scientific reports

OPEN

Efficient memristor accelerator for transformer self-attention functionality

Meriem Bettayeb[1,2], Yasmin Halawani[3], Muhammad Umair Khan[1], Hani Saleh[1] & Baker Mohammad[1]✉

The adoption of transformer networks has experienced a notable surge in various AI applications. However, the increased computational complexity, stemming primarily from the self-attention mechanism, parallels the manner in which convolution operations constrain the capabilities and speed of convolutional neural networks (CNNs). The self-attention algorithm, specifically the matrix-matrix multiplication (MatMul) operations, demands a substantial amount of memory and computational complexity, thereby restricting the overall performance of the transformer. This paper introduces an efficient hardware accelerator for the transformer network, leveraging memristor-based in-memory computing. The design targets the memory bottleneck associated with MatMul operations in the self-attention process, utilizing approximate analog computation and the highly parallel computations facilitated by the memristor crossbar architecture. Remarkably, this approach resulted in a reduction of approximately 10 times in the number of multiply-accumulate (MAC) operations in transformer networks, while maintaining 95.47% accuracy for the MNIST dataset, as validated by a comprehensive circuit simulator employing NeuroSim 3.0. Simulation outcomes indicate an area utilization of 6895.7 μm^2, a latency of 15.52 seconds, an energy consumption of 3 *mJ*, and a leakage power of 59.55 μW. The methodology outlined in this paper represents a substantial stride towards a hardware-friendly transformer architecture for edge devices, poised to achieve real-time performance.

Transformer networks combine the advantages of convolutional neural networks (CNN) along with attention models. The attention mechanism is well recognised and very effective owing to its capacity to process input sequences of diverse durations and prioritize the most relevant components of the input[1]. The structure of the transformer model is comprised of six identical encoder blocks and six identical decoder blocks, each featuring similar module configurations, as described in Vaswani et al.[2]. Within each encoder block, two key modules are present: multi-head self-attention and feed-forward. Following the execution of each module, a further step involves the combination of the input and output via an add and normalization block. This process computes the layer normalization, as seen in Fig. 1. The input source for the multi-headed attention layer is obtained from either the preceding encoder block or the input embedding. Furthermore, as seen in Figure 1, each decoder block consists of three fundamental functional layers: two multi-headed self-attention layers and a feed-forward layer. It is worth mentioning that, in comparison to the encoder block, the decoder block integrates an extra encoder-decoder self-attention module, where one of its inputs is linked to the output of the encoder stack.

Large-scale transformer networks have significant power and computational requirements, preventing them from being deployed on edge devices or in resource-constrained environments such as Internet of Things (IoT) systems[3–5]. Recent attempts to compress and speed natural language processing (NLP) paradigms on embedded systems like field programmable gateway arrays (FPGAs) have been reported[6]. For instance, Li et al.[6] compressed NLP models using an improved representation based on a block-circulant matrix. Additionally, they suggested a novel design for the FPGA architecture that would enable resource management to be more efficient while maintaining high throughput and low latency. As a consequence, they may achieve 27 times, 3 times, and 81 times the performance (as measured by frames per second), decreased power consumption, and increased energy efficiency, when compared to the central processing unit (CPU) for the RoBERTa model[7]. To accomplish this,[8] they proposed constructing hardware-aware transformers (HAT) via the use of neural architecture search techniques such as[9–11]. To be more precise, a SuperTransformer is trained to estimate its performance without completely training it. This model encompasses the largest model feasible in the search space while sharing weights across common parts. Finally, an evolutionary search is conducted while taking into account hardware

[1]System-on-Chip Lab, Computer and Information Engineering, Khalifa University, Abu Dhabi, UAE. [2]Computer Science and Information Technology Department, College of Engineering, Abu Dhabi University, Abu Dhabi, UAE. [3]College of Engineering and IT, University of Dubai, Dubai, UAE. ✉email: baker.mohammad@ku.ac.ae

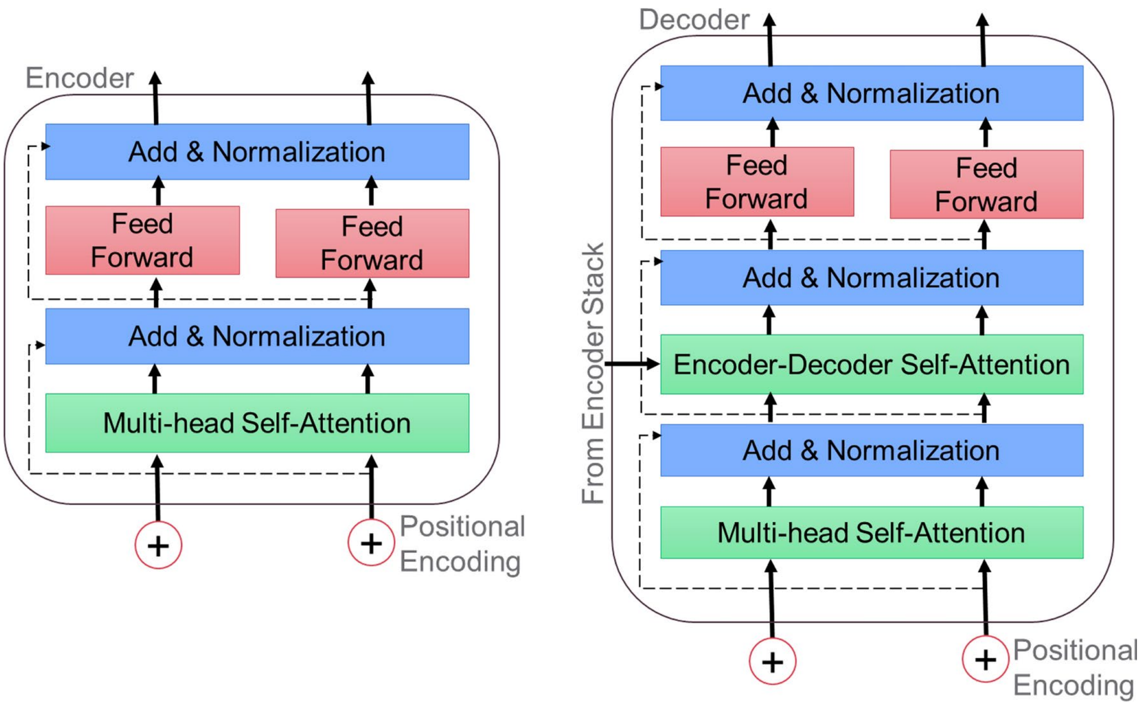

Figure 1. Encoder and decoder structure of the transformer. Each encoder has two layers of multi-head self-attention and a position-wise fully connected feed-forward network. Each decoder uses multi-head self-attention, multi-head attention, and a position-wise feed-forward neural network.

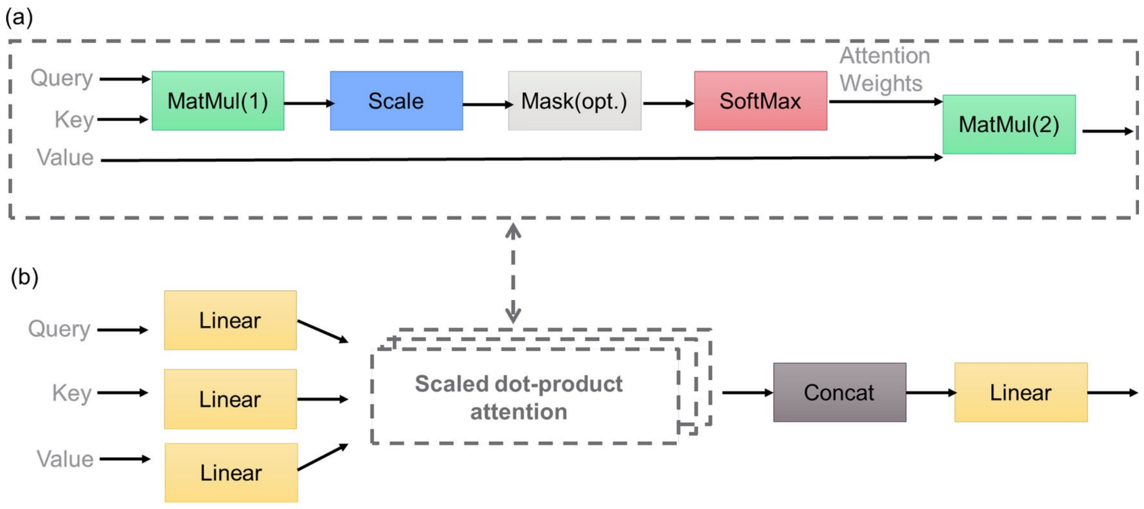

Figure 2. (**a**) Scaled dot-product attention and (**b**) multi-head attention represent essential components of the model. The use of multi-head attention involves the simultaneous use of a collection of *n* attention layers, referred to as heads. This approach allows for the extraction of separate representation subspaces, allowing the model to highlight different locations efficiently.

latency restrictions in order to identify a subTransformer model that is adequate for the target hardware platform (e.g., IoT device, graphics processing unit (GPU), or CPU). Laguna et al.[12] suggested accelerating transformer networks by combining crossbar arrays (XBars) with content addressable memories (CAMs). Furthermore, Lu et al.[13] focused on building two main components of the transformer on FPGA, which are: the multi-head attention (MHA) ResBlock and the position-wise feed-forward network (FFN) ResBlock. Additionally, Peng et al.[14] used FPGA to create a sparse attention operator and a length-aware hardware resource scheduling system.

Qu et al.[15] proposed an algorithm optimization technique for lowering the self-attention mechanism's quadratic cost. The proposed method effectively finds and eliminates weak links within attention graphs, hence avoiding the need for corresponding computations and memory accesses. The suggested method of attention detection was developed utilizing efficient hardware specialization methods in conjunction with the TSMC 22nm standard cell library and static random access memory (SRAM) module. Tu et al.[16] developed a 28nm transformer computing in-memory (TranCIM) device based on the reconfigurable digital in-Memory computing (IMC) design concept. Three elements address the memory and compute issues posed by transformer models' attention mechanisms. TranCIM links its in-memory computing engines through a reconfigurable streaming network (RSN) with specialized modes for various transformer layers: pipeline mode for attention layers and parallel mode for fully connected layers. The SRAM-CIM macro in TranCIM is developed with a bitline-transpose structure to align the input feeding and weight writing directions. Thus, transposing K is accomplished in the QKT-pipeline mode without the need for extra storage or buffer access. TranCIM incorporates a sparse attention scheduler (SAS) that dynamically configures the in-memory computing workload to accommodate various sparse attention patterns. Song et al.[17] presented a two-level pruning to accommodate transformers on mobile devices, but their study ignores the underlying implementation. Qi et al.[18], on the other hand, presented an efficient acceleration system that combines balanced model compression with FPGA implementation optimization. Transformers with block-balanced sparsity were subjected to block-based weight pruning. Peng et al.[19] optimized transformers with column-balanced block-wise sparsity using block-based weight pruning. The number of sparse rows or columns in each weight block is kept the same, or the number of sparse blocks along each column is kept the same in these approaches. This allows for maximum exploitation of compute resources and high throughput in hardware implementations. Wang et al.[20] introduced a transformer processor with three critical characteristics to address these issues: (1) A big-exact-small-approximate (BESA) processing element (PE) saves 1.62 MAC power for WR-Tokens by using a self-gating approximate multiplier. (2) A bidirectional asymptotic speculation unit (BASU) saves 46.7 percent of superfluous computations by capturing sparsity. BASU makes use of attention's local features to quickly locate the variable *Xmax*, enhancing the potential to exploit sparsity. (3) By rearranging operands to permit dovetailing of two operations into one multiplication, which omits "0" PPs processing, an out-of-order PE-line computing scheduler (OPCS) increases hardware usage by 1.81.

In addition to conventional solutions, the memristor crossbar architecture presents itself as a promising design choice owing to its extensive parallelism, high density, low access power, and the potential for multi-level cell and 3D integration, as evidenced by recent studies[21–23]. In the form of programmable conductance, a single memristor cell has the capacity to store binary or multi-bit information, providing a versatile platform for both analog and digital IMC operations, as discussed in the literature[24]. The execution of vector matrix multiplication (VMM) necessitates a single read cycle. In order to execute matrix-to-matrix multiplication operations, it is possible to split the input matrix into input vectors. Subsequently, the MatMul calculations may be carried out by using multiple VMMs. This "analog" approach to vector-matrix multiplication has the potential to yield significantly greater efficiency than any digital ASIC, particularly when the crossbar array dimensions are scaled to the greatest feasible extent, as suggested by prior research[25].

Yang et al.[26] introduced the concept of ReTransformers, marking the inaugural deployment of a resistive random access memory (RRAM)-based transformer architecture in the realm of NLP. Significantly, they integrated dot product operations and the softmax function into their design, recognizing these as the crucial components responsible for both energy consumption and time usage during the self-attention operation of the transformer. In their study, Yang et al.[27] introduced a comprehensive circuit design for the transformer utilising memristors. The proposed implementation consists of several key components: (1) A memristor crossbar module is employed to retain the weights of the transformer and carry out vector-matrix multiplications. (2) An analog signal memory module is utilised to directly store the analog signal in near-memory mode, eliminating the need for data transfer. (3) Function circuit modules are incorporated to facilitate five essential transformations: Softmax, Layer Normalization, ReLU, Multiply-add, and Residual. (4) A timing signal generation module is employed to schedule operations effectively within the circuit. By integrating these components, the authors present a novel approach to realise the transformer architecture using memristors. The proposed transformer circuit used analog signals to calculate without analog to digital converters (ADCs), digital to analog converters (DACs), or digital memory. PSPICE character image recognition investigations were used to prove the circuit's functionality.

This paper presents a novel low-complexity and real-time hardware-friendly architecture for the transformer algorithm. The contributions of this work could be summarized as follows:

- A novel, efficient memristor implementation of the transformer algorithm's main functions, specifically MatMul.
- A 10x acceleration of the transformer's self-attention operations using memristors-based HW when compared to its digital equivalent counterpart.
- Efficiently utilize the IMC characteristics of emerging memristor devices to reduce the power and improve the speed of the conventional transformer algorithm. To achieve this, this work presents:
  - Analyzing the transformer's computing complexity and identifying the main functions that need be to be accelerated in order to improve power, latency, and cost. In the transformer model, when it comes to the self-attention phase, the computations that have the biggest impact on latency and power consumption are MatMul and Softmax.
  - Mapping the algorithmic operations into corresponding logic compatible with the memristor crossbar architecture. To solve the low crossbar utilization challenge, a design with finer granularity should be utilized.
  - Quantifying the proposed approach's area, power, and latency and analyzing the system's accuracy.
- A significant acceleration for the MAC operation in the self-attention mechanism of a transformer.

- The proposed method presented a high-speed and an energy-efficient design with a moderate drop in accuracy when tested on MNIST dataset.

The remainder of the paper is organized as follows: Section II explains the architecture of the proposed vision transformer. Section III consolidates the significant research and results reported in the transformers model and its implementation in different hardware architectures, including memristor-based IMC designs. Moreover, the simulation results for the presented transformer network hardware accelerator are discussed. Section IV include the discussion about the proposed work limitation. Finally, Section V concludes the findings and demonstrates future studies.

Architecture

The proposed memristor architecture for the transformer algorithm with novel methodologies for RRAM-based IMC design that are particularly suited for self-attention mechanisms is presented here along with the simulation results. As illustrated in Fig. 2, the multi-head self-attention layer consists of many heads, each of which includes three trainable matrices: the Query matrix (W_Q), the Key matrix (W_K), and the Value matrix (W_V). These matrices are implemented as linear layers. The incorporation of this multi-head self-attention mechanism serves to enhance model performance by allowing it to focus on different positions within the data. To compute the attention scores, a scaled dot-product attention layer is employed, and its computation is described by the following equation:

$$Attention(Q,K,V) = Softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V,$$

The scaling factor applied to the dot-product attention is proportional to the square of the depth. This scaling is employed to mitigate the expansion of the dot product's magnitude for high depth values, as it can lead to the softmax function assigning tiny gradients to certain positions, causing an overly challenging softmax operation.

In this context, we have matrices X $\in R^{n*d_{model}}$, W_Q, $W_K \in R^{d_{model}*d_k}$, and $W_V \in R^{d_{model}*d_v}$, where Q and K $\in R^{n*d_k}$, and V $\in R^{n*d_v}$. Here, d_{model}, d_k, and d_v represent the dimensions of the model, key, and value matrices, respectively, and the resulting matrix $\in R^{d_k*d_v}$.

The scaled dot-product attention layer is composed of three essential operations: MatMul, scaling, and softmax. The MatMul operation entails the multiplication of matrix Q with the transpose of matrix K. Following that, the scaling operation involves the multiplication of the MatMul result by a scaling factor represented as $\frac{1}{\sqrt{d_k}}$

. The subsequent MatMul procedure involves the multiplication of the softmax outcome with the matrix V. This series of operations is designed to maintain the integrity of the tokens of interest while efficiently attenuating the influence of irrelevant tokens. The process of multiplying the concatenation of attention outcomes from each head with the weight matrix W_O leads to the formation of multihead attention, which is presented in the following equation:

$$MultiHead(Q,K,V) = Concat(head^1, ..., head^h) * W_O, where head^i = Attention(X * W_Q^i, X * W_K^i, X * W_V^i).$$

Where h is the number of heads in multi-head attention.W_Q^i, $W_K^i \in R^{d_{model}*d_k}$, $W_V^i \in R^{d_{model}*d_v}$, $W_O^i \in R^{hd_v*d_{model}}$.

Lastly, a point-wise feed-forward network is made up of multiple fully connected layers separated by a ReLU activation. The following equation presents the position-wise feed-forward layer after passing the output through a self-attention:

$$FFN(X) = max(0, X * W_1 + b_1) * W_2 + b_2,$$

Where $W_1 \in R^{d_{model}*d_{ff}}$, $W_2 \in R^{d_{ff}*d_{model}}$ and the result FFN(x) $\in R^{d_{model}}$. Furthermore, d_{ff} is the hidden layer dimension. By observing the number of encoding and decoding stacks presented in Fig. 3, it can be concluded that the majority of computations: MatMul and Softmax account for the majority of delay and power consumption

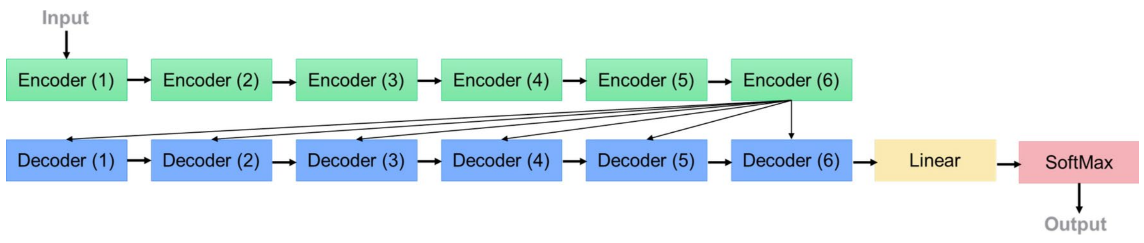

Figure 3. Encoder and decoder stacks in the transformer model structure. The typical basic structure of the transformer model uses six encoders and decoders.

in the self-attention phase calculations of the transformer model. For that reason, this work concentrates on accelerating these two operations.

As for computations such as the feed-forward and the crossbar architecture, similar procedures will be used as presented in the literature[28,29]. Our work follows the same procedure for the RRAM-based crossbar for AI inference[30,31]. Regarding the scaling operation, the scale operation in the scaled dot product attention, as shown by the scale factor $\frac{1}{\sqrt{d_k}}$, is associated with the scaling operation. The dimensionality of the trained matrices W_Q and W_K is represented by the variable d_k. In the study conducted by Vaswani et al.[2], it was shown that the value of d_k most often used was 64. The necessary scaling procedure is achieved by using a 3-bit left-shift, and then storing the outcomes on the RRAM crossbar for further processing. In cases when the square root of d_k does not yield a power of 2, the scaling operation is executed by using a combination of bit shifting and constant multiplication. This involves storing the constant value in the RRAM crossbar array and then multiplying this constant with the appropriate matrices. Furthermore, the author's prior work with the shift-based RRAM architecture can be used for this work implementation[32].

Results

Implementation of matrix-matrix multiplication

The flowchart illustrating the overarching training and inference procedures is illustrated in Fig. 4. The training phase is conducted within the software environment, where the weight matrix is transformed into the memristor's conductance matrix using a designated mapping algorithm. Subsequently, the inference stage is executed on the hardware platform, leveraging a hybrid circuit model. In order to account for the discontinuous nature of conductance values in the memristor, a mapping method is used to establish the relationship between a given operand and its corresponding resistance value.

In this section, the focus is on the implementation details of MatMul in the proposed memristor-based transformer's design. Results show a 10× acceleration of transformer self-attention via memristor-based HW compared to its digital counterpart.

During the first phase of the inference stage, the values of K and V are initialised and thereafter stored in two separate memristor crossbars. In order to perform the MatMul operation between the matrices Q and K, denoted as $Out = Q \times K^T$, inside the memristor-based IMC module, it is necessary to store one of the intermediate results, either Q or K, in a memristor crossbar, as in equation,

$$\begin{bmatrix} q_{11} & \cdots & q_{1n} \\ q_{21} & \cdots & q_{2n} \\ \vdots & \ddots & \vdots \\ q_{m1} & \cdots & q_{mn} \end{bmatrix} \times \begin{bmatrix} k_{11} & \cdots & k_{1p} \\ k_{21} & \cdots & k_{2p} \\ \vdots & \ddots & \vdots \\ k_{n1} & \cdots & k_{np} \end{bmatrix} = \begin{bmatrix} out_{11} & \cdots & out_{1p} \\ out_{21} & \cdots & out_{2p} \\ \vdots & \ddots & \vdots \\ out_{m1} & \cdots & out_{mp} \end{bmatrix}$$

In the context of this discussion, it is assumed, without loss of generality, that the transpose of matrix K, denoted as K^T, is stored inside the memristor crossbar. As shown in the study on memory architectures by Abunahla et al.[33], the programming of a memristor crossbar necessitates a row-by-row approach. The cumulative

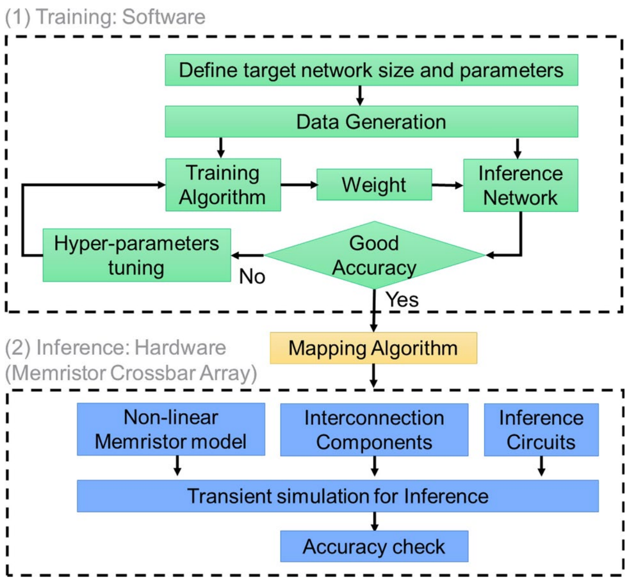
(1) Training: Software
Define target network size and parameters
Data Generation
Training Algorithm
Weight
Inference Network
Hyper-parameters tuning
Good Accuracy
No
Yes
(2) Inference: Hardware (Memristor Crossbar Array)
Mapping Algorithm
Non-linear Memristor model
Interconnection Components
Inference Circuits
Transient simulation for Inference
Accuracy check

Figure 4. Flowchart depicting the comprehensive training and inference procedures.

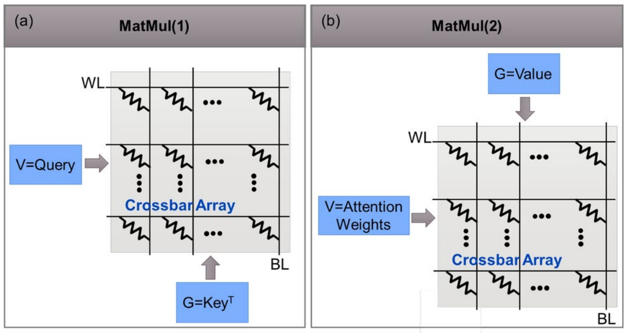

Figure 5. The suggested approach involves the utilization of two memristor devices to enhance the acceleration of scaled dot-product attention.

Characteristics	Values
GPU	Intel Iris Plus Graphics 2020
GPU cores	64 execution cores
Processor	2.0GHz quad-core
Main memory	32 GB
Power consumption	58W

Table 1. Detailed specifications of the experimental setup for GPU configurations.

RRAM Array	**Subarray size**	**128×128**
	Cell percision	**2 bit**
	R_{off}/R_{on}	**1$M\Omega$/ 100$k\Omega$**
ADC	Precision	5 bits
Shift and add	Precision	14 bits

Table 2. Detailed specifications of the experimental setup for the configurations related to the transformer model.

programming delay of a memristor crossbar with dimensions $N \times N$ is estimated to be *N* write cycles, assuming that each cycle is dedicated to programming a single row. The computational latency associated with VMM on a memristor crossbar architecture is equivalent to a single read cycle. In the interim, another memristor crossbar is initialize with *V*. After receiving the attention weights from the softmax function in the high-level code, $Out2 = attentionweights \times V$ operations are conducted on the memristor crossbar storing *V*. The full procedure is represented in Fig. 2a. As a remark, the calculation of $Out2 = attentionweights \times V$ cannot be done until the computation of $Out = Q \times K^T$ completes. Figure 5 presents the proposed memristor-based scaled dot-product attention acceleration. The choice for each matrix in the mapping to memristor device is shown in Fig. 5. A neuromorphic system using a memristor crossbar array performs matrix operations in an analog method which is done by the integration of computation into memory using the KCL rule by detecting the current at each column from the multiplication of voltage and conductance of the memristor.

In our design, *K*, *Q*, and *V* have negative values and Q, and *V* will be mapped to conductance values to put them in the memristor crossbar array. The conductance G_{ideal} is the cross-point device conductance matrix for an ideal crossbar array. Because negative conductances are not possible, mapping an arbitrary matrix *W* to G_{ideal} requires first executing a uniform shift to handle any negative values. If *A* includes negative values, the constant W_{SHIF} is added to all entries, and this impact on the vector-matrix multiplication is eliminated at the final step by subtracting $W_{SHIF} \times sum(X)$, where *sum*(*X*) is the summation of the input vector. There are two linear mapping coefficients produced[25]:

$$a = \frac{G_{on} - G_{off}}{W_{max} - W_{min}}$$
$$b = G_{on} - a \times W_{max}$$

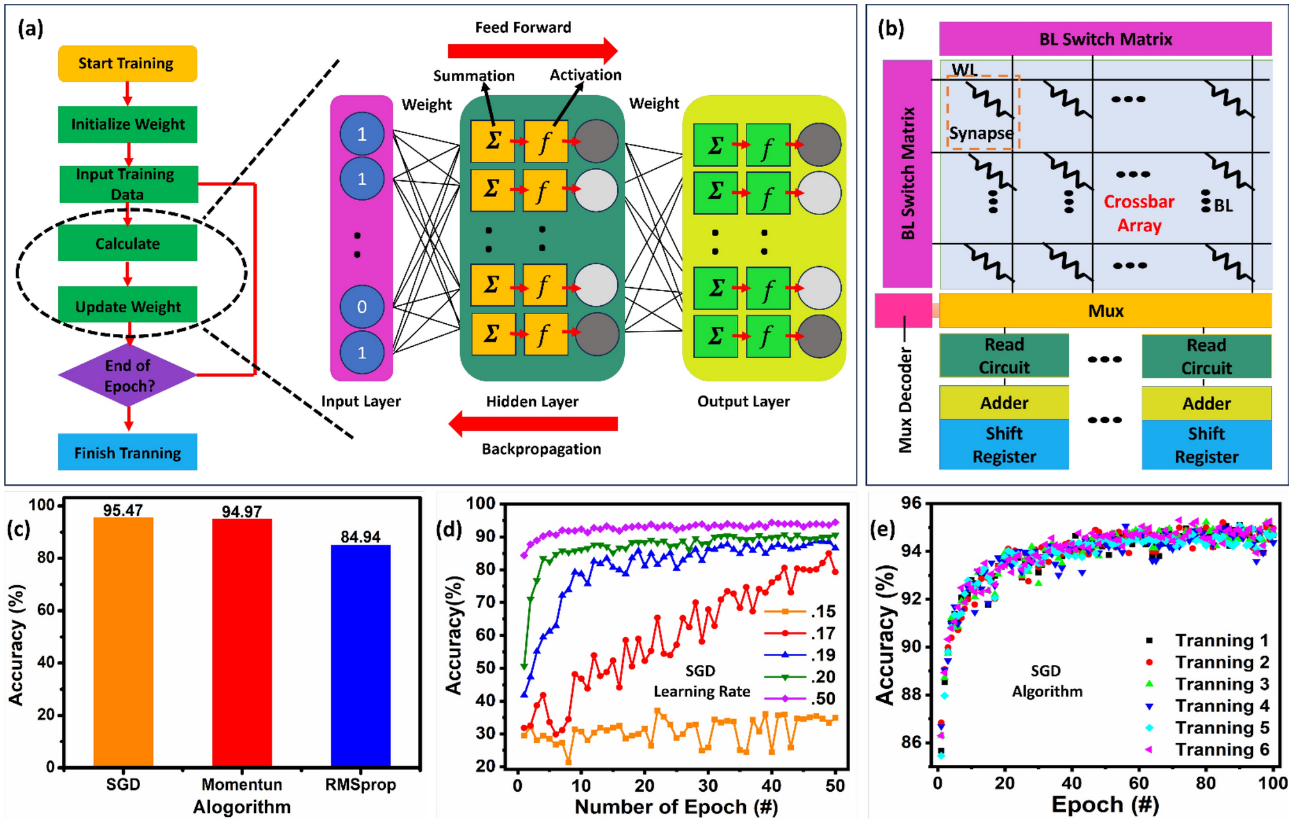

Figure 6. (**a**) Flow chat of training process, (**b**) NeuroSim V3.0 crossbar array design for its synaptic cores, (**c**) The MLP's experimental accuracy in classification using various training algorithms. (**d**) The precision concerning the learning rate and training epochs within the memristor array. (**e**) The precision of NeuroSim V3.0 output simulation using ideal devices with continuous 6 training.

Performance metrics	Value
Accuracy	95.47%
Area	6895.7 μm^2
Latency	15.52 s
Energy	3 mJ
Leakage power	59.55 μW

Table 3. The accuracy of MatMul outputs throughout 100 writing epochs and circuit-level performance indicators were evaluated based on the simulation results obtained from NeuroSim.

The maximum and lowest memristor conductance values are G_{on} and G_{off}, while the maximum and minimum matrix values are W_{max} and W_{min}. Finally, G_{ideal} may be computed by the linear transformation[34]:

$$G_{ideal} = a \times W + b$$

The lowest and greatest resistance range demonstrated by our memristor crossbar array design and employed in the mapping procedure is 1 M-100 k Ω. Upon the completion of the VMM operation, the output results undergo a process of shifting back by the following procedure:

$$\begin{bmatrix} q_{11} & \cdots & q_{1n} & 1 \\ q_{21} & \cdots & q_{2n} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ q_{m1} & \cdots & q_{mn} & 1 \end{bmatrix} \times \begin{bmatrix} k_{11} + |k_{min}| & \cdots & k_{1p} + |k_{min}| \\ k_{21} + |k_{min}| & \cdots & k_{2p} + |k_{min}| \\ \vdots & \ddots & \vdots \\ k_{n1} + |k_{min}| & \cdots & k_{np} + |k_{min}| \end{bmatrix} = \begin{bmatrix} out_{11} & \cdots & out_{1p} \\ out_{21} & \cdots & out_{2p} \\ \vdots & \ddots & \vdots \\ out_{m1} & \cdots & out_{mp} \end{bmatrix}$$

By removing the contribution of $|k_min|$, the output results are identical to those in the first equation.

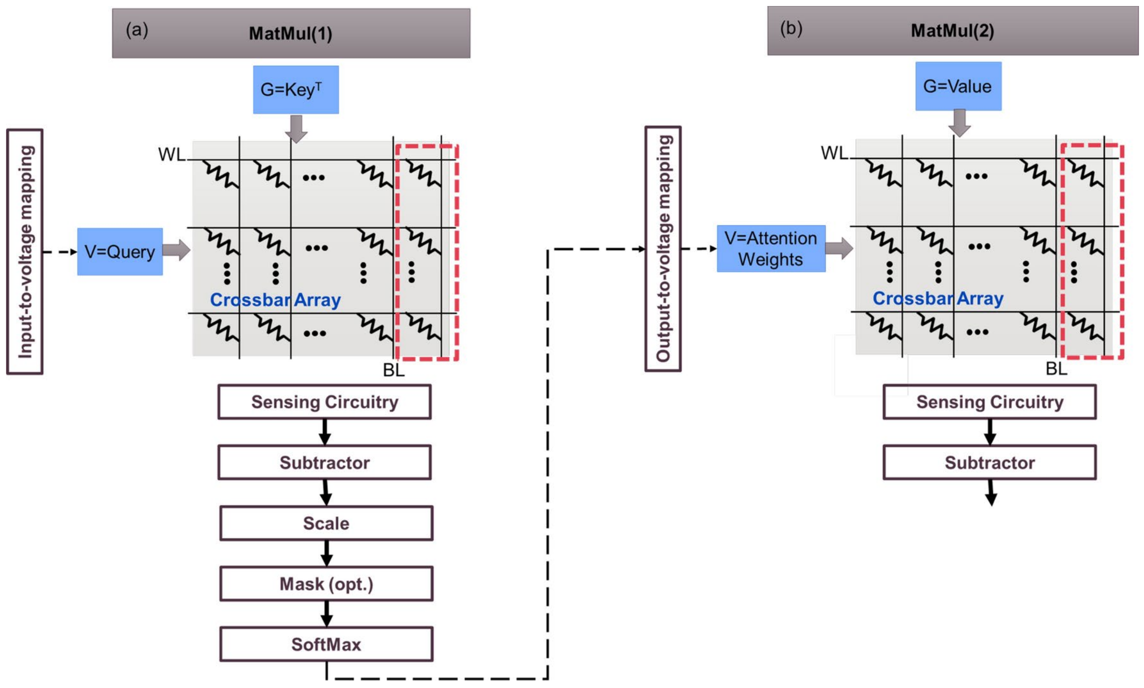

Figure 7. (**a**,**b**) show the first and second layers of fabricated MR crossbars to accelerate the VMM of the classification task during the inference phase.

$$\left[out_1 - (|k_{min}| \times \sum_{i=1}^{n} q_i) \;\; out_2 - (|k_{min}| \times \sum_{i=1}^{n} q_i) \;\; \ldots \;\; out_n - (|k_{min}| \times \sum_{i=1}^{n} q_i)\right]$$

Where OUT_n is the output calculated from multiplying and adding the Q by the K^T. The biases are represented by $k_{13} - k_{15}$ and multiplied by 1.

In our study, we deliberately chose certain matrices to be mapped onto memristors due to their status as parameters that undergo the least frequent alterations. However, it's crucial to acknowledge that even these relatively stable matrices experience changes over time. This recognition underscores the necessity for future devices to exhibit resilience and a marked reduction in write energy, ensuring that they can effectively handle dynamic requirements while maintaining overall efficiency.

Hardware implementation of softmax function

The proposed memristor design of the Softmax calculations is divided into two parts:

1. The use of RRAM-based compare and select logics is employed in order to determine the maximum value, denoted as x_{max}.
2. The use of look-up tables for the execution of exponential and logarithmic functions.Negative numbers are used in exponential step and division is substituted with logarithm as shown in the following equation:

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{d_k} e^{x_j}} = \frac{e^{x_i - x_{max}}}{\sum_{j=1}^{d_k} e^{x_j - x_{max}}} = exp[x_i - x_{max} - log(\sum_{j=1}^{d_k} e^{x_j - x_{max}})]$$

Each element x_i ($i = 0, ..., d_k$ - 1) in the input vector X of length d_k is subjected to subtraction by the maximum element in the same vector, denoted as x_{max}. For the max operation, it will be implemented by *xor* logic on the memristor. To execute exponential and logarithmic functions, we employ lookup tables for implementation, as provided by the sources referenced in[35,36].

Simulation setup

In order to evaluate the feasibility of integrating memristor technology from an industrial standpoint and to measure the hardware efficiency of the RRAM-based IMC architecture, we use NeuroSim[37]. This approach involves estimating various parameters, including area, latency, and energy consumption. NeuroSim will serve as a valuable tool for the assessment of circuit-level performance. It enables the creation of an integrated framework characterized by a hierarchical structure, commencing at the device level, encompassing synaptic device properties, advancing to the circuit level, involving array architectures, and finally extending to the algorithm level, which includes neural network topology. This comprehensive methodology enables a precise assessment of learning accuracy and real-time monitoring of circuit-level performance metrics throughout the online learning process. Employing the transformer model as a case study, this research delves into the impact of non-ideal properties inherent in emerging nonvolatile memory (eNVM) devices with an "analog" nature. Furthermore, it conducts a comparative analysis to discern the trade-offs between architectures based on digital and analog eNVM.

In our experiment, the proposed RRAM-based IMC transformer architecture is compared to a GPU platform and digital implementation. Benchmarks that run on the GPU platform are based on Python. Table 1 summarizes the GPU platform configurations. An average/lower-end GPU is used since the speed up is in the inference stage, and so such a GPU is more representative of real inference environments.

Utilizing the RRAM and peripheral configurations as detailed in Table 2 and informed by prior research[38], the simulation of RRAM circuit parameters is conducted employing NeuroSim[39]. The precision of the RRAM cells is standardized at 2 bits, aligning with the approach taken in a previous study[40]. The read and write pulse timings for RRAM cells are established based on data from[41]. The resistance values, denoted as R_{on} and R_{off} , for RRAM crossbars are derived from the reference provided in[42]. The framework for designing exponential and logarithmic lookup tables within the hybrid softmax architecture is implemented using the methodology outlined in Brito et al.[36]. Furthermore, the CMOS baseline implemented in the softmax design is a modified version of the design that was first proposed in a study by Du et al.[43].

Simulation results

The foremost and pivotal metric in any machine learning or neural network application is accuracy. In this context, the model has achieved a remarkable accuracy rate of 95.47%. This achievement underscores the effectiveness of the design and implementation in accurately classifying data, a fundamental requirement in numerous real-world applications, such as image recognition in our specific case. Furthermore, as presented in Table 3, the area utilization stands at 6895.7 μm^2, indicative of a relatively compact design. This compactness renders it well-suited for deployment in embedded systems or situations where physical space is at a premium. Moreover, the design showcases an impressive latency of 15.52 seconds, demonstrating its efficiency in data processing speed. This attribute is particularly valuable in applications necessitating timely responses. Additionally, with an energy consumption of a mere 3 *mJ*, the design proves highly efficient in energy utilization. This characteristic has the potential to extend the battery life of mobile devices and reduce operational costs in energy-intensive environments like data centers. Furthermore, the leakage power, measured at 59.55 μW, attests to minimal power wastage during idle periods, enhancing overall energy efficiency. It mitigates power loss when the circuit is not actively engaged in data processing. The combination of these outstanding results in terms of accuracy, area utilization, latency, energy efficiency, and leakage power renders this design versatile and suitable for a broad spectrum of real-world applications. It can be seamlessly integrated into edge devices for the IoT and deployed in high-performance computing clusters for demanding deep learning tasks.

A quantitative comparison with the prior research conducted by Yang et al.[26] underscores the distinctions between their retransformer design and our proposed design, particularly in terms of latency and power consumption. Yang et al.'s retransformer exhibited power consumption of approximately $3 \times 10^{-1} W$ and a latency of 4250*ns*. In contrast, our design demonstrated significantly improved efficiency with a power consumption of $5.955 \times 10^{-5} W$ and a latency of $1552 \times 10^{7} ns$.

Notably, Yang et al.'s design consumed more power but exhibited reduced time for matrix multiplication by employing matrix decomposition to avoid writing intermediate results and eliminate data dependencies. As part of our future work, we intend to incorporate weight approximation techniques to further optimize the matrix multiplication operation and subsequently reduce latency.

The examination of data pertaining to long-term potentiation (LTP) and long-term depression (LTD), which signify the enhancement and reduction of synaptic connections between neurons in the brain, was conducted using a read voltage of 0.3 V and a pulse width of 7.2 ns. In order to incorporate both potentiation (enhancement of synaptic strength) and depression (reduction of synaptic strength) effects in a single memristor device, a write voltage of 1 V and -1 V was applied. This was achieved by utilizing a form of bipolar programming in synapses based on memristor technology, with a pulse width of 7.2 ns. The researchers in Ref.[41] established the upper limit of conductance at 0.00001 S and the lower limit at 0.000001 S, while maintaining a R_{off}/R_{on} ratio of 10. The maximum permissible quantity of LTP and LTD states was established at 100, respectively. The choice of a 7.2 ns pulse width is of significant importance, as it governs the precise timing and duration of electrical pulses applied to the memristors. This pulse width plays a critical role in determining the extent and duration of changes in synaptic strength during LTP and LTD processes. It is a key factor in controlling the plasticity and stability of neural connections, which is a crucial aspect of neural network learning and memory storage.

The training process of MLP NeuroSim V3.0 is illustrated in Fig. 6a through a detailed flowchart. Before introducing the training data, the weights of the input layer and hidden layer are initialized randomly, specifically with values of 0.15, 0.17, 0.19, 0.10, and 0.50. The training process involves two key operations: feed-forward (FF) and backpropagation (BP), both of which are elaborated upon. In the feed-forward phase, the training data are fed into the input layer, progressing through the network via weighted summation and neuron activation until reaching the output layer. The results obtained at the output layer during feedforward are then compared with the corresponding labels to determine the deviation for subsequent steps. The backpropagation process involves the propagation of deviations from the output layer back to the input layer. During this phase, adjustments to the weights in each layer are made in a manner that anticipates minimal deviation. This iterative process of feed-forward and backpropagation is crucial for the training of MLP NeuroSim V3.0, enabling the neural network to learn and optimize its weights for improved performance in subsequent tasks.

The study of device characteristics was conducted using MLP NeuroSim V3.0 at the 32nm technology node. This analysis used a synaptic core architecture that is based on the parallel read-out of analog eNVM, as seen in Fig. 6b[44]. The NeuroSim V3.0 framework comprises of two neural network layers that are completely linked. These layers consist of 400 neurons in the input layer, 100 neurons in the hidden layer, and 10 neurons in the output layer. The multiplication between a vector and a matrix is performed using a memristor crossbar parallel read-out array, as described by the equation provided.

$$\begin{bmatrix} I_1 \\ \vdots \\ I_N \end{bmatrix} = \begin{bmatrix} G_{11} & \cdots & G_{1N} \\ \vdots & \ddots & \vdots \\ G_{N1} & \cdots & G_{NN} \end{bmatrix} \times \begin{bmatrix} V_1 \\ \vdots \\ V_N \end{bmatrix}$$

Within the present framework, we designate the variables V_n, I_n, and G_{ij} as the input voltage, output current, and conductance, correspondingly. Consequently, the measured value I_n is obtained from every bit line. The training process consists of two crucial steps, namely feedforward and backpropagation. During the feedforward phase, the main objective is to determine the difference between the actual value and the obtained output. Subsequently, during the backpropagation process, the weights are adjusted based on the feedback obtained from the feedforward step, and this iterative procedure is repeated until the error is minimized. For the training phase, the MNIST training dataset is utilized as input over the course of 250 epochs, with each epoch involving 8000 training images. The simulation results reveal the effectiveness of the ideal device, which has a weight range bounded between -1 and 1. Memristor-based hardware is typically utilized for inference, where the network applies its learned knowledge to make predictions. Upon the completion of the NeuroSim simulation, the outcomes are shown in Table 3.

To assess the performance of the constructed neural network, the post-processed MNIST dataset is employed for training and testing the memristor-based MLP neural network. Various training algorithms, including SGD, momentum, and RMSprop, are compared, as illustrated in Fig. 6c. The results highlight that the SGD method yields the highest recognition accuracy for the MLP, reaching 95.47%. Consequently, the SGD algorithm is adopted for subsequent training and testing. Furthermore, an exploration of the relationships between recognition accuracy, learning rate, and epoch is conducted. To scrutinize the impact of the learning rate within the training process of the SGD algorithm, weight statistics are graphically represented in Fig. 6d. The input layer's weights are initialized randomly at learning rates of 0.15, 0.17, 0.19, 0.10, and 0.50. Observations reveal significant variations in weight distribution after multiple training epochs, indicating a learning process that enhances recognition accuracy, particularly evident at a learning rate of 0.50. The analysis demonstrates a notable increase in accuracy with an escalating learning rate. Conversely, when the rate falls below 0.15, the optimization for recognition results becomes less discernible. After careful consideration, a learning rate of 0.50 is selected, as it strikes a favorable compromise and yields high accuracy. We have conducted a thorough analysis by measuring the accuracy of weight changes over six runs for 100 epochs. Our findings reveal a consistent system performance, with an average accuracy of 95.47%, as illustrated in Fig. 6e. To provide a comprehensive understanding of the results. This additional information supports the stability of our proposed approach, demonstrating its ability to maintain high accuracy levels even under the influence of device variability. The potential for even higher accuracy is suggested, prompting further exploration through tests conducted at increased epoch levels[45,46].

The selection of the MNIST dataset for evaluation is due to the need to perform end-to-end analysis at the circuit level. The same methodology can be applied to other networks and datasets. We added the next subsection, which is a case study of a transformer with a translation task, to prove the concept is valid, but further optimization is left for future work.

Case study: transformer with translation

Here, the effect of using the Transformer model in NLP language translation is explored. Many variants of transformers are used in NLP, and researchers give their models their own names. For example, GPT-2, a generative pre-training for transformer, is a transformer created by Open AI with pre-training. It is good at generating texts that are news magazines which was documented by the economist. BERT, bidirectional encoder representations from transformers, which was created by the Google AI language team, is another famous transformer used for learning text representations. It is also known as T5, which stands for Text-to-Text Transfer Transformer, and it was also created by Google. It is a multitask transformer that can do question answering among a lot of different tasks[47]. So, transformers are suitable for a wider range of NLP applications. In this work, we chose text translation as the main application. The Portuguese-English translation dataset was used. This dataset contains approximately 50000 training examples, 1100 validation examples, and 2000 test examples. The key concept underlying a transformer model is self-attention, which is the capacity to pay attention to diverse parts of an input sequence in order to compute a representation of it. The Transformer, as previously explained, builds stacks of self-attention layers. Instead of employing RNNs or CNNs, a transformer model uses stacks of self-attention layers to handle variable-sized input. There are many benefits to this generic design. For starters, it makes no assumptions about the data's temporal or spatial connections. This is perfect for dealing with a set of objects (for example, StarCraft units). Second, instead of being computed in a series like an RNN, layer outputs may be calculated in parallel. Third, elements that are far apart may influence each other's output without having to go through a lot of RNN steps or convolution layers. Finally, it is capable of learning long-term dependencies. This is a challenge in many sequential tasks. This architecture, however, has drawbacks. The output for a time-step in a time-series is computed using the complete history rather than just the inputs and present hidden state. This might be less effective. If the input has a temporal/spatial link, such as text, some positional encoding must be provided, otherwise the model will just see a bag of words. TensorFlow and Keras were used to simulate the transformer network for NLP applications. To keep this example small and relatively fast, the values for the hypermeters (num of layers, dimension of the model, dff, num-heads, dropout-rate, num-epochs) have been reduced. The base model described in Ref.[2] is used with the following parameters:

- num of layers = 6
- dimension of the model = 512
- dff = 2048
- num of heads = 8
- dropout rate = 0.1An accuracy of 85.34% in a translated Portuguese-to-English data set is achieved. Moreover, a higher number of memristor conductance levels can be used to further increase the mapping accuracy and, as a consequence, the overall system performance.

In the MR crossbar hardware implementation presented in Figure 7(a,b), this operation can be translated into adding a fourth column of MR devices to hold the minimum weight value as it was mentioned in the previous section for the matrix-matrix multiplication mapping steps; hence, a 64 × 61 and a 60 × 65 crossbar size are used for simulation. Then each output is subtracted from the last column. The output results are input to the output layer, as illustrated in Fig. 7. Moreover, other VMM operations take place; then, the results are shifted back to produce the final output.

The detailed framework for the self-attention Transformer with translation is shown in Fig. 8. Matrix multiplication using memristor crossbar will result in (64 × 60 × 8 heads), and (60 × 60 × 8 heads) for the first and second MatMul blocks, respectively. This will result in 59,520 MAC operations in total. Consequently, for any digital implementation, the first block will result in ((64 MUL + 63 ADD) × 60 × 5 bits × 8 heads). To be specific, for an array of size 64 rows with 60 columns, per column: 64 multiplications and 63 additions times the number of bits assumed, which will be 5 in our case. So, for the whole array with 8 heads, the total number of MAC operations is given by (64 MUL + (64-1) ADD) × 60 × 5 bits × 8 heads. While the second block will need to perform ((60 MUL + 59 ADD) × 60 × 5 bits × 8 heads). This will give rise to 590400 MAC operations in digital implementation. The detailed matrix-multiplication (MM) calculations of the digital implementation steps are shown in Fig. 9. Moreover, the dynamic power consumption and switching behaviour in digital circuits implementing DNNs are influenced by the distribution of operands fetched from the input and weight matrices[48]. The calculations can be different based on the length of the input. In our work, the length of the input is 60.

Discussion

Results show a 10× acceleration of transformer self-attention via memristor-based HW compared to its digital counterpart. The NeuroSim memristor simulation results show 93.37% with approximately 6% drop compared to the original design, which showcases a well-balanced solution that excels in terms of accuracy, energy efficiency, and compactness, as presented in Table 3. These strengths position the design for success in various application domains, making it a strong choice for practical implementation in neural network-based systems. Future work may include incorporating architecture or circuit-level optimizations for the crossbars.

In transformer models, a significant challenge arises from MatMul operations, particularly those involving excitation vectors within attention computations, which are constrained by the finite endurance of memristors, potentially resulting in degradation and reliability issues due to the high volume of repetitive write-erase cycles. Additionally, memristor-based storage devices face limitations in transformer models, including the need for frequent writes due to low precision and the requirement for multiple write-and-verify steps when aiming for multi-bit conductance precision, particularly problematic in self-attention scenarios typical of transformers, given the limited write endurance of memristor devices.

The problem doesn't affect the MatMul operations used for generating *K*, *Q*, and *V* from static weight matrices in input embeddings, as these are stationary and can be efficiently handled by crossbar arrays without causing

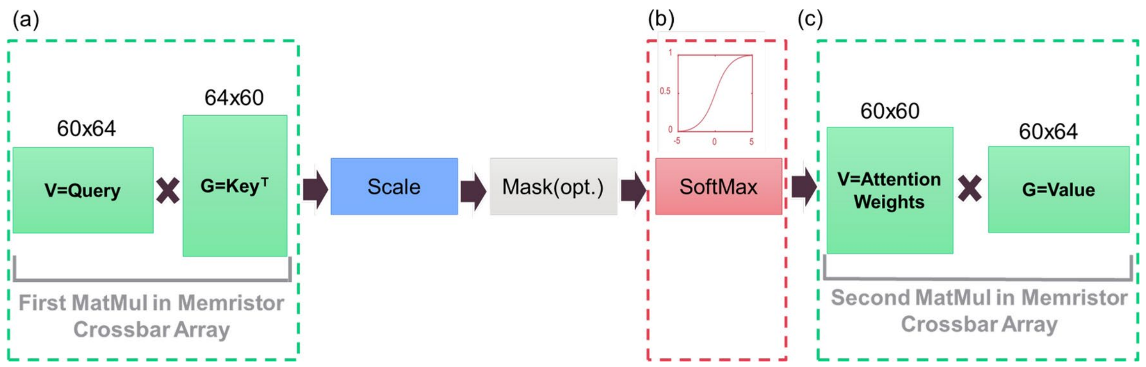

Figure 8. The algorithmic flow of the attention mechanism reveals that the operations denoted as (**a**), (**b**), and (**c**) are the most resource-intensive in terms of both power consumption and processing latency. Consequently, these specific operations are earmarked for implementation within a memristor crossbar architecture to expedite the scaled dot-product attention process.

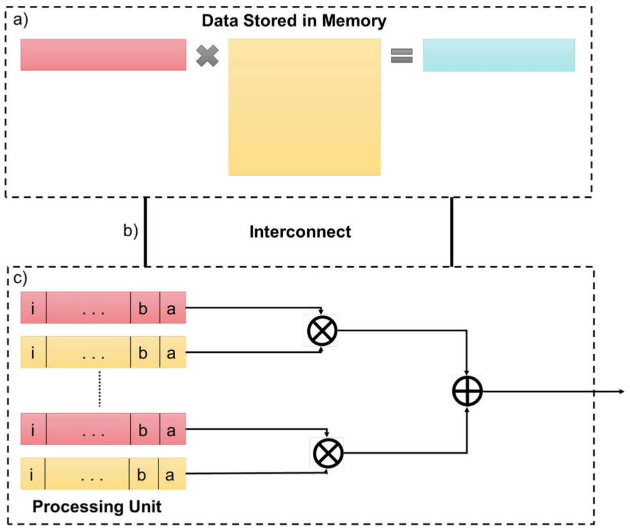

Figure 9. (**a**) Matrix-multiplication (MM) is the core building block of many applications, including neural networks. (**b**) Data operands are fetched from the memory serially and moved to the processing unit through the interconnects. (**c**) Digital implementation of the multiply-add between the different operands.

endurance problems when mapped to memristors. However, the performance bottleneck in transformers arises from MatMul operations in self-attention, which we are actively working on accelerating in our project.

Our research successfully accelerated the MAC operation in the self-attention mechanism of a transformer, significantly enhancing its performance. However, this acceleration led to a higher number of write operations, posing a challenge due to the elevated power consumption associated with current memristor technology.

Many challenges are related to memristor devices, such as limited endurance levels compared to CMOS and programming speeds. Nonetheless, a lot of work is focused on utilizing different materials, architectures, and designs to achieve ultra-fast responses and higher endurance levels[49–51]. Zhang et al.[49] demonstrated the resistive switching characteristics of the PBDTT-BQTPA polymer memristors, which, with a record-high production yield of 90% and remarkable reliability, not only enable the implementation of multilevel logic operations but also showcase the material's potential for achieving high-speed information processing with minimal energy consumption in in-memory computing applications. Liu et al.[50] proposed a room-temperature-fabricated perovskite synaptic device, featuring passivation for enhanced material properties, that stands out in neuromorphic technology. With a switching ratio exceeding 103, it offers multi-level synaptic plasticity for versatile information storage. The device's improved optoelectronic characteristics, achieved through passivation, contribute to its ultra-fast response frequency of up to 4.17 MHz, setting a new benchmark for speed in perovskite-based synaptic devices. Notably, it showcases attojoule-level energy efficiency, marking a significant advancement in low-energy consumption for artificial synaptic technology in neuromorphic systems. Poddar et al.[51] proposed a Re-RAM developed with perovskite Quantum Wire/Quantum Well (QW/NW) arrays which exhibited a remarkable ON/OFF ratio of $\sim 10^7$, showcasing its potential for multilevel data storage, while the utilization of perovskite material contributed to efficient down-scalability, ultra-fast programming speed (~ 100 ps), and excellent energy efficiency, positioning it as a promising candidate for future non-volatile memories with advanced features. Moreover, in-situ memristor programming has been utilized in different works, such as but not limited to the ReTransformer architecture by Yang et al.[26]. ReTransformer utilizes a ReRAM-based Processing-in-Memory (PIM) architecture, employing ReRAM crossbar arrays for computations in the Transformer model. Its key innovation is an optimized MatMul operation, which resolves data dependency challenges and improves the utilization of ReRAM crossbars. These crossbars, equipped with computing controllers, form the backbone for executing scaled dot-product attention and feed-forward operations. Additionally, ReTransformer incorporates a hybrid softmax design and a sub-Matrix pipeline for further efficiency gains. Overall, this architecture significantly enhances computing efficiency and reduces power consumption compared to traditional GPU and ReRAM-based designs like PipeLayer.

To address these challenges, our focus shifted to investigating methods to reduce the increased write energy demand. Our goal is to strike a balance between optimizing the computational efficiency of self-attention in transformers and mitigating the amplified energy consumption resulting from intensified write operations in memristor-based systems, aiming for more sustainable and energy-efficient implementations in the future. Furthermore, device-level enhancements can be included in a future paper that discusses the architecture design and fabrication of our current proposed work.

Conclusion

This article presents a novel, efficient memristor implementation of the transformer algorithm's main functions, specifically MatMul. Results indicate a $\sim$ 10x acceleration of the transformer's self-attention operations using memristors-based HW compared to its digital equivalent counterpart. In comparison to the traditional transformer's self-attention scheme, the proposed method presented a high-speed and an energy-efficient design with 4.53% drop in accuracy when tested on MNIST dataset. This accuracy difference will be expected to reduce further if we further increase the training epochs with the optimization algorithm, SGD.

In the future, efforts will focus on implementing the outlined methodology on the vision transformer model, as they don't suffer from the quadratic increase in memory and computation requirements of traditional transformers. This will pave the way to implementing high-accuracy vision transformers for image classification tasks trained on large data. Also, we can expand other datasets other than MNIST in future work. Furthermore, in our future work, we will explore retention time, writing time, and endurance.

Data availability

The datasets generated and/or analysed during the current study are the MNIST[52] which are available online for public use.

Received: 2 November 2023; Accepted: 1 October 2024
Published online: 15 October 2024

References

1. Mahdi, O. & Nassif, A. B. Transformation invariant cancerous tissue classification using spatially transformed densenet. In *2022 Advances in Science and Engineering Technology International Conferences (ASET)* (ed. Mahdi, O.) 1–6 (IEEE, 2022).
2. Vaswani, A. *et al.* Attention is all you need. *Adv. Neural Inf. Process. Syst.*, 5998–6008 (2017).
3. Bettayeb, M., Hassan, E., Mohammad, B. & Saleh, H. Spatialhd: Spatial transformer fused with hyperdimensional computing for ai applications. In *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (ed. Bettayeb, M.) 1–5 (IEEE, 2023).
4. Bettayeb, M. et al. Adapting spatial transformer networks across diverse hardware platforms: A comprehensive implementation study. In *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)* (ed. Bettayeb, M.) 547–551 (IEEE, 2024).
5. Hassan, E., Bettayeb, M. & Mohammad, B. Advancing hardware implementation of hyperdimensional computing for edge intelligence. In *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)* (ed. Hassan, E.) 169–173 (IEEE, 2024).
6. Li, B. *et al.* Ftrans: energy-efficient acceleration of transformers using fpga. In: *Proc. ACM/IEEE International Symposium on Low Power Electronics and Design*, 175–180 (2020).
7. Liu, Y. *et al.* Roberta: A robustly optimized bert pretraining approach. Preprint at arXiv:1907.11692 (2019).
8. Wang, H. *et al.* Hat: Hardware-aware transformers for efficient natural language processing. Preprint at arXiv:2005.14187 (2020).
9. Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V. & Le, Q. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning* (ed. Bender, G.) 550–559 (PMLR, 2018).
10. Guo, Z. et al. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision* (ed. Guo, Z.) 544–560 (Springer, 2020).
11. Pham, H., Guan, M., Zoph, B., Le, Q. & Dean, J. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* (ed. Pham, H.) 4095–4104 (PMLR, 2018).
12. Laguna, A. F., Kazemi, A., Niemier, M. & Hu, X. S. In-memory computing based accelerator for transformer networks for long sequences. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (ed. Laguna, A. F.) 1839–1844 (IEEE, 2021).
13. Lu, S., Wang, M., Liang, S., Lin, J. & Wang, Z. Hardware accelerator for multi-head attention and position-wise feed-forward in the transformer. In *2020 IEEE 33rd International System-on-Chip Conference (SOCC)* (ed. Lu, S.) 84–89 (IEEE, 2020).
14. Peng, H. *et al.* A length adaptive algorithm-hardware co-design of transformer on fpga through sparse attention and dynamic pipelining. In: *Proc. 59th ACM/IEEE Design Automation Conference*, 1135–1140 (2022).
15. Qu, Z. *et al.* Dota: detect and omit weak attentions for scalable transformer acceleration. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 14–26 (2022).
16. Tu, F. et al. A 28nm 15.59 μj/token full-digital bitline-transpose cim-based sparse transformer accelerator with pipeline/parallel reconfigurable modes. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* Vol. 65 (ed. Tu, F.) 466–468 (IEEE, 2022).
17. Song, Y. et al. Dancing along battery: Enabling transformer with run-time reconfigurability on mobile devices. In *2021 58th ACM/IEEE Design Automation Conference (DAC)* (ed. Song, Y.) 1003–1008 (IEEE, 2021).
18. Qi, P. *et al.* Accommodating transformer onto fpga: Coupling the balanced model compression and fpga-implementation optimization. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, 163–168 (2021).
19. Peng, H. et al. Accelerating transformer-based deep learning models on fpgas using column balanced block pruning. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)* (ed. Peng, H.) 142–148 (IEEE, 2021).
20. Wang, Y. et al. A 28nm 27.5 tops/w approximate-computing-based transformer processor with asymptotic sparsity speculating and out-of-order computing. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)* Vol. 65 (ed. Wang, Y.) 1–3 (IEEE, 2022).
21. Bettayeb, M., Zayer, F., Abunahla, H., Gianini, G. & Mohammad, B. An Efficient In-Memory Computing Architecture for Image Enhancement in AI Applications. *IEEE Access* (2022).
22. Bettayeb, M., Tesfai, H., Mohammad, B. & Saleh, H. Asic-based implementation of random spray retinex algorithm for image enhancement. In *2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS)* (ed. Bettayeb, M.) 1–4 (IEEE, 2022).
23. Hassan, E., Bettayeb, M., Mohammad, B., Zweiri, Y. & Saleh, H. Hyperdimensional computing versus convolutional neural network: Architecture, performance analysis, and hardware complexity. In *2023 International Conference on Microelectronics (ICM)* (ed. Hassan, E.) 228–233 (IEEE, 2023).
24. Bettayeb, M., Halawani, Y., Khan, M. U., Mohammad, B. & Saleh, H. Memristor-based in-memory computing. In *In-Memory Computing Hardware Accelerators for Data-Intensive Applications* (ed. Bettayeb, M.) 97–121 (Springer, 2023).
25. Hu, M. et al. Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication. In *2016 53nd acm/edac/ieee design automation conference (dac)* (ed. Hu, M.) 1–6 (IEEE, 2016).
26. Yang, X., Yan, B., Li, H. & Chen, Y. Retransformer: Reram-based processing-in-memory architecture for transformer acceleration. In: *Proc. 39th International Conference on Computer-Aided Design*, 1–9 (2020).
27. Yang, C., Wang, X. & Zeng, Z. Full-circuit implementation of transformer network based on memristor. *IEEE Trans. Circuits Syst. I Regul. Pap.***69**, 1395–1407 (2022).
28. Chi, P. et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *ACM SIGARCH Computer Architecture News* **44**, 27–39 (2016).
29. Shafiee, A. et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News* **44**, 14–26 (2016).

30. Halawani, Y., Mohammad, B., Lebdeh, M. A., Al-Qutayri, M. & Al-Sarawi, S. F. Reram-based in-memory computing for search engine and neural network applications. *IEEE J. Emerg. Select. Top. Circuits Syst.* **9**, 388–397 (2019).
31. Halawani, Y., Mohammad, B. & Saleh, H. Design exploration of ReRAM-based crossbar for AI inference. *IEEE Access* **9**, 70430–70442 (2021).
32. Halawani, Y., Hassan, E., Mohammad, B. & Saleh, H. Fused rram-based shift-add architecture for efficient hyperdimensional computing paradigm. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)* (ed. Halawani, Y.) 179–182 (IEEE, 2021).
33. Abunahla, H., Halawani, Y., Alazzam, A. & Mohammad, B. Neuromem: Analog graphene-based resistive memory for artificial neural networks. *Sci. Rep.* **10**, 1–11 (2020).
34. Li, C. et al. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **1**, 52–59 (2018).
35. Halawani, Y. et al. RRAM-based CAM combined with time-domain circuits for hyperdimensional computing. *Sci. Rep.* **11**, 1–11 (2021).
36. Brito, D., Rabuske, T. G., Fernandes, J. R., Flores, P. & Monteiro, J. Quaternary logic lookup table in standard cmos. *IEEE Transactions on very large scale integration (vlsi) systems* **23**, 306–316 (2014).
37. Chen, P.-Y., Peng, X. & Yu, S. Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **37**, 3067–3080 (2018).
38. Peng, X., Liu, R. & Yu, S. Optimizing weight mapping and data flow for convolutional neural networks on rram based processing-in-memory architecture. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)* (ed. Peng, X.) 1–5 (IEEE, 2019).
39. Peng, X., Huang, S., Luo, Y., Sun, X. & Yu, S. 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)* (ed. Peng, X.) 32–5 (IEEE, 2019).
40. Long, Y., Na, T. & Mukhopadhyay, S. Reram-based processing-in-memory architecture for recurrent neural network acceleration. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **26**, 2781–2794 (2018).
41. Sheu, S.-S. et al. A 4mb embedded slc resistive-ram macro with 7.2 ns read-write random-access time and 160ns mlc-access capability. In *2011 IEEE International Solid-State Circuits Conference* (ed. Sheu, S.-S.) 200–202 (IEEE, 2011).
42. Chang, M.-F., Chiu, P.-F. & Sheu, S.-S. Circuit design challenges in embedded memory and resistive ram (rram) for mobile soc and 3d-ic. In *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)* (ed. Chang, M.-F.) 197–203 (IEEE, 2011).
43. Du, G. *et al.* Efficient softmax hardware architecture for deep neural networks. In *Proc. 2019 on Great Lakes Symposium on VLSI*, 75–80 (2019).
44. Khan, M. U. *et al.* Asymmetric GaN/ZnO Engineered Resistive Memory Device for Electronic Synapses. *ACS Appl. Electron. Mater.* (2022).
45. Khan, M. U., Abbas, Y., Rezeq, M., Alazzam, A. & Mohammad, B. Unidirectional neuromorphic resistive memory integrated with piezoelectric nanogenerator for self-power electronics. *Adv. Func. Mater.* **34**, 2305869 (2024).
46. Abbas, Y. et al. Stopping voltage-dependent pcm and rram-based neuromorphic characteristics of germanium telluride. *Adv. Func. Mater.* **34**, 2214615 (2024).
47. Wolf, T. *et al.* Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 38–45 (2020).
48. Halawani, Y. & Mohammad, B. Forsa: Exploiting filter ordering to reduce switching activity for low power cnns. *Authorea Preprints* (2023).
49. Zhang, B. et al. 90% yield production of polymer nano-memristor for in-memory computing. *Nat. Commun.* **12**, 1984 (2021).
50. Liu, J. et al. A bioinspired flexible neuromuscular system based thermal-annealing-free perovskite with passivation. *Nat. Commun.* **13**, 7427 (2022).
51. Poddar, S. et al. Down-scalable and ultra-fast memristors with ultra-high density three-dimensional arrays of perovskite quantum wires. *Nano Lett.* **21**, 5036–5044 (2021).
52. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).

Acknowledgements
This publication is based upon work supported by the System-On-Chip lab at the Khalifa University of Science, Technology under Award No. [RC1-2018-020]. Furthermore, this research work was supported by Abu Dhabi National Oil Company (ADNOC), Emirates NBD, Sharjah Electricity Water & Gas Authority (SEWA), Technology Innovation Institute (TII) and GSK as the sponsors of the 4th Forum for Women in Research (QUWA): Sustaining Women's Empowerment in Research & Innovation at University of Sharjah.

Author contributions
M.B., Y.H. and B.M. devised the main conceptual idea of the RRAM-based MatMUL acceleration and the high-level implementation of the transformer algorithm. NeuroSim simulations were led by M.U.K. with feedback from M.B., Y.H. and B.M. Technical analysis and writing were led by M.B. and achieved collaboratively by all authors. All authors discussed the results and commented on the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to B.M.