Harmony: A Hardware-Mapping Co-Exploration Framework for Hybrid CIM-based Vision Transformer Accelerator

Yihang ZUO[†], Zexin Fu[†‡], Cong Wang[†], Yuchao Wu[†], Jiayi Huang[†], Yuzhe Ma[†*]

[†]Thrust of Microelectronics, Hong Kong University of Science and Technology (Guangzhou)

[‡]Guangdong-Macao Joint Laboratory for Modular Chip Design and Testing, Hong Kong University of Science and Technology (Guangzhou)

{yzuo099,zexin.fu,cwang841,ywu092}@connect.hkust-gz.edu.cn, {hjy,yuzhema}@hkust-gz.edu.cn [*]Corresponding Author

***Abstract*—Computing-in-memory (CIM) architectures have successfully enhanced convolutional neural network (CNN) performance, but the automation of high-performance CIM-based transformer accelerators is still challenging. Specifically, the design space of hardware design and mapping is extremely large due to the complex model structure and data flow. To address this problem, we propose Harmony, a hardware and mapping co-exploration framework to optimize the hybrid CIM-based vision transformer accelerator. We define a universal design space representation for implementing vision transformers in CIM-based accelerators that support hybrid and heterogeneous features. The corresponding design space comprises the hardware configuration of CIM macros and their spatial mapping scheme. Furthermore, we propose the knowledge-guided grid search (KGGS) algorithm and improved genetic algorithm (IGA) to boost exploration efficiency. The orthogonal experiment and dominance analysis of KGGS could obtain the exploration probabilities of different parameters and ensure its stability, while the unique order crossover and swapping mutation of IGA could retain relative order to avoid legalization processes during the iteration. Performance experimental results show that Harmony achieves 48% area reduction, 13% latency reduction, 32% energy reduction, and $1.27\times$ energy efficiency on average compared with the baseline. The accuracy experiment demonstrates that our hybrid architecture achieves a trade-off between accuracy and performance compared with all-SRAM CIM-based accelerators.**

***Index Terms*—computing-in-memory, mapping optimization, design space exploration, vision transformer**

I. Introduction

Many studies have proved the abilities of Computing-in-memory (CIM) to accelerate deep neural networks (DNNs) and substantively improve performance and energy efficiency [1]–[3]. However, implementing CIM-based AI accelerators specifically for the transformer architecture introduces several challenges, which are outlined below:

For **hardware design**, the main challenge is determining the optimal CIM macro configurations. On the one hand, the solution space combined by macro size, circuit configuration, and other parameters is as enormous as 10^{15} [4], which calls for efficient search algorithms. Previous studies often take hours or even days, so the efficiency of these algorithms limited the exploration of architectural optimization. On the other hand, most existing studies on hardware configuration design space exploration fail to adequately align with emerging accelerator design paradigms characterized by **hybrid SRAM/RRAM** [5]–[11] and **heterogeneous CIM macro sizes** [4], [5]. As evidenced by Fig. 1, the substantial variation in optimal configurations across different transformer layers fundamentally necessitates heterogeneous architectural solutions. Furthermore, our analysis in V demonstrates that strategically replacing SRAM-based CIM macros with RRAM-based CIM macros in some transformer layers enables superior accuracy-performance trade-offs compared to all SRAM designs.

For **spatial mapping (SPM)**, the main challenge stems from the change of data flow. Because of the transformer's larger scale and complex data flow, some CIM-based accelerators use NoC to replace H-tree. Unlike H-tree, a NoC-based interconnect scheme sometimes needs to transform outputs from one CIM tile to another directly, rather than the global buffer. So, the spatial position of CIM tiles is no longer irrelevant because it determines the latency and energy of data flow transmission. Previous works focus on weight matrix transformation [12], partition, and duplication [13] for the convolution layer. They usually apply a layer-sequential mapping strategy [14] or greedy algorithm [15] to place layers with the largest data movement in the center of the accelerator. This simple solution limits the opportunities to fully optimize communication, which becomes increasingly important as the scale of accelerators and the structural complexity of networks increase. The transformer's complex network structure increases the bandwidth pressure of NoC because of the sharing or interleaving of data paths, which blocks the data movement and harms the latency and throughput. So, Harmony aims to optimize the spatial mapping strategy to reduce the impact of other communication flows.

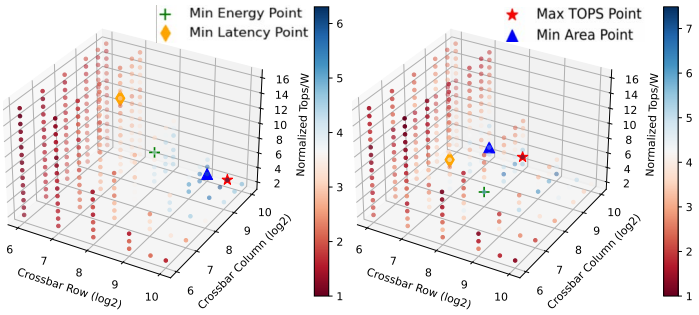


Fig. 1 Energy efficiency results from exploring CIM crossbar row size, CIM crossbar column size, and CIM macro row size for ViT-base's two layers.

To address these problems, we propose *Harmony*, **a hardware-mapping co-exploration framework for hybrid CIM-based vision transformer accelerator**. It supports various CIM device types and configurations, and its mapping space reflects the resource allocation and data dependencies of each layer and tile. Harmony includes a hardware exploration engine with a knowledge-guided grid search (KGGS) and a mapping engine using an improved genetic algorithm (IGA) to explore the design space efficiently. In summary, our contributions are as follows:

- We present a universal hardware template for the CIM-based AI accelerator, which considers the design space of **hybrid** and **heterogeneous** features of CIM macros.
- We propose and implement efficient exploration algorithms, including the knowledge-guided grid search algorithm and the improved genetic algorithm.
- Based on the hardware template and exploration algorithm, we develop Harmony, a hardware and mapping co-exploration framework to generate CIM-based transformer accelerators automatically.
- Compared to the baselines, Harmony's co-optimized hardware and

mapping achieve 48% area reduction, 13% latency reduction, 32% energy reduction, and 1.27× energy efficiency on average.

The rest of the paper is organized as follows: Section II introduces the background and motivation, Section III provides the framework overview and design space, Section IV presents the exploration algorithm, Section V conducts evaluations followed by discussion in Section VI, Section VII summarizes our work and future directions.

II. Background

A. Design Automation of CIM-based Accelerator

Traditional CIM-based accelerators highly rely on designers' experience, and the architecture changes with neural network configuration. Some recent studies have tried to achieve the automatic design tool, usually including a design space exploration engine to determine optimal parameters and a mapper to map weight into the computing array to reduce the cost of human and time resources. As illustrated in Table I, NeuroSIM [14] uses a greedy strategy to maximize the utilization rate of computing resources, while PIM-HLS [16] prunes the design space to accelerate design flow, both at the cost of the potential optimal solution. The algorithms of PIMSYN [17], CoMN [15], and [4] are time-consuming which usually require more than a few hours and days. Besides, the spatial mapping space needs further exploration, as shown above.

TABLE I Comparison of representative research and tools for CIM-based accelerators.

	Hardware Opt.		SPM Opt.	Application
	Algorithm	Speed		
PIM-HLS [16]	Enumeration	-	-	CNN
NeuroSIM [14]	Greedy algorithm	Mins	-	CNN
PIMSYN [17]	Evolution algorithm	Hours	-	CNN
[4]	Beam search	Hours	-	CNN
CoMN [15]	Bayesian opt.	Days	✓	CNN
Ours	**KGGS**	**Mins**	✓	**Transformer**

B. Vision Transformer

The transformer has become a critical deep learning architecture with widespread application across computer vision (CV) domains such as ViT [18], DeiT [19], and Swin transformer [20]. Although various variants exist, Vision Transformer (ViT) models typically rely on attention layers as their core components.

A multi-head self-attention (MHSA) block consists of multiple operators, including linear projections (Linear W_Q, W_K, W_V) and output layer, dot product computation and weighted sum ($Matmul(Q, K^T)$ & $Matmul(A, V)$), as well as scaling and softmax operations [21]. The MHSA mechanism in transformer architectures enables the model to capture relationships between distant pixels in an image, thereby improving its ability to handle long-range dependencies. However, the complexity of the network structure poses significant challenges in designing an efficient accelerator. The transformer architecture's wide application has enhanced its status as a foundational model in deep learning and artificial intelligence, further inspiring the need for a specialized hardware accelerator.

III. Harmony Framework

A. Overview

Our proposed framework, Harmony, is demonstrated in Fig. 2. The framework receives vision transformer structures and basic architecture information users determine as inputs. Harmony instantiates different modules based on hardware templates automatically, and the mapping process determines their counts, accelerator data flow, and access number of buffers. Then, we develop a performance evaluator that includes SPICE simulation, NeuroSIM [14], AutoDCIM [22], and DSENT [23] to estimate the PPA of different parts. The metrics, such as latency, area, and energy efficiency, will guide the exploration loop. Harmony also has a knowledge-guided grid search algorithm to find the optimal macro parameters and an improved genetic algorithm to optimize the spatial mapping scheme. This flow iterates until the desired optimized architecture is achieved.

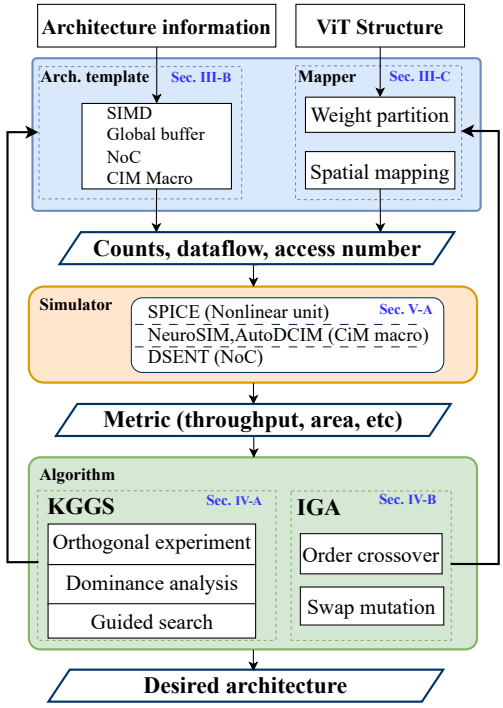


Fig. 2 Overview of Harmony Framework

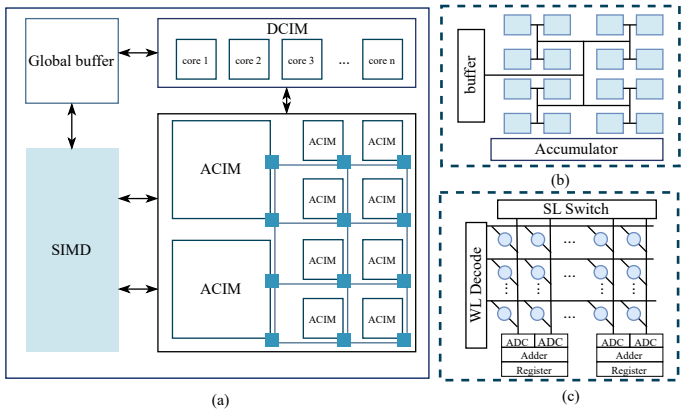


Fig. 3 The overview of architecture, (a) global architecture; (b) CIM tile or CIM macro; (c) CIM crossbar.

B. Hardware Template

Harmony's architecture template is derived by extracting common features from existing **hybrid** SRAM/RRAM CIM-based accelerators and considering implementation [4]–[8], [24], [25]. Like NeuroSIM [14], the digital SRAM-based CIM module has a two-level macro-crossbar structure [22]. RRAM-based CIM tiles are connected in a 2D-mesh NoC, with internal connections via an H-tree structure [14]. The RRAM-based CIM module has two configurations placed on opposite sides for manufacturability and connected to a NoC router. SRAM-based CIM tiles and other modules are placed around the RRAM-based CIM module and directly connected with the bus. SIMD core is the digital circuit module that conducts various complex operations,

including accumulation, Softmax, GELU, and layer normalization.

Digital SRAM-based CIM Tiles: The basic macro architecture is based on AutoDCIM [22], with added input buffers. It has four configurable parameters: (1) *Crossbar row number* (X_r), (2) *Crossbar column number* (X_c), (3) *Macro row number* (M_r), and (4) *Macro column number* (M_c). The configuration is represented as $D = [X_r, X_c, M_r, M_c]$. Different configurations impact the SRAM word line length, peripheral circuits, and adder hierarchy, influencing area, latency, energy efficiency, and PPA.

Analog RRAM-based CIM Tiles: This CIM tile, based on NeuroSim, includes a CIM crossbar, ADCs, peripheral circuits, and input buffers. Seven parameters define its configuration: (1) *Crossbar row number* (X_r), (2) *Crossbar column number* (X_c), (3) *Macro row number* (M_r), (4) *Macro column number* (M_c), (5) *Tile row number* (T_r), (6) *Tile column number* (T_c), and (7) *ADC shared column number* (C_m). These affect the number of resources (e.g., ADCs, word line length, and peripheral circuits) and overall performance. The configuration is represented as $A_n = [X_r, X_c, M_r, M_c, T_r, T_c, C_m]$.

SIMD core: Leveraging a suite of fundamental functional units built within NeuroSim, including shifters, registers, multiplexers, and adders, the critical performance data (such as latency, power consumption, and area) for these units is accurately modeled based on low-level circuit simulations. Building upon this foundation, we further abstract and combine these elemental modules to construct more complex computational functions, such as multipliers, dividers, and maximum-value detectors, to support various computational needs. Finally, based on instructions, the SIMD core can dynamically invoke the corresponding computational module to execute parallel operations.

C. Hardware Feature

Hybrid: Based on the device characteristics of SRAM and RRAM, we achieve a hybrid CIM architecture. RRAM is typically used as the analog Compute-in-Memory (ACIM) device for static weight matrix operations due to its low power consumption, high device density, and relatively low reliability. SRAM is widely used as the digital CIM (DCIM) device due to its high write speed, high endurance, and high reconfigurability, which allow it to perform dynamic weight matrix multiplications with frequent writing. As shown in Fig. 4, we map layers of the neural network to different computing modules based on device characterizations. The large weight matrix is divided into parts according to the CIM crossbar sizes. The CIM device simulates MAC operations to compute the input row vectors in parallel. Partial results are accumulated from the local adder tree and transferred to the next module.

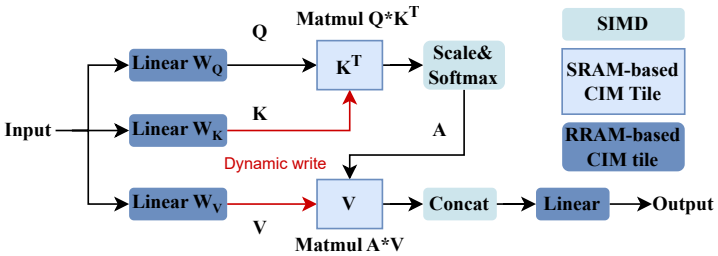


Fig. 4 Dataflow in multi-head attention block

Heterogeneous: We provide two different RRAM-based CIM macro configurations, considering the difference between the best parameters of different transformer layers is huge, as proved in Fig. 1. The ViT architecture includes four static weight matrix multiplication sizes: the projection layer, MLP layer 1, MLP layer 2, and the output layer. So we could use $L = [L_1, L_2, L_3, L_4]$, where $L_i = 1, 2$ to represent two different RRAM-based CIM macro configurations.

TABLE II Design space of CIM macro configurations

Component	Descriptions	Candidates
X_r	the row number of crossbar	32,64,128, 256,512,768
X_c	the column number of crossbar	32,64,128, 256,512,768
M_r	the number of crossbars per row in the CIM macro	2, 3, 4, 5
M_c	the number of crossbars per column in the CIM macro	2, 3, 4, 5
T_r	the number of macros per row in the CIM tile	2, 3, 4, 5
T_c	the number of macros per column in the CIM tile	2, 3, 4, 5
C_m	muxed column	1, 2, 4, 8
L_i	the RRAM-based CIM macro type of layer i	1, 2

So, a hybrid heterogeneous architecture specification is $V = [A_1, A_2, D, L]$. When $A_1 = A_2$ or $L_1 = L_2 = L_3 = L_4$, it represents the homogeneous architecture.

D. Mapping

Mapping refers to the process of associating computational operators with hardware modules, partitioning weights according to the constrained tile and crossbar capacities, and determining the mapping of each layer to specific tiles, which significantly affects the data transmission paths and movement latency in the network-on-chip (NoC) mesh [26]. As illustrated in Fig. 4, compared to convolutional layers, transformers exhibit a more complex network structure, which indicates that the spatial position of weights plays a crucial role in inter-layer delay. This section introduces and analyzes the spatial mapping scheme specifically designed for transformers.

We will use the homogeneous architecture to illustrate our spatial mapping scheme because homogeneous is a particular case of heterogeneous, which can be considered separately in two parts. The **spatial mapping scheme** for each layer has two attributes: Resource ($R_i = (L_i, IF_i, WGT_i, OF_i)$) and Data path ($DP_i = (L_i, IS_i, OD_i)$). *Resource* represents the feature map of the input (IF_i), weight (WGT_i), and output (OF_i) matrix for $layer_i$, which decides the needed computing resources and data movements. *Data path* contains the data sources of input (IS_i) and the destination of output (OD_i) for $layer_i$.

Besides, we use $P = [(L_1, N_1), (L_2, N_2), ..., (L_i, N_i)]$ to refer to the mapping position of layers, where L_i means the index of $layer_i$. N_i is the tile number of $layer_i$ and could be calculated by

$$N_i = [\frac{L_i^r}{T_r}]\lceil\frac{L_i^c}{T_c} \times C_n] \tag{1}$$

where L_i^r and L_i^c are row and column of $layer_i$ and C_n means cells per number. Furthermore, Harmony uses the zigzag encoding method to convert P into a mapping result.

As shown in Fig. 5(a), we propose a zigzag encoding strategy to transform the mapping sequence $P = [(1, 2), (3, 2), (2, 3), (4, 2)]$ to a realistic result. This strategy is designed to optimize data movement and prevent inefficient mapping schemes. For instance, $layer_1$, is mapped to tiles (1-1) and (1-2). Subsequently, $layer_3$, which requires two tiles, occupies the adjacent tiles (3-1) and (3-2). By placing tiles belonging to the same layer physically close, such as (3-2) being next to (3-1) rather than the leftmost part of the second row, we significantly reduce intra-layer data movement paths. This effective utilization of the zigzag encoding strategy ensures a more logical and efficient mapping.

Once mapping is complete, the data flow for each layer is also determined. As illustrated in Fig. 5(b), solid lines refer to input data paths, while dashed lines represent output data paths for each layer. The primary challenge arises from overlapping these distinct data paths, which can lead to significant communication conflicts and bandwidth pressure. Critically, the transmission latency is directly influenced by the

data path length and the resources required for transmission. Therefore,

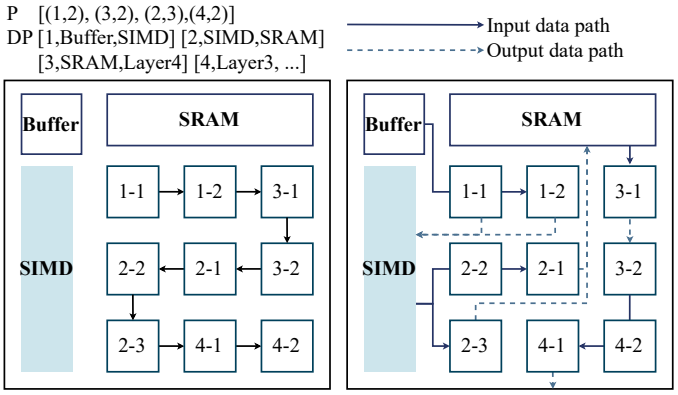


Fig. 5 Parsing encoded sequence into SPM scheme, (a) zigzag encoding method; (b) data path of each layer.

IV. Algorithm

A. Knowledge-Guided Grid Search

Based on TABLE II, the design space will be approximately 5.6×10^{12}, making it impractical to explore all possible combinations. During the traditional accelerator design flow, engineers often test several representative parameter configurations first and then gradually modify the configurations based on the importance of the parameters. Inspired by the designers' behavior, we propose a knowledge-guided grid search algorithm with three steps: orthogonal experiment, dominance analysis-based knowledge extraction and knowledge-guided search.

The *orthogonal experiment* is a widely adopted experimental statistical design method that does not test all parameter combinations. Instead, it uses an orthogonal array (OA) to conduct a minimum amount of representative experiments, significantly reducing experimentation's time and resource costs [27]. Orthogonal array follows the balance principle with the following two properties: (1) each parameter's value occurs the same number of times in each column, and (2) each possible level combination of any two given parameters occurs the exact times in the array. The algorithm could automatically select OA depending on the specific design space [28].

After conducting the orthogonal experiment, we apply *dominance analysis* to evaluate the importance (D) of various parameters, which can also be viewed as knowledge extraction. Line 3 of Algorithm 1 means getting all subsets of the design space V, where $\mathcal{P}(V)$ refers to the power set of V. $R^2(S)$ refers to calculating the R^2 value for parameter combination S as shown in Eq.2 and Eq.3, where *SSR* is the residual sum of squares, *SST* is the total sum of squares and $\hat{y}(S)$ is the prediction value of linear regression for parameter combination S.

$$R^2(S) = 1 - \frac{\text{SSR}}{\text{SST}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i(S))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2}$$

$$\hat{y}(S) = \beta_0 + \sum_{j \in S} \beta_j x_j \tag{3}$$

Then, the algorithm calculates each variable's contribution by comparing the R^2 values with and without v for all subsets $S \in \mathcal{P}(V)$. Finally, the dominance scores D are derived from the average contribution across all subsets, as shown in line 7 of the algorithm. In conclusion, dominance analysis identifies the relative contribution of each variable in explaining the variance of the dependent variable by comparing their impact across different model combinations [29].

Algorithm 1 Knowledge-Guided Grid Search

Input: Design Space: V; Orthogonal arrays: OAs; Iteration number: N

Output: Optimal Configuration: P_N^*

1: Initialize P_0 by OAs, V ▷ Step 1: orthogonal experiment
2: $P_0^* \leftarrow MAX(P_0)$
3: $\mathcal{P}(V) \leftarrow \{S \mid S \subseteq V\}$ ▷ Step 2: dominance analysis
4: **for** $S \in \mathcal{P}(V)$ **do**
5: **for** $v \in V \setminus S$ **do**
6: $\Delta R^2 \leftarrow R^2(S \cup \{v\}) - R^2(S)$ ▷ eq.(2) and eq.(3)
7: $D[v] \leftarrow D[v] + \Delta R^2$
8: **end for**
9: **end for**
10: $K_0 \leftarrow \frac{D}{|\mathcal{P}(V)|}$
11: **for** $i \leftarrow 1$ to N **do** ▷ Step 3: guided search
12: $P_i \leftarrow Mutation(P_{i-1}^*, K_{i-1})$
13: $P_i^* \leftarrow MAX(P_i, P_{i-1}^*)$
14: $K_i \leftarrow (P_i^* - P_{i-1}^*)/P_{i-1}^* \times K_{i-1}$
15: **end for**

The knowledge obtained from dominance analysis helps determine the exploration probability of different variables for the next *grid search*. Specifically, each variable is treated as a separate search dimension, forming a hypercube where each parameter combination corresponds to a point on the grid. Starting from the initial optimal candidate P_0^*, which is derived from our previous orthogonal experiment, we modify its variables to generate new populations based on the knowledge vector K. We modify variables of P^* repeatedly until new populations are generated during the $Mutation()$ process, and the modified probability is based on the dominance values of variables. After each iteration, we update the probability K_i and optimal candidate P_i^* dynamically, as shown in lines 13-14 of Algorithm 1.

B. Improved Genetic Algorithm

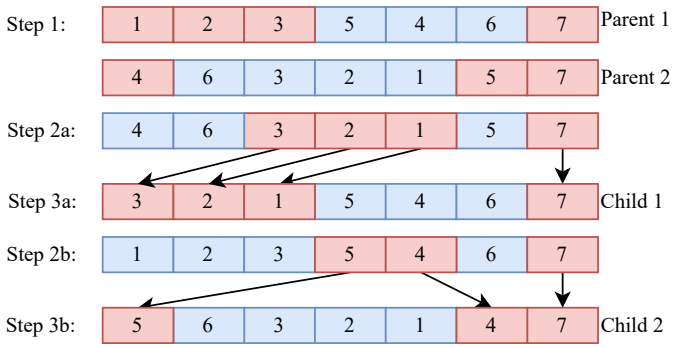


Fig. 6 Order crossover

Unlike the vanilla genetic algorithm [30], we define special operators (order crossover and swap mutation) to avoid legalization processes during the iteration, which often significantly reduce the algorithm's efficiency, ensuring that each offspring complies with the rules.

Order Crossover. Order crossover (OX) is a special operator that ensures the sequence elements' number and relative order do not change. It selects a segment of genes from two parents, keeping the gene order unchanged while filling other genes into the children's genes in the same relative order. This method is particularly suitable for solving optimization problems that require maintaining a particular order and number of elements, and it demonstrates excellent performance in maintaining critical information [30]. As shown in the step 1 of Fig. 6, blue gene segments $S_1 = |546|$ and $S_2 = |6321|$ are selected from two parents $P_1 = (1235467)$ and $P_2 = (4632157)$. It should be noted that

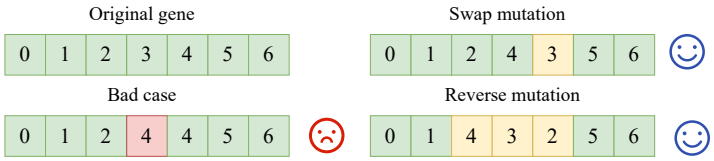


Fig. 7 The diagram of swap mutation.

we ignore N_i because it is decided by Eq.1 when we get the optimal hardware configuration. The gene is first extracted and placed in the same position as $(\times\times\times|546|\times)$ and $(\times|6321|\times\times)$. Then, the gene segment is deleted from the other parents' genes $O_1 = (\times\times 321\times 7)$ and $O_2 = (\times\times\times 54\times 7)$. Finally, we fill the remaining gene segment in the genes in order and get two child genes, $C_1 = (3215467)$ and $C_2 = (5632147)$. From two parents, OX would obtain two children, which are noted as branch a and branch b in Fig. 6. We implement order crossover and multiple variants by varying the gene segments' position, length, and order.

Mutation. Harmony implements two mutation strategies: swap mutation and reverse swap mutation. Swap mutation selects a sub-segment in the parent and exchanges it with a sub-segment in another position. Reverse swap mutation selects a continuous segment in a sequence and reverses the order of its internal elements, which helps test the impact of local structural changes on the solution. As illustrated in Fig. 7, our employed mutation method is designed to avoid unfavorable bad cases and enhance efficiency. This strategy introduces significant genetic variation while maintaining the local sequence structure, enabling the algorithm to discover novel potential solutions.

V. Evaluation

A. Setup

We use the RRAM-based and SRAM-based CIM macros correspondingly from NeuroSim [31] and AutoDCIM [22]. Both technology nodes of the CIM macro are evaluated at 22nm, and data have been validated through post-silicon. Harmony implements Network-on-Chip mesh interconnections using an in-house performance model with DSENT [23] integrated to evaluate performance, power, and area (PPA). Additional architectural parameters and bandwidth references are sourced from ISAAC [32].

We quantize the Vision Transformer (ViT) model to INT8 format using the I-ViT method, replacing its nonlinear functions with integer-only approximations [33]. We also implement an integer-only SIMD module to support I-ViT's Shiftmax, ShiftGELU, and I-layerNorm operations, which are transferred from the Softmax, GELU, and layerNorm functions of the ViT architecture. TABLE III details our benchmark models: DeiT-Tiny, DeiT-Small, and ViT-Base, each having distinct parameter configurations [18], [34]. All evaluations are conducted on the ImageNet 2012 dataset [35].

TABLE III Vision transformer specifications

	DeiT-Tiny	**DeiT-Small**	**ViT-Base**
#Parameters	5M	22M	86M
Embedding dimension	192	384	768
#Heads	3	6	12
Image size	$(224, 224, 3)$		
#Tokens	198		
Head dimension	64		
MLP ratio	4		

Our baseline consists of two components: the architecture and the search algorithm. We use NeuroSIM's architecture with SRAM (**S-Arch**) as baseline architecture, and the architecture explored by our Harmony framework is **H-Arch**. We adopt the evolution algorithms (EA) of PIMSYN [17] as the baseline search algorithm for algorithm evaluation [1] because NeuroSIM's greedy strategy of maximizing utilization rate performs poorly on such complex architectures. We compare three configurations, S-Arch+EA, H-Arch+EA, and H-Arch+KGGS, optimizing for area, latency, and energy on benchmarks. The starting points of EA and KGGS come from orthogonal experiments, so they are the **same** for each experiment. Unless otherwise specified, the results represent the average of ten tests.

B. Architecture Search Result

Fig. 8 compares **architectures** and **search algorithms** for different optimization objectives.

S-Arch vs. H-Arch: Architecture Advantages The results demonstrate that **H-Arch+EA** achieves about 40% area reduction and 8% latency reduction on average. However, its energy efficiency improvement is negligible. In contrast, our **H-Arch+KGGS** significantly outperforms the baseline, yielding a 56% area reduction, 13% latency reduction, and 32% energy reduction. H-Arch offers a distinct advantage over S-Arch by simultaneously leveraging the strengths of different types of CIM macros and increasing the flexibility of CIM macro selection. This allows H-Arch to assign more suitable macro configurations to each layer, leading to more suitable solutions. Therefore, H-Arch, including EA and KGGS, often finds sounder solutions than S-Arch.

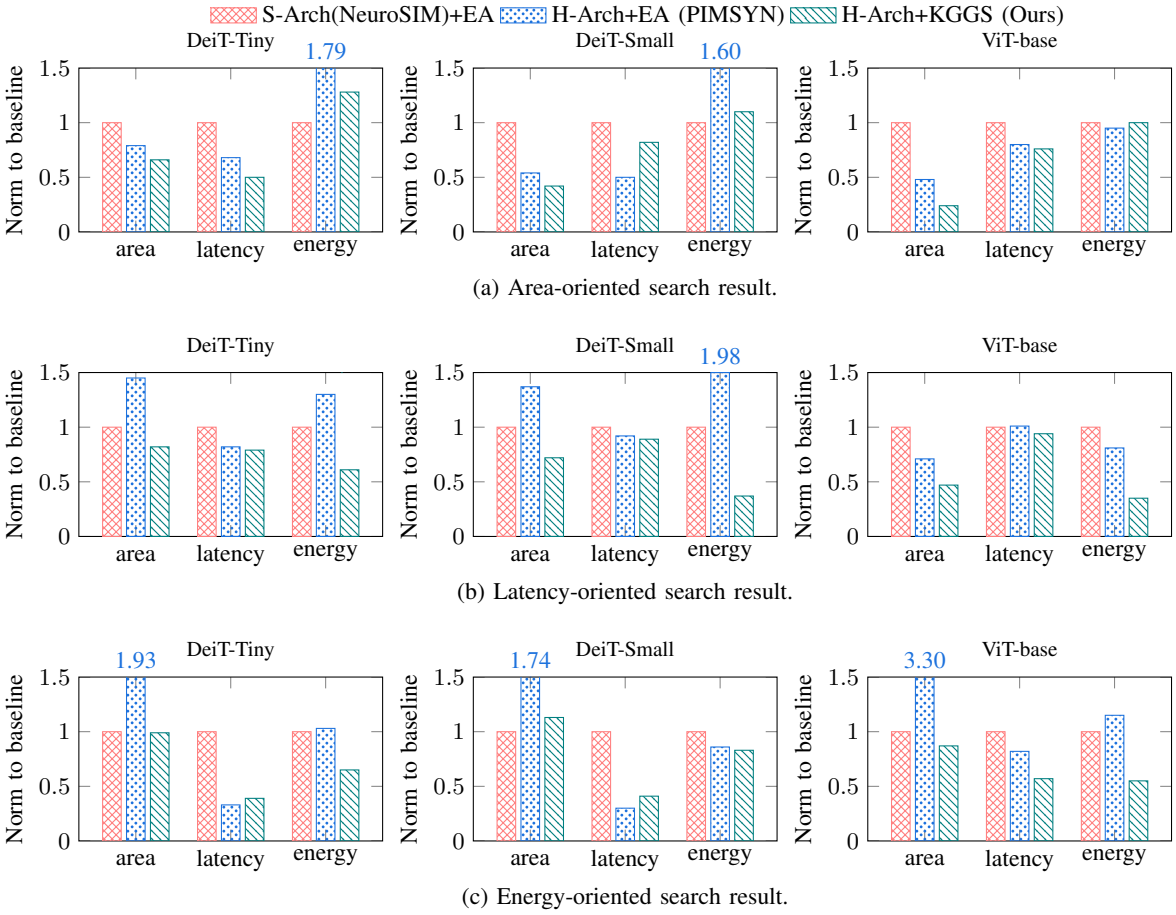
EA vs. KGGS: Algorithm Efficiency It is worth noting that the performance improvement of H-Arch+EA sometimes appears inconsistent with the characteristics of CIM types: ACIM typically boasts higher cell density and lower power, while DCIM provides the benefit of low latency [7]. For example, the latency-oriented search result for ViT-base of **H-Arch+EA** is larger than the baseline, and its energy efficiency improvement is very little. Furthermore, it often comes at the cost of other performance, as the blue numbers indicate. This discrepancy stems from the larger design space of H-Arch compared to S-Arch, which sometimes makes it challenging for EA to find a better configuration. In contrast, **H-Arch+KGGS** consistently achieves greater improvements than **H-Arch+EA**. We attribute this to the higher efficiency of the KGGS algorithm. To further substantiate this, we will detail a comprehensive analysis of the search efficiency of both algorithms in the subsequent section.

We record the best results of algorithms during the iteration and draw convergence curves in Fig. 9(a). Compared with random grid search (RS) and EA, KGGS demonstrates significantly higher sample efficiency because of the specific knowledge. As shown in Fig. 9(b), we use a box plot to illustrate the statistical distribution (min, max, median, and quarter medians) of results obtained over 10 trials. For comparison, we also mark data for two state-of-the-art hybrid CIM-based vision transformer accelerators: H3DAtten [5] (utilizing 16 nm SRAM DCIM and 40 nm RRAM ACIM) and AESHA [36] (whose energy efficiency is scaled to 22 nm). The results demonstrate two key advantages of KGGS: its superior stability, evidenced by the narrower range of its results compared to others, and its significantly better performance across all metrics (lowest, highest, and average values) when compared to both the accelerators and other algorithms.

C. Mapping Optimization Result

Next, we verify the efficiency of IGA compared with other mapping algorithms such as NeuroSim's layer-sequential (LS) mapping strategy [14], random search (RS), and CoMN's greedy algorithm (Greedy) [15]. We select the optimal latency-oriented results of three networks and calculate the latency improvement and geometric average. Although both the RS and Greedy algorithms have achieved specific improvements compared to the baseline, the magnitude is limited

[1] Since PIMSYN does not specify the used evolution algorithm (EA), we selected the classic genetic algorithm.

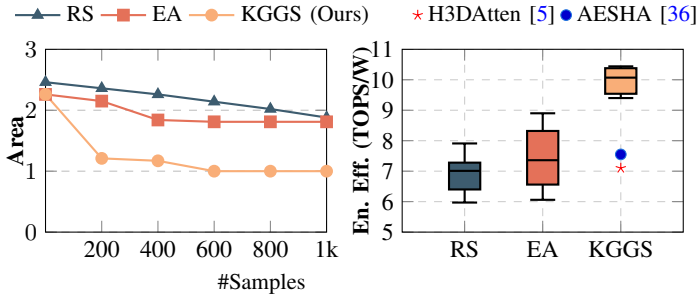


(a) Area-oriented search result.

(b) Latency-oriented search result.

(c) Energy-oriented search result.

Fig. 8 Energy, area, and latency of the optimal architectures in different objectives for three ViT models (normalized to the baseline).



(a) Convergence curve (b) Distribution Box-plot

Fig. 9 The results of KGGS compared with other baselines in the hardware design space explorations.

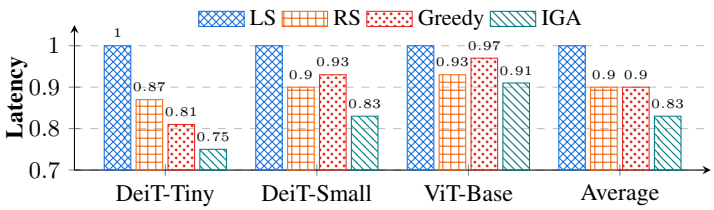


Fig. 10 Comparison of latency

and rapidly decreases as the search space increases. As shown in Fig. 10, IGA achieves over 9% improvement compared to the baseline on ViT-base, while it was 25% higher on DeiT-Tiny. The significant improvement on average 17% proves the IGA's effectiveness compared to traditional mapping schemes.

D. Accuracy Result

TABLE IV Accuracy (%) of different architectures.

Architecture	DCIM	ACIM[1]	ACIM[2]	Hybrid CIM
DeiT-Tiny	72.2	69.14	71.25	71.11
DeiT-Small	79.8	76.65	78.94	79.01
ViT-Base	81.8	75.91	80.05	79.71

ACIM[1]: The SRAM-based ACIM tile with 7-bit ADC
ACIM[2]: The SRAM-based ACIM tile with 8-bit ADC

As discussed in [37]–[39], RRAM suffers from significantly higher latency (1–2 orders of magnitude) compared to SRAM at the same technology node and has limited endurance (approximately 10^6–10^9 conservative writes), which severely impacts their lifetime and constrains their applicability to Transformer models. Given these limitations, using RRAM for all layers in a Transformer is impractical. Therefore, we compare an all-SRAM architecture with our proposed hybrid architecture (which incurs **no accuracy loss**). As shown in TABLE IV, the hybrid architecture achieves an accuracy of 79.71%, which is close to the all-SRAM architecture (original accuracy). Similar to the results in MISCim [9], we find that achieving a comparable accuracy with analog SRAM-based CIM requires setting the ADC to 8/9 bits, resulting in even higher latency and power consumption than digital SRAM-based CIM. In summary, the hybrid architecture offers

significant performance improvements (Section V-B) while maintaining acceptable accuracy loss.

VI. Discussion

Hybrid or heterogeneous CIM-based Accelerator The hybrid Computing-in-Memory (CIM) architecture is widely adopted because it can simultaneously utilize the complementary advantages of different CIM types. For example, some approaches combine SRAM crossbars with RRAM to compensate for variations and enhance accuracy [7]. Others, like HARDSEA, strategically employ both high-efficiency analog RRAM-CIM and high-precision digital SRAM-CIM [8]. The exploration of 3D RRAM and SRAM heterogeneous stacking further demonstrates the versatility of these architectures [5]. Building on these advancements, this work introduces an efficient design space exploration algorithm for hybrid CIM systems.

Adapt Harmony to Other Design Space. Thanks to pre-designed orthogonal tables of statisticians, our proposed algorithm achieves strong scalability, primarily because it can autonomously determine suitable orthogonal arrays. This means our algorithm doesn't need modification even when the design space evolves, such as with new vision transformer variants or more feature levels. This self-adaptive approach is essential for addressing increasingly complex problems effectively.

VII. Conclusion

This article proposes Harmony, a hardware and mapping co-exploration framework, to generate CIM-based transformer accelerators automatically. Harmony has a knowledge-guided grid search (KGGS) algorithm and an improved genetic algorithm (IGA) to improve exploration efficiency. The experimental results demonstrate the algorithm's efficiency and the design space's enormous potential. In the future, we will provide a plug-in system-level CIM simulator.

Acknowledgment

This work was in part supported by the Guangdong Basic and Applied Basic Research Foundation (No. 2023A1515110353), the Guangdong Science and Technology Department (No. 2025A0505000023 and No. 2025B1212150003), and a Guangdong Provincial Project (No. 2023QN10X252).

References

[1] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, 2016.

[2] Z. Chen, X. Chen, and J. Gu, "15.3 a 65nm 3t dynamic analog ram-based computing-in-memory macro and cnn accelerator with retention enhancement, adaptive analog sparsity and 44tops/w system energy efficiency," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021.

[3] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M.-F. Chang, X. Li, H. Yang *et al.*, "14.3 a 65nm computing-in-memory-based cnn processor with 2.9-to-35.8 tops/w system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, 2020.

[4] J. Park, Y. Shin, and H. Sung, "Multi-objective architecture search and optimization for heterogeneous neuromorphic architecture," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023.

[5] W. Li, M. Manley, J. Read, A. Kaul, M. S. Bakir, and S. Yu, "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.

[6] Y. Ding, C. Liu, M. Duan, W. Chang, K. Li, and K. Li, "Haima: A hybrid sram and dram accelerator-in-memory architecture for transformer," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023.

[7] G. Krishnan, Z. Wang, I. Yeo, L. Yang, J. Meng, M. Liehr, R. V. Joshi, N. C. Cady, D. Fan, J.-S. Seo *et al.*, "Hybrid rram/sram in-memory computing for robust dnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[8] S. Liu, C. Mu, H. Jiang, Y. Wang, J. Zhang, F. Lin, K. Zhou, Q. Liu, and C. Chen, "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.

[9] C. Wang, Z. Chen, and S. Huang, "Micsim: A modular simulator for mixed-signal compute-in-memory based ai accelerator," in *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, 2025.

[10] Z. Fu, Y. Zuo, Y. Ma, and J. Huang, "Optimizing Heterogeneous Compute-in-Memory with Hybrid Dataflow and In-Network Reduction for Vision Transformer," in *2025 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2025.

[11] C. Wang, Z. Fu, J. Huang, and S. Huang, "Hemlet: A heterogeneous compute-in-memory chiplet architecture for vision transformers with group-level parallelism," *arXiv preprint arXiv:2511.15397*, 2025.

[12] X. Peng, R. Liu, and S. Yu, "Optimizing weight mapping and data flow for convolutional neural networks on processing-in-memory architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2019.

[13] R. Pelke, J. Cubero-Cascante, N. Bosbach, F. Staudigl, R. Leupers, and J. M. Joseph, "Clsa-cim: A cross-layer scheduling approach for computing-in-memory architectures," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.

[14] P.-Y. Chen, X. Peng, and S. Yu, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[15] L. Han, R. Pan, Z. Zhou, H. Lu, Y. Chen, H. Yang, P. Huang, G. Sun, X. Liu, and J. Kang, "Comn: Algorithm-hardware co-design platform for non-volatile memory based convolutional neural network accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[16] Y. Zhu, Z. Zhu, G. Dai, F. Tu, H. Sun, K.-T. Cheng, H. Yang, and Y. Wang, "Pim-hls: An automatic hardware generation tool for heterogeneous processing-in-memory-based neural network accelerators," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023.

[17] W. Li, X. Sun, X. Wang, L. Wang, Y. Han, and X. Chen, "Pimsyn: Synthesizing processing-in-memory cnn accelerators," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, 2020.

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, 2017.

[22] J. Chen, F. Tu, K. Shao, F. Tian, X. Huo, C.-Y. Tsui, and K.-T. Cheng, "Autodcim: An automated digital cim compiler," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023.

[23] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, 2012.

[24] Z. Zhu, H. Sun, T. Xie, Y. Zhu, G. Dai, L. Xia, D. Niu, X. Chen, X. S. Hu, Y. Cao *et al.*, "Mnsim 2.0: A behavior-level modeling tool for processing-in-memory architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[25] G. Krishnan, S. K. Mandal, C. Chakrabarti, J.-S. Seo, U. Y. Ogras, and Y. Cao, "Impact of on-chip interconnect on in-memory acceleration of deep neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2021.

[26] J. Cai, Z. Wu, S. Peng, Y. Wei, Z. Tan, G. Shi, M. Gao, and K. Ma, "Gemini: Mapping and architecture co-exploration for large-scale dnn chiplet accelerators," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024.

[27] D. Li, S. Yao, Y.-H. Liu, S. Wang, and X.-H. Sun, "Efficient design space exploration via statistical sampling and adaboost learning," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016.
[28] K.-T. Fang and Y. Wang, *Number-theoretic methods in statistics*. CRC Press, 1993, vol. 51.
[29] D. V. Budescu, "Dominance analysis: a new approach to the problem of relative importance of predictors in multiple regression." *Psychological bulletin*, 1993.
[30] K. Deep and H. Mebrahtu, "New variations of order crossover for travelling salesman problem," *International Journal of Combinatorial Optimization Problems and Informatics*, 2011.
[31] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "Dnn+ neurosim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *2019 IEEE international electron devices meeting (IEDM)*, 2019.
[32] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, 2016.
[33] Z. Li and Q. Gu, "I-vit: integer-only quantization for efficient vision transformer inference," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.
[34] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*, 2021.
[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, 2015.
[36] X. Yu, T. Ni, X. Sheng, Y. Pan, L. He, and L. Zhao, "Aesha: Accelerating eigen-decomposition-based sparse transformer with hybrid rram-sram architecture," ser. ICCAD '24. New York, NY, USA: Association for Computing Machinery, 2025. [Online]. Available: https://doi.org/10.1145/3676536.3676660
[37] I. Chakraborty, M. Ali, A. Ankit, S. Jain, S. Roy, S. Sridharan, A. Agrawal, A. Raghunathan, and K. Roy, "Resistive crossbars as approximate hardware building blocks for machine learning: Opportunities and challenges," *Proceedings of the IEEE*, 2020.
[38] F. Zahoor, T. Z. Azni Zulkifli, and F. A. Khanday, "Resistive random access memory (rram): an overview of materials, switching mechanism, performance, multilevel cell (mlc) storage, modeling, and applications," *Nanoscale research letters*, 2020.
[39] A. Ankit, I. El Hajj, S. R. Chalamalasetti, S. Agarwal, M. Marinella, M. Foltin, J. P. Strachan, D. Milojicic, W.-M. Hwu, and K. Roy, "Panther: A programmable architecture for neural network training harnessing energy-efficient reram," *IEEE Transactions on Computers*, 2020.