. 5. **Illustration of Algorithmic Optimizations.** (a) Mixture of Experts: During inference, a router aims to choose concise and proficient expert networks that depending on the input [41]. (b) Computation in Superposition: Input samples are bound with high-dimensional keys to project the samples into quasi-orthogonal subspaces. The results of the individual samples are retrieved at the end of the network by unbinding with corresponding keys [52].

Note that reducing the computation load per inference by using MoE comes at a significant cost. MoE models are known to have ineffective utilization of on-chip memory, as a significant portion of the model's parameters remain dormant during inference. This problem is exacerbated by the large memory demands of LLMs, which have far outpaced the memory capacity of contemporary AI hardware [51]. These challenges are a motivation to the research community to build new memory-centric hardware architectures targeting sparse data-intensive workloads (discussed later in Section V-A).

C. Federated Learning (FL)

FL is a collaborative machine learning technique that enables multiple decentralized devices to train a shared model without exchanging their local data. Instead of sending raw data to a central server, each participant (or a client) trains the model locally using their own data and only shares the updated model parameters or gradients. These updates are then aggregated centrally, often using averaging methods such as FedAvg [53], to improve the global model. FL is especially advantageous in edge computing environments, as it ensures data privacy and addresses the challenges of limited computational resources [54].

With the rise of pre-trained LLMs, the application of FL has taken a new dimension, aiming to provide decentralized methods that efficiently fine-tune these models. Fine-tuning, required to adapt LLMs to perform specific tasks or improve their performance in real-world scenarios, relies heavily on massive data generated through user interactions with both the environment and the large model [55], [56]. Adopting FL in this context seems intuitive but remains challenging, as clients have restricted access to LLM parameters and they are often ill-equipped to support the computational and storage demands of these large models.

To tackle the above challenges, federated black-box prompt tuning is increasingly used, allowing clients to treat the LLM as a black-box and focus on optimizing prompts locally using proxy data distributions and gradient-free optimization methods [57], [58]. This approach is highly effective since clients are not required to store or access the LLM parameters, and only inference of the model is conducted during local optimization. When used to fine-tune Llama 2-7B, this federated approach reduces the number of trainable parameters to only 500, and hence the communication cost in one round is also reduced to only 4KB [57].

As NeSy models have grown sophisticated and capable, integrating them with FL strategies has emerged as an important research subject. In this context, the application of FL aims to handle client heterogeneities symbolically, enabling data-driven clients to develop personalized symbolic reasoning capabilities [59]. This approach allows each client to adapt the shared model while concurrently creating unique symbolic rules or logical properties that reflect the client's demands or characteristics [60], [61]. Communication efficiency in this federated scheme is notably enhanced by the ability to compress complex rule patterns into latent variables, significantly reducing the amount of data exchanged between clients and the central server. This is particularly achieved by adopting compact symbolic representations like VSA, which is well-suited for robust communication and efficient storage of symbolic information [62].

D. Computation in Superposition (CIS)

CIS has emerged as a promising paradigm that significantly enhances the efficiency of LLM systems. This paradigm is inspired by the principle of superposition from quantum computing, allowing multiple computational pathways to be explored simultaneously [63]. This capability offers substantial speedup benefits, particularly for LLM frameworks that require concurrent computation over extensive long-form prompts. One of such frameworks is retrieval-augmented generation (RAG), which seeks to augment an LLM with access to a dynamic, curated knowledge base to improve its output [64]. RAG involves a process that retrieves information from this knowledge base and combines it with the existing knowledge of the LLM. Prompting for RAG is often optimized such that tokens can be processed semi-independently, thereby providing an excellent opportunity for acceleration by superposition [65]. By enabling CIS within LLM's self-attention, it is possible to significantly reduce the time and resources needed for model inference.

Advances in this context essentially focus on developing and analyzing suitable mathematical representations that facilitate superposition, rather than modifying the underlying self-attention mechanism [66]. It is no surprise that the algebraic nature of VSA presents an opportunity to realize CIS by bundling multiple computational channels [52]. Specifically, VSA offers algebraic methods for the encoding and binding of multiple attention tokens and activation operations, as illustrated in Fig. 5(b). VSAs can also represent data structures like directed graphs, which are typically used to model dependencies among tokens [10], [65]. Using distributed vector representations is highly effective in reducing distortion caused by inter-channel interferences, especially as the length of vectors is significantly increased.

The superposition capability of VSA not only speeds up inference computations by processing multiple inputs in parallel but also enhances the efficiency and dynamics of search in NeSy systems. This is particularly evident in rule-based systems used for factorization, described earlier in Section II-C. This system relies on superposition to efficiently search through the combinatoric solution space without directly enumerating all possible factorizations [26]. This approach reduces computational overhead, enabling rapid convergence to accurate solutions. Moreover, the inherent robustness of this approach allows the system to tolerate noise and partial information, making it adaptable to real-world scenarios where data is often incomplete or ambiguous [67].

V. Memory-Centric Hardware Developments

The integration of Transformers with NeSy models aligns closely with the principles of MCC. The central role of memory in computation can be further enhanced to improve efficiency and adaptability. This section discusses hardware developments in this direction, covering methodologies of MCC (Section V-A), new trends in building hybrid memory systems (Section V-B), and logic-memory integration through 2.5D/3D stacked architectures (Section V-C).

A. Memory-Centric Computing (MCC)

MCC is a promising paradigm that addresses the memory-wall problem in AI hardware accelerators. With MCC, key operations in LLMs and NeSy methods, such as MatMul and associative memory search, could be performed in the mixed-signal domain within memory sub-arrays (compute-in-memory "CIM"), or in the outer digital domain directly connected to memory banks (compute-near-memory "CNM"), or via content addressable memories ("CAM"), leading to significant advances in energy efficiency and throughput [34], [68], [69]. Fig. 6 illustrates the difference between CIM and CNM. The boundary between CIM and CNM is drawn based on a combination of factors, including memory technology, computational characteristics of dataflows, and target performance—necessary for addressing the diverse computational demands of AI workloads (refer to Section III-B).

CIM is particularly suitable for accelerating weight-stationary dataflows, designed to hold pre-computed parameters in memory cells while arbitrary input vectors are brought into these cells for parallel computation. For instance, the Linear blocks shown in Fig. 2(b) can be mapped to an in-memory matrix-vector multiplication, which is commonly implemented in RRAM [70], [71]. In VSA, RRAM is also employed to improve the efficiency of memory search, where RRAM devices are configured to output the closest match to an input vector, i.e., implement content-addressable memory

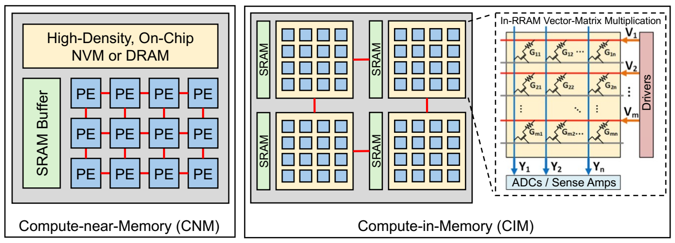

High-Density, On-Chip NVM or DRAM
SRAM Buffer
PE
Compute-near-Memory (CNM)
SRAM
In-RRAM Vector-Matrix Multiplication
V_1
V_2
V_m
G_{11} G_{12} G_{1n}
G_{21} G_{22} G_{2n}
G_{m1} G_{m2} G_{mn}
Drivers
Y_1 Y_2 Y_n
ADCs / Sense Amps
Compute-in-Memory (CIM)

Fig. 6. **Memory-Centric Computing Approaches.**

operations [72]. RRAM is a nonvolatile memory technology and thereby can memorize and store parameters without external power supply.

Despite its low-power consumption, RRAM suffers from intrinsic problems, particularly process variations and lifetime [73], [74]. Therefore, RRAM is impractical to use for high-precision computations and is not ideal for dynamic dataflows whose inputs and parameters are generated at runtime, such as MatMul blocks in Fig. 2(b). The low endurance of RRAM reduces its lifetime, especially when read-write operations become more frequent (Table I). This challenge has led to the development of SRAM-based CIM solutions, which offer higher endurance and better support for dynamic processing needs, despite being less favorable in terms of energy efficiency and memory density [68]. To enable CIM, SRAM is typically modified from the conventional 6T bit cell, thus enabling in-place memory updates and logic operations [75]. Furthermore, the mixed-signal nature of these systems allows to tune precision, as SRAM sub-arrays can be designed to support both floating-point and integer operations [76]. This adaptability is especially crucial in LLMs, where precision impacts both performance and energy consumption.

Emerging data-intensive workloads often involve sparse computations that face challenges related to latency and memory capacity. Large MoE models, discussed in Section IV-B, illustrate these complexities, where expert activation is inherently sparse and executed dynamically at runtime. For such applications, DRAM-based CNM presents a compelling solution by offering a hardware architecture capable of efficiently storing large data volumes and reducing latency through localized processing [77]. Modern DRAM architectures are structured hierarchically with enhanced memory banks integrated via high-bandwidth memory (HBM) interfaces [78]. Research focused on optimizing DRAM for CNM includes incorporating advanced row buffer management schemes, integrating specialized units for bulk-bitwise operations like filtering and aggregation, and deploying various low-power states for efficient in-memory computations [79]. The growing interest in implementing diverse DRAM optimizations, capitalizing on its high density and endurance, enables integrating CNM across a broader spectrum of applications [80].

B. Hybrid Memory Systems

Research into hybrid memory systems is advancing rapidly, offering optimized performance, efficiency, and adaptability by integrating diverse memory technologies on a single chip. These innovations cater to the specific demands of AI edge applications, which often face constraints like limited power and high precision [81]. For example, in event-based target tracking systems, integrating RRAM and SRAM on the same chip leverages their complementary strengths [82], [83]. RRAM provides CIM capabilities that enhance speed and reduce power consumption, making it ideal for high-speed tracking. Meanwhile, SRAM supports high-precision near-memory computations, ensuring reliable target identification. This hybrid integration balances power efficiency with the occasional need for high-performance computations—an approach that can also be beneficial in other flows, such as LLM inference and factorization of NeSy representations [35], [71].

C. 2.5D/3D System Integration

Compute demands and memory footprints for the envisioned NeSy-LLM systems may soon exceed what today's 2D chips could possibly offer. As two-dimensional scaling approaches its physical and economic limits, the shift to three-dimensional integration and heterogeneous system design becomes essential to meet diverse design targets in a cost-effective manner [84]. Breaking the single-die limits also brings new design opportunities with component technologies (new memories, new logic, new interconnects) specifically optimized for the algorithmic needs in NeSy and LLM models. Memory-centric architectures can readily benefit from both capacity and bandwidth advances brought by 3D integration, with HBM being a prime example in the recent success of LLMs. Bringing computations closer to or immersed in HBM stacks will require new chiplet architectures where specialized cores, DRAM die stacks, and horizontal/vertical interconnects are co-designed based on model requirements.

With the 2.5D/3D integration becoming a vital platform technology going forward, designers are having an increasingly vast design space. For example, it becomes feasible to map the requirements from the diverse compute kernels seen in LLM and NeSy models to a variety of logic nodes, memory types, and specialty devices not available in silicon logic process. The mixture of technologies and designs, which is then realized by the right combination of manufacturing and assembly techniques, allows much more fine-grained system-technology co-optimization. With co-designs, it is possible to envision a set of NeSy-friendly and LLM-friendly nanokernels for chip and chiplet integration, leveraging advancements in new semiconductor device technologies [85]. Examples include scaled oxide-semiconductor FETs, low-dimensional semiconductor FETs, as well as various volatile and non-volatile memories.

The hybrid nature of NeSy, LLM, and NeSy-LLM models in terms of dataflows and model components will further define unique requirements and drive optimizations for the inter-connectivity in the tailored 2.5D/3D systems, which can be quite different than that in conventional 2D systems. For substrate interconnects in 2.5D/3D chiplets [86], the progress in interposers and wafer-level fan-out layers will continue to define the dataflow contraints and costs at the system level. In the third dimension, as micro-bumps, through-silicon vias (TSV), hybrid bonding, and monolithic inter-layer vias (ILV) are continuously advancing in density and reliability, the vertical connectivity may become increasingly useful, enabling new nanokernel designs and chiplet architectures [87].

Recent research literature showcases several examples of stacked designs optimized for efficient kernel processing in Transformer and VSA models. One notable example is H3DAtten, a heterogeneous 3D integrated architecture specifically designed for MHA kernels in Vision Transformer models [36]. Leveraging fine-pitch hybrid bonding and TSVs, H3DAtten vertically stacks dies from different process nodes, combining a 40nm RRAM-based analog CIM dataflow with a 16nm SRAM-based logic. This heterogeneous integration is also applied to optimize VSA-based factorization, thus demonstrating the benefits of 3D stacked designs in symbolic reasoning [88]. Other examples include a 3D stacked implementation of RRAM-based one-shot learning [89] and a 3D integration of various transistor and memory technologies for efficient VSA-based pattern recognition [90].

VI. Future Directions and Research Opportunities

Below is a list of research topics that could further advance the memory-centric approaches presented in this paper.

A. *Variability-Aware In-Memory Fine-tuning of LLM*

Recent research has explored techniques to make neural networks more robust when implemented on RRAM devices, which suffer from inherent non-idealities due to stochastic process variations. Variability-aware training algorithms and hardware/software co-design methods have been developed to compensate for RRAM conductance variations and maximize neural network accuracy [91], [92]. While these are promising approaches, their application in the fine-tuning of LLMs remains under-explored. In-memory fine-tuning of LLMs on RRAM devices presents unique challenges due to the scale and sensitivity of these models to minute variations in weight values. Addressing these challenges requires advanced variability-aware techniques that not only mitigate hardware-induced inaccuracies but also preserve the linguistic and contextual nuances captured by the LLM during its initial training. Innovative solutions, such as adaptive weight scaling, error correction schemes, and dynamic reconfiguration of memory arrays, are essential to enable efficient and reliable in-memory fine-tuning. By leveraging these techniques, it is possible to achieve high-performance LLM deployments on energy-efficient RRAM-based systems.

B. *Dynamic Reconfiguration for Hybrid CIM-Based Systems*

Runtime reconfiguration of CIM fabrics is an emerging and promising research area, driven by the increasing demand for hybrid cognitive computing solutions. Existing reconfiguration techniques have largely focused on adapting single CIM-based designs to support various neural network models [93]. These approaches have shown success in enabling reconfigurable weight mapping and dynamic weight reloading, optimizing the handling of diverse input data. However, the next frontier in this field lies in extending reconfiguration capabilities to CIM-based systems incorporating multiple CIM fabrics and technologies. Achieving this will require a holistic approach combining several key strategies: developing modular memory subsystems that support multi-precision computing [94], enabling dynamic data transfer between CIM fabrics through adaptive network-on-chip solutions, and creating technology-aware reconfiguration algorithms capable of seamlessly coordinating operations across heterogeneous memory technologies and workloads [95]. By advancing these strategies, we can better support hybrid cognitive models.

C. *Evaluation of Hybrid CIM-Based Systems*

Evaluating hybrid CIM-based systems necessitates a hierarchical approach that integrates optimization and evaluation characteristics across both memory fabrics and chip-level effects. Traditional evaluation frameworks, such as DNN+NeuroSim [96], have been effective for single CIM fabrics but are insufficient for chips incorporating heterogeneous CIM fabrics. To address this, a comprehensive evaluation framework must be developed that accounts for the complexities introduced by varying CIM technologies within a single chip. This framework should include detailed assessments of interconnect performance, caching strategies, and the interaction between different CIM fabrics. Incorporating state-of-the-art benchmarks like MLPerf [97] will be crucial for optimizing these systems, as these benchmarks provide standardized metrics for evaluation and comparison.

VII. Conclusion

In this paper, we have explored memory-centric design and optimization strategies for cognitive systems integrating LLMs and NeSy models. We addressed key computational challenges, including the memory wall problem and diverse computational characteristics. We discussed various algorithmic optimizations, such as model compression, mixture of experts, federated learning, and computation in superposition. Additionally, we presented hardware developments for memory-centric computing, including effective integration methods like 2.5D/3D-stacked design and chiplet-based approaches. Finally, we outlined potential research directions to further enhance efficiency and scalability. This paper represents a significant step towards the efficient realization of advanced cognition.

Acknowledgements

We gratefully acknowledge support from CoCoSys, one of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA; and from the US National Science Foundation (NSF) CCF-2328805.

References

[1] T. Brown *et al.*, “Language Models are Few-Shot Learners,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.

[2] J. Achiam *et al.*, “GPT-4 Technical Report,” *arXiv preprint arXiv:2303.08774*, 2023.

[3] J. Devlin *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] J. Yang *et al.*, "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 18, no. 6, pp. 1–32, 2024.

[5] A. Vaswani *et al.*, "Attention is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[6] H. Fan *et al.*, "Embodied Intelligence in Manufacturing: Leveraging Large Language Models for Autonomous Industrial Robotics," *Journal of Intelligent Manufacturing*, pp. 1–17, 2024.

[7] A. J. Thirunavukarasu *et al.*, "Large Language Models in Medicine," *Nature Medicine*, vol. 29, no. 8, pp. 1930–1940, 2023.

[8] E. Kasneci *et al.*, "ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[9] A. d. Garcez and L. C. Lamb, "Neurosymbolic AI: The 3rd Wave," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12 387–12 406, 2023.

[10] 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.

[11] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, pp. 139–159, 2009.

[12] F. Shi *et al.*, "Large Language Models Can Be Easily Distracted by Irrelevant Context," in *Proceedings of International Conference on Machine Learning (ICML)*, 2023, pp. 31 210–31 227.

[13] L. Sun *et al.*, "TrustLLM: Trustworthiness in Large Language Models," *arXiv preprint arXiv:2401.05561*, 2024.

[14] K. Valmeekam *et al.*, "On the Planning Abilities of Large Language Models : A Critical Investigation," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 75 993–76 005, 2023.

[15] D. Patterson *et al.*, "Carbon Emissions and Large Neural Network Training," *arXiv preprint arXiv:2104.10350*, 2021.

[16] H. Xiong *et al.*, "Converging Paradigms: The Synergy of Symbolic and Connectionist AI in LLM-Empowered Autonomous Agents," *arXiv preprint arXiv:2407.08516*, 2024.

[17] W. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, 1995.

[18] A. Gholami *et al.*, "AI and Memory Wall," *IEEE Micro*, 2024.

[19] Z. Wan *et al.*, "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.

[20] 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.

[21] D. Cunnington *et al.*, "The Role of Foundation Models in Neuro-Symbolic Learning and Reasoning," *arXiv preprint arXiv:2402.01889*, 2024.

[22] S. Jha *et al.*, "Challenges and Opportunities in Neuro-Symbolic Composition of Foundation Models," in *Proceedings of IEEE Military Communications Conference (MILCOM)*, 2023, pp. 156–161.

[23] V. Pallagani *et al.*, "On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS)," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 34, 2024, pp. 432–444.

[24] M. Hersche *et al.*, "A Neuro-Vector-Symbolic Architecture for Solving Raven's Progressive Matrices," *Nature Machine Intelligence*, vol. 5, no. 4, pp. 363–375, 2023.

[25] M. Imani *et al.*, "Exploring Hyperdimensional Associative Memory," in *Proceedings of IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 445–456.

[26] 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.

[27] A. Renner *et al.*, "Neuromorphic Visual Scene Understanding with Resonator Networks," *Nature Machine Intelligence*, pp. 1–12, 2024.

[28] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory Devices and Applications for In-Memory Computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[29] A. Gebregiorgis *et al.*, "A Survey on Memory-Centric Computer Architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 4, pp. 1–50, 2022.

[30] S. Salahuddin, K. Ni, and S. Datta, "The Era of Hyper-Scaling in Electronics," *Nature Electronics*, vol. 1, no. 8, pp. 442–450, 2018.

[31] Q. Dong *et al.*, "A 351TOPS/W and 372.4 GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 242–244.

[32] M. Zhou *et al.*, "TransPIM: A Memory-based Acceleration via Software-Hardware Co-Design for Transformer," in *Proceedings of IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 1071–1085.

[33] X. Yang *et al.*, "Research Progress on Memristor: From Synapses to Computing Systems," *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, vol. 69, no. 5, pp. 1845–1857, 2022.

[34] S. Shou *et al.*, "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)*. IEEE, 2023, pp. 1–9.

[35] S. Liu *et al.*, "HARDSEA: Hybrid Analog-ReRAM Clustering and Digital-SRAM In-Memory Computing Accelerator for Dynamic Sparse Self-Attention in Transformer," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[36] W. Li *et al.*, "H3DAtten: Heterogeneous 3-D Integrated Hybrid Analog and Digital Compute-in-Memory Accelerator for Vision Transformer Self-Attention," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2023.

[37] D. Kleyko *et al.*, "Classification and Recall With Binary Hyperdimensional Computing: Tradeoffs in Choice of Density and Mapping Characteristics," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 29, no. 12, pp. 5880–5898, 2018.

[38] A. Gholami *et al.*, "A Survey of Quantization Methods for Efficient Neural Network Inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[39] Z. Liu *et al.*, "LLM-QAT: Data-Free Quantization Aware Training for Large Language Models," *arXiv preprint arXiv:2305.17888*, 2023.

[40] ——, "Post-Training Quantization for Vision Transformer," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 28 092–28 103, 2021.

[41] Z. Yuan *et al.*, "LLM Inference Unveiled: Survey and Roofline Model Insights," *arXiv preprint arXiv:2402.16363*, 2024.

[42] X. Wu *et al.*, "Block-Wise Mixed-Precision Quantization: Enabling High Efficiency for Practical ReRAM-based DNN Accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2024.

[43] X. Shen *et al.*, "Agile-Quant: Activation-Guided Quantization for Faster Inference of LLMs on the Edge," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 18 944–18 951.

[44] J. Mao *et al.*, "The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision," in *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[45] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty Years of Mixture of Experts," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, vol. 23, no. 8, pp. 1177–1193, 2012.

[46] W. Fedus, B. Zoph, and N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.

[47] B. Zoph *et al.*, "ST-MoE: Designing Stable and Transferable Sparse Expert Models," *arXiv preprint arXiv:2202.08906*, 2022.

[48] A. Q. Jiang *et al.*, "Mixtral of Experts," *arXiv preprint arXiv:2401.04088*, 2024.

[49] P. Soulos *et al.*, "Differentiable Tree Operations Promote Compositional Generalization," in *Proceedings of International Conference on Machine Learning (ICML)*, 2023, pp. 32 499–32 520.

[50] J. Thomm *et al.*, "Terminating Differentiable Tree Experts," *arXiv preprint arXiv:2407.02060*, 2024.

[51] H. Huang *et al.*, "Towards MoE Deployment: Mitigating Inefficiencies in Mixture-of-Expert (MoE) Inference," *arXiv preprint arXiv:2303.06182*, 2023.

[52] N. Menet *et al.*, "MIMONets: Multiple-Input-Multiple-Output Neural Networks Exploiting Computation in Superposition," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024.

[53] L. Collins *et al.*, "FedAvg with Fine Tuning: Local Updates Lead to Representation Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 10 572–10 586, 2022.

[54] X. Wang *et al.*, "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[55] Z. Tan *et al.*, "Democratizing Large Language Models via Personalized Parameter-Efficient Fine-tuning," *arXiv preprint arXiv:2402.04401*, 2024.

[56] Y. Cao *et al.*, "Aligning Large Language Models with Recommendation Knowledge," *arXiv preprint arXiv:2404.00245*, 2024.

[57] J. Sun *et al.*, "FedBPT: Efficient Federated Black-box Prompt Tuning for Large Language Models," in *Proceedings of International Conference on Machine Learning (ICML)*, 2024.

[58] J. Zhang *et al.*, "Towards Building The Federated GPT: Federated Instruction Tuning," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 6915–6919.

[59] P. Xing, S. Lu, and H. Yu, "Federated Neuro-Symbolic Learning," in *Proceedings of International Conference on Machine Learning (ICML)*, 2024.

[60] Z. An, T. T. Johnson, and M. Ma, "Formal Logic Enabled Personalized Federated Learning through Property Inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 10, 2024, pp. 10 882–10 890.

[61] Y. Zhang and H. Yu, "LR-XFL: Logical Reasoning-based Explainable Federated Learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 19, 2024, pp. 21 788–21 796.

[62] S. Zhang *et al.*, "On Hyperdimensional Computing-based Federated Learning: A Case Study," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, 2023, pp. 1–8.

[63] A. Manju and M. J. Nigam, "Applications of Quantum-Inspired Computational Intelligence: A Survey," *Artificial Intelligence Review*, vol. 42, pp. 79–156, 2014.

[64] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[65] T. Merth *et al.*, "Superposition Prompting: Improving and Accelerating Retrieval-Augmented Generation," in *Proceedings of International Conference on Machine Learning (ICML)*, 2024.

[66] A. Zou *et al.*, "Representation Engineering: A Top-Down Approach to AI Transparency," *arXiv preprint arXiv:2310.01405*, 2023.

[67] S. J. Kent *et al.*, "Resonator networks, 2: Factorization Performance and Capacity Compared to Optimization-Based Methods," *Neural computation*, vol. 32, no. 12, pp. 2332–2388, 2020.

[68] S. Yu *et al.*, "Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[69] B. Kim *et al.*, "The Breakthrough Memory Solutions for Improved Performance on LLM Inference," *IEEE Micro*, 2024.

[70] Z. Lu *et al.*, "An RRAM-Based Computing-in-Memory Architecture and Its Application in Accelerating Transformer Inference," *IEEE Transactions on Very Large Scale Integration Systems*, 2023.

[71] J. Langenegger *et al.*, "In-Memory Factorization of Holographic Perceptual Representations," *Nature Nanotechnology*, vol. 18, no. 5, pp. 479–485, 2023.

[72] R. Mao *et al.*, "Experimentally Validated Memristive Memory Augmented Neural Network with Efficient Hashing and Similarity Search," *Nature Communications*, vol. 13, no. 1, p. 6284, 2022.

[73] B. Crafton *et al.*, "Improving Compute In-Memory ECC Reliability with Successive Correction," in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 745–750.

[74] Z. Wan *et al.*, "Rram-ecc: Improving reliability of rram-based compute in-memory," in *13th Annual Non-Volatile Memories Workshop (NVMW)*, 2022.

[75] F. Tu *et al.*, "TranCIM: Full-Digital Bitline-Transpose CIM-based Sparse Transformer Accelerator With Pipeline/Parallel Reconfigurable Modes," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 6, pp. 1798–1809, 2023.

[76] H. Mori *et al.*, "A 4nm 6163-TOPS/W/b 4790-TOPS/mm^2/b SRAM Based Digital-Computing-in-Memory Macro Supporting Bit-Width Flexibility and Simultaneous MAC and Weight Update," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 132–134.

[77] Z. Du *et al.*, "SiDA: Sparsity-Inspired Data-Aware Serving for Efficient and Scalable Large Mixture-of-Experts Models," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 224–238, 2024.

[78] J. Kim and Y. Kim, "HBM: Memory Solution for Bandwidth-Hungry Processors," in *IEEE Hot Chips Symposium (HCS)*, 2014, pp. 1–24.

[79] J. D. Ferreira *et al.*, "pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2022.

[80] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A Modern Primer on Processing in Memory," in *Emerging Computing: From Devices to Systems: Looking Beyond Moore and Von Neumann*. Springer, 2022, pp. 171–243.

[81] M. Chang *et al.*, "A 40nm 60.64TOPS/W ECC-Capable Compute-in-Memory/Digital 2.25MB/768KB RRAM/SRAM System with Embedded Cortex M3 Microprocessor for Edge Recommendation Systems," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[82] ——, "A 73.53TOPS/W 14.74TOPS Heterogeneous RRAM In-Memory and SRAM Near-Memory SoC for Hybrid Frame and Event-Based Target Tracking," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, 2023, pp. 426–428.

[83] A. S. Lele *et al.*, "A Heterogeneous RRAM In-Memory and SRAM Near-Memory SoC for Fused Frame and Event-Based Target Identification and Tracking," *IEEE Journal of Solid-State Circuits*, 2023.

[84] K. Akarvardar and H.-S. P. Wong, "Technology Prospects for Data-Intensive Computing," *Proceedings of the IEEE*, vol. 111, no. 1, pp. 92–112, 2023.

[85] L. Zheng and H. Li, "CMOS+X Technologies for Neuro-Vector-Symbolic Computing," in *Proceedings of Device Research Conference (DRC)*, 2024, pp. 1–2.

[86] C. Douglas *et al.*, "Foundry Perspectives on 2.5 D/3D Integration and Roadmap," in *Proceedings of IEEE International Electron Devices Meeting (IEDM)*, 2021, pp. 3–7.

[87] S. Yu *et al.*, "Compute-in-Memory: From Device Innovation to 3D System Integration," in *Proceedings of IEEE European Solid-State Device Research Conference (ESSDERC)*, 2021, pp. 21–28.

[88] Z. Wan *et al.*, "H3DFact: Heterogeneous 3D Integrated CIM for Factorization with Holographic Perceptual Representations," in *Proceedings of IEEE/ACM Design, Automation & Test in Europe Conference (DATE)*, 2024, pp. 1–6.

[89] Y. Li *et al.*, "Monolithic Three-Dimensional Integration of RRAM-Based Hybrid Memory Architecture for One-Shot Learning," *Nature Communications*, vol. 14, no. 1, p. 7140, 2023.

[90] T. F. Wu *et al.*, "Hyperdimensional Computing Exploiting Carbon Nanotube FETs, Resistive RAM, and their Monolithic 3D Integration," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 53, no. 11, pp. 3183–3196, 2018.

[91] A. Glukhov *et al.*, "End-to-End Modeling of Variability-Aware Neural Networks Based on Resistive-Switching Memory Arrays," in *Proceedings of IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2022, pp. 1–5.

[92] T. Shahroodi *et al.*, "Swordfish: A Framework for Evaluating Deep Neural Network-based Basecalling using Computation-In-Memory with Non-Ideal Memristors," in *Proceedings of IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 1437–1452.

[93] A. Lu *et al.*, "A Runtime Reconfigurable Design of Compute-in-Memory–Based Hardware Accelerator for Deep Learning Inference," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 26, no. 6, pp. 1–18, 2021.

[94] F. Tu *et al.*, "A 28nm 29.2TFLOPS/W BF16 and 36.5TOPS/W INT8 Reconfigurable Digital CIM Processor with Unified FP/INT Pipeline and Bitwise In-Memory Booth Multiplication for Cloud Deep Learning Acceleration," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[95] S. Huang *et al.*, "Hardware-aware Quantization/Mapping Strategies for Compute-in-Memory Accelerators," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 3, pp. 1–23, 2023.

[96] X. Peng *et al.*, "DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies," in *Proceedings of IEEE international electron devices meeting (IEDM)*, 2019, pp. 32–5.

[97] V. J. Reddi *et al.*, "MLPerf Inference Benchmark," in *Proceedings of ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2020, pp. 446–459.