Edge-PoolFormer: Modeling and Training of PoolFormer Network on RRAM Crossbar for Edge-AI Applications

Tiancheng Cao, *Student Member, IEEE*, Weihao Yu, Yuan Gao, *Member, IEEE*, Chen Liu, Tantan Zhang, Shuicheng Yan, *Fellow, IEEE*, Wang Ling Goh, *Senior Member, IEEE*

***Abstract*—PoolFormer is a subset of Transformer neural network with a key difference of replacing computationally demanding token mixer with pooling function. In this work, a memristor-based PoolFormer network modeling and training framework for edge-AI applications is presented. The original PoolFormer structure is further optimized for hardware implementation on RRAM crossbar by replacing the normalization operation with scaling. In addition, the non-idealities of RRAM crossbar from device to array level as well as peripheral readout circuits are analyzed. By integrating these factors into one training framework, the overall neural network performance is evaluated holistically and the impact of nonidealities to the network performance can be effectively mitigated. Implemented in Python and PyTorch, a 16-block PoolFormer network is built with 64×64 4-level RRAM crossbar array model extracted from measurement results. The total number of the proposed Edge PoolFormer network parameters is 0.246M, which is at least one order smaller than the conventional CNN implementation. This network achieved inference accuracy of 88.07% for CIFAR-10 image classification tasks with accuracy degradation of 1.5% compared to the ideal software model with FP32 precision weights.**

Index Terms*— *Transformer, PoolFormer, memristor crossbar, Edge-AI, neuromorphic system.

I. Introduction

Transformer has demonstrated remarkable improvement in classification accuracy in the areas of computer vision and natural language processing [1-6]. Transformer's excellent performance is mainly attributed to the self-attention process which costs substantial computation resources. Therefore, the transformer algorithm is normally implemented in a cloud cluster or a powerful dedicated accelerator chip [7–9]. All of these solutions consume considerable amount of power, hence they are not suitable for battery-powered resource constraint edge-AI devices. Recently, a few lightweight transformer models have been reported [10-12]. HAT [10] focused on searching for the best Transformer structure based on hardware resources constraints. LeViT [11] , Vit-Lite, CVT and CCT [12] deployed convolution to extract low-level information and removing class tokens to reduce the model parameters . Furthermore, recent research suggests that the computationally intensive self-attention process can be replaced with pooling operation [13]. Since it requires no parameters in the memory and the processing procedure is much simplified, tremendous amount of computation resources can be saved.

On the other hand, many methods have been investigated to implement attention operation and the associated peripheral operations like active function and logical operations. There are also efforts to build hardware-software co-design framework to simulate and evaluate the performance of Transformer [14-17]. However, these reported solutions still require considerable computation resources. The ADC/DAC consumes around 80% of the total power to realize the essential signal transfer between analog and digital processes. Furthermore, the impact of nonidealities of analog memristor model and crossbar array on Transformer network are not examined. To include the non-idealities into MLP and CNN training and inference procedures, a variety of approaches have been investigated and controlling the model size still holds immense significance within edge device applications [18-26]. Thus, there is an urgent requirement for a modeling and simulation tool centered on memristor crossbars, particularly tailored for assessing the efficiency of lightweight transformer neural networks [27-28].

This work presents a memristor-based PoolFormer network modeling and training framework for edge-AI applications. The major contributions of this work are listed as follows,

1) Firstly, a unified hardware-friendly PoolFormer neural network, Edge-PoolFormer is introduced. This network uses

Manuscript received MMM-DD, YYYY, revised MMM-DD, YYYY; accepted MMM-DD, YYYY. This work was supported by Agency for Science, Technology and Research (A*STAR), Singapore under the Nanosystems at the Edge programme, Grant No. A18A1b0055 and High Linearity Silicon Germanium Photonic Modulator for 6G Analog Radio over Fiber Project, Grant No. M24M8b0004. (Corresponding author: Yuan Gao)

T. Cao is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 and he is also with Institute of Microelectronics (IME), A*STAR, Singapore 138634. (e-mail: CAOT0006@e.ntu.edu.sg).

W. Yu is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077 (e-mail: weihaoyu6@gmail.com).

Y. Gao, C. Liu and T. Zhang are with the Institute of Microelectronics (IME), A*STAR, Singapore 138634. (e-mail: liu_chen@ime.a-star.edu.sg, gaoy@ime.a-star.edu.sg).

S. Yan is with the Skywork AI, Kunlun 2050 Research, Singapore. (e-mail: shuicheng.yan@gmail.com).

W.L. Goh is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798. (e-mail: ewlgoh@ntu.edu.sg).

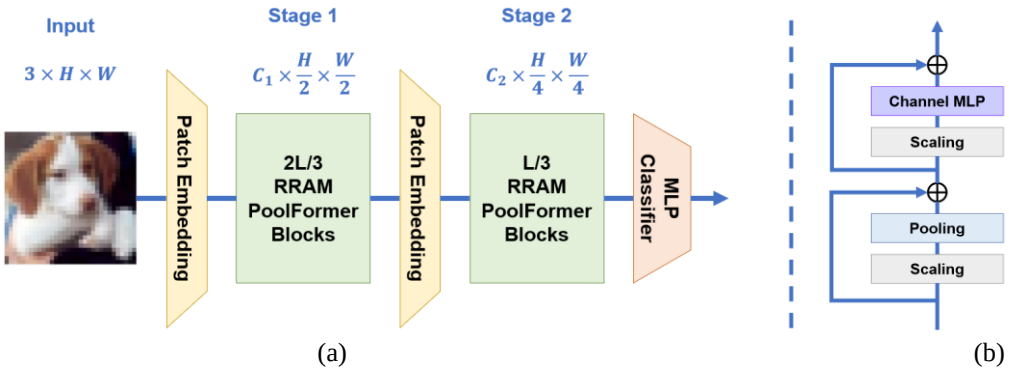

Fig. 1. (a) The overall framework of Edge PoolFormer that adopts hierarchical architecture with 2 stages. For a model with L Edge PoolFormer blocks, stage [1, 2] have [2L/3, L/3] blocks, respectively. The embedding dimension stage *i* is C_i shown in the figure. (b) The Edge PoolFormer block architecture in software training is presented, which differs from the Transformer block by using a simple non-parametric operator, pooling, instead of attention as basic token mixer.

basic operators such as scaling and pooling to serve as a highly efficient biasing and token mixer.

2) Secondly, by capitalizing on the processing-in-memory feature of memristors, we integrate weight storage and various layer functions into the crossbar array. This integration facilitates computation within the array, eliminating the separation between weight representation and computation.

3) Lastly, we meticulously account for memristor non-idealities across devices, arrays, and readout circuits during the network training process. This compensation strategy effectively enhances the Edge PoolFormer design and optimization. The performance of the proposed system is evaluated with impacts of nonidealities and achieved satisfactory results compared to conventional CNN based edge device models.

To assess the performance of the suggested framework, a 16-block Edge PoolFormer network is constructed on a low resolution 4-level 64×64 RRAM array. On the CIFAR-10 dataset [29], the results show an inference accuracy of 88.07%.

The rest of this paper is organized as follows. Section II introduce the overall Edge-PoolFormer framework. Section III presents the implementation of neural network layers on RRAM crossbar array and the impact of non-idealities. Section IV presents the edge-poolformer simulation results. Lastly, Section V conclude the work.

II. Edge-PoolFormer

The proposed Edge-PoolFormer structure is shown in Fig. 1. Compared to the standard transformer structure, the attention operation is replaced by the non-parametric average pooling operation. Since it requires no parameters in the memory and the process is simple, huge computation resources and storage are saved and simple analog process is applicable, which fits the demand of edge devices. Additionally, compared to the conventional PoolFormer, the proposed structure has following novelties to adapt to edge device applications.

A. Replacement of Layer Normalization with Scaling

Layer normalization plays an important role in the original PoolFormer. However, the computation of the average value and the variance in analog manner is not trivial. Although

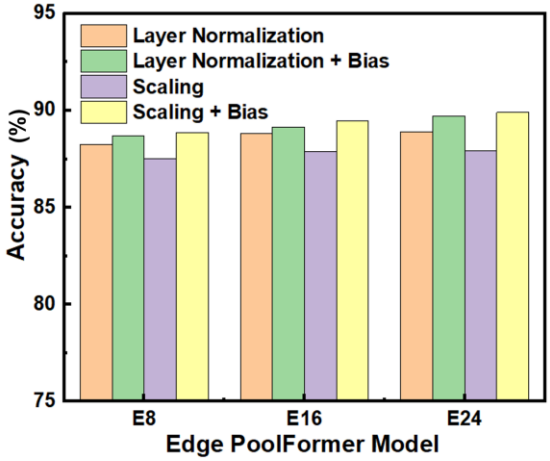

Fig. 2. The comparison of Edge-PoolFormer model performance with conventional layer normalization and proposed channel scaling layer on CIFAR-10.

previous work has proposed circuit solution for normalization process [17], this solution includes subtraction, summation and square root functions, which consume large static power and area. In view of the aforementioned considerations, a re-evaluation of layer normalization is necessary. In the context of small-scale models for edge applications, the integration of shortcut paths has already proven to be satisfactory in fulfilling the prerequisites for achieving profound model convergence. To improve the generalization ability of the model, data standardization is necessary. However, most edge computing applications have small sample sizes and the same feature shapes, where batch normalization is more suitable for training and the process is shown below.

$$\mu_c = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x \tag{1}$$

$$\sigma_c = \sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(x-\mu_c)^2 + \epsilon} \tag{2}$$

$$x_{BN} = \gamma\left(\frac{x-\mu_c}{\sigma_c}\right) + \beta$$

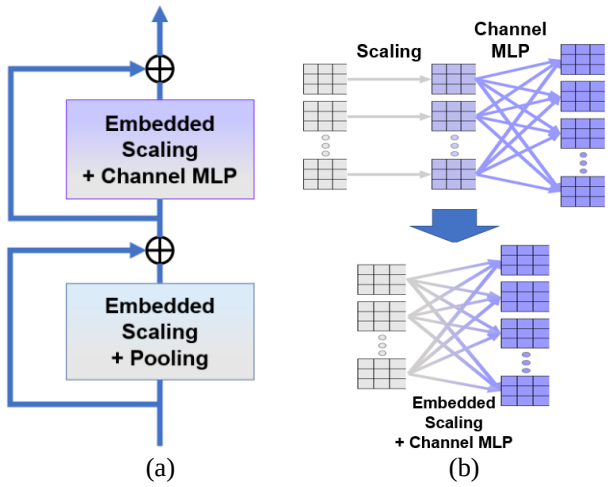

Fig. 3. (a) The architecture of Edge-PoolFormer block in implementation framework. (b) The diagram of the change of scaling and channel MLP operations from software training to implementation framework.

$$= \frac{\gamma}{\sigma_c}x + \left(\beta - \frac{\gamma\mu_c}{\sigma_c}\right) = scale \cdot x + shift \quad (3)$$

where x is the input sample of size (n, c, h, w), μ_c is the mean value of the channel, σ_c is the standard deviation of the channel, γ and β are the learnable parameters and x_{BN} is the batch normalization output.

It is observed that the output of batch normalization only comprises scale and shift operations. The inherent bias of the MLP within each block effectively serves as the shift operation, rendering the addition of supplementary scaling to each channel a logical choice. Although these additional scaling constants necessitate initialization and learning, potentially increasing model instability, their practical impact remains negligible for small-scale models. Moreover, these learnable scaling parameters also enhance the model generalization ability and overall accuracy. An evaluation of the effects of channel scaling and layer normalization on 200-epoch Edge-PoolFormer models is presented in Fig. 2. The results indicate that the channel scaling layer, in conjunction with bias, provides a complete equivalent to conventional layer normalization within this model.

1) *2-stage structure and smaller embedding dimension*

The conventional Vision Transformer setup typically involves four stages using block distributions of $L/6$, $L/6$, $L/2$ and $L/6$ for data processing. For edge applications with small neural network, the proposed Edge-PoolFormer adopts a 2-stage structure employing $2L/3$ and $L/3$ block distributions. Moreover, to effectively reduce parameters, embedding dimensions of 32 and 64 are chosen for each stage. With these improvements, there is a noticeable reduction in the model parameters and Multiplication and Accumulation (MAC) operations.

2) *Scaling is embedded in implementation*

During software training, scaling typically functions as a standalone layer. However, in our proposed system, during crossbar array implementation, scaling is strategically integrated with other operations like pooling and channel MLP within the training process. This consolidation optimizes processing efficiency, as depicted in the implementation framework sketch of the Edge PoolFormer block in Fig. 3(a). An illustrative example showcasing scaling embedded within the channel MLP is detailed in Fig. 3(b).

This cohesive integration of operations significantly economizes the space between layers, effectively reducing system latency and power consumption. Such optimization aligns perfectly with the requirements of edge devices, where resource efficiency is paramount. This approach ensures that the system maximizes its performance within the limitations inherent to edge environments.

Table I. Edge-PoolFormer hyper-parameters

Stage	# Token	Layer Specification		Edge PoolFormer		
				E8	E16	E24
1	$\frac{H}{2}\times\frac{W}{2}$	Patch Embedding	Patch Size	3×3, stride 2		
			Embed. Dim.	32		
		Edge PoolFormer Block	Pooling Size	3×3, stride 1		
			MLP Ratio	4		
			# Block	6	12	18
2	$\frac{H}{4}\times\frac{W}{4}$	Patch Embedding	Patch Size	3×3, stride 2		
			Embed. Dim.	64		
		Edge PoolFormer Block	Pooling Size	3×3, stride 1		
			MLP Ratio	4		
			# Block	2	4	6
Parameters (M)				0.13	0.246	0.358
MACs (M)				1.75	2.1	2.44

B. *Performance Analysis*

In this work, three models with 8, 16 and 24 Edge PoolFormer blocks are evaluated on CIFAR-10 dataset. Their hyper-parameters are shown in Table. I. Each model has 2 stages with $\frac{H}{2}\times\frac{W}{2}$ and $\frac{H}{4}\times\frac{W}{4}$ tokens, respectively. The models underwent 300 training cycles using stochastic gradient descent (SGD) optimization method with batch size of 125, weight decay $1e^{-4}$, peak learning rate $1e^{-3}$ and momentum 0.8. The dropout is set as 0.5 and the droppath is set to 0.1 to help train the models. Our implementation is built on the PoolFormer codebase [13], and the experiments are conducted using GPUs. Table. II shows the performance of the Edge-PoolFormer on CIFAR-10 classification. Edge-PoolFormer-E8 reaches the top-1 accuracy of more than 89.13%, requiring only 0.13M parameters and 1.7M MACs. In comparison, the widely used ResNet18 [30] achieves a slightly higher accuracy of 90.27%. However, it requires 86 times more parameters and 24 times more multiply-accumulate operations. To obtain similar accuracy, a small-scale Transformer model ViT-Lite7/8 [31] needs 29 times parameters as well as 35 times computation while only 89.1% accuracy is attained. When compared with the advanced transformer variants [31], the Edge PoolFormer still shows a better balance between the performance and

Table. II. Comparison of network top-1 validation accuracy on CIFAR-10 dataset

Architecture	Model	Image Size	Parameters (M)	MACs (M)	Epoch	Top-1%
Convolutional Network	ResNet18	32	11.18	40	300	90.27
	ResNet34	32	21.29	80	300	90.51
	MobileNetV2/0.5	32	0.7	10	300	84.78
	MobileNetV2/1.0	32	2.24	10	300	89.07
Vision Transformer	ViT-12/16	32	85.63	430	300	83.04
	ViT-Lite7/16	32	3.89	20	300	78.45
	ViT-Lite6/16	32	3.36	20	300	78.12
	ViT-Lite7/8	32	3.74	60	300	89.1
	ViT-Lite7/8	32	3.22	60	300	88.29
Compact Transformer	CVT-7/8	32	3.74	60	300	89.79
	CVT-6/8	32	3.21	50	300	89.5
	CCT-2/3	32	0.28	40	300	89.75
PoolFormer	Edge PoolFormer-E8	32	**0.13**	**1.7**	300	**89.13**
	Edge PoolFormer-E16	32	**0.24**	**2.1**	300	**89.52**
	Edge PoolFormer-E24	32	**0.35**	**2.4**	300	**90.28**

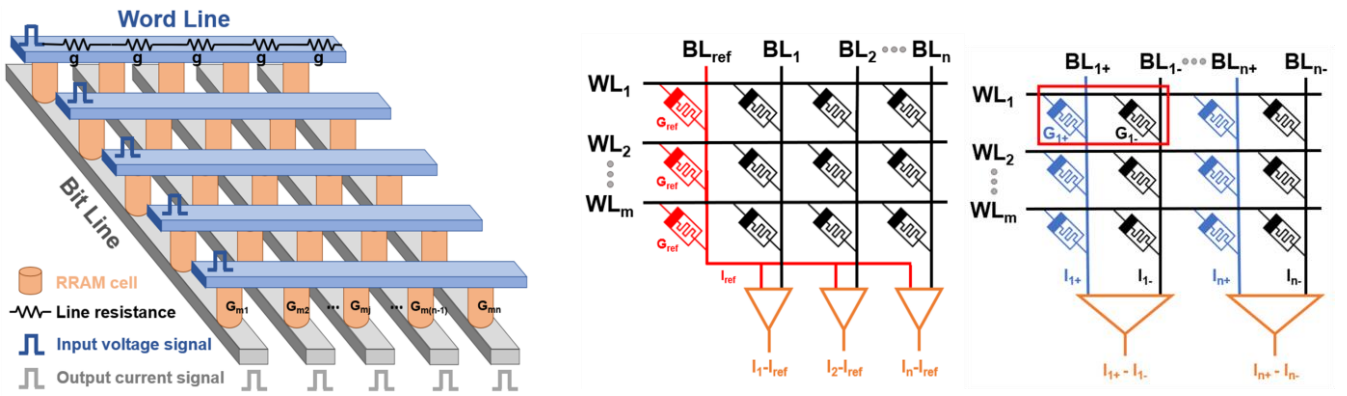

Fig. 4. (a) Block diagram of a $m \times n$ crossbar array with horizontal word lines and vertical bit lines, with interconnect line resistances. Line resistance refers to the ohm resistance of the interconnect traces along the top and bottom electrodes of the memristor devices. Structures of (b) one-cell-one-weight scheme with reference current and (c) two-cell-one-weight scheme with differential pairs.

resource requirements. Specifically, CCT-2/3 obtains 89.75% top-1 accuracy with 0.28M parameters and 40M MACs while the Edge PoolFormer-E16 reaches 89.52% with 15% fewer parameters (0.24M) and 95% fewer MACs (2.1M).

III. Memristor Crossbar Array

A. VMM mapping on crossbar array

A typical memristor crossbar array is shown in Fig. 4(a). It comprises of memristor devices at each cross point along with a group of horizontal word lines (WL) and vertical bit lines (BL). Line resistance refers to the resistance in ohms of the connection traces that run between the top and bottom electrodes of the memristor devices. Using a deep submicron CMOS technology, the unit line conductance g between two neighboring memristor devices is in the region of a few Ohms [32]. When a voltage vector, V_{IN}, is applied to the WL, ignoring the influence of line resistance initially, the output current, I_{Oj}, of the j^{th} column can be written as follows

$$I_{Oj} = \sum_{i=1}^{m} V_{INi} \cdot G_{ij} \quad (4)$$

where $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n$. V_{INi}, represents the input voltage of i^{th} row, and G_{ij} represents the conductance of memristor device at i^{th} row and j^{th} column.

For crossbar-based computing, the mapping of negative weights is an unavoidable problem. The approach of having one weight per cell with a reference current shown in Fig. 4(b) has been reported in several earlier works [33]. Large-scale arrays are currently not possible because of the fluctuation of devices. Thus, the two-cell-one-weight technique shown in Fig. 4(c) is applied [34]. One weight is used demonstrated by every two

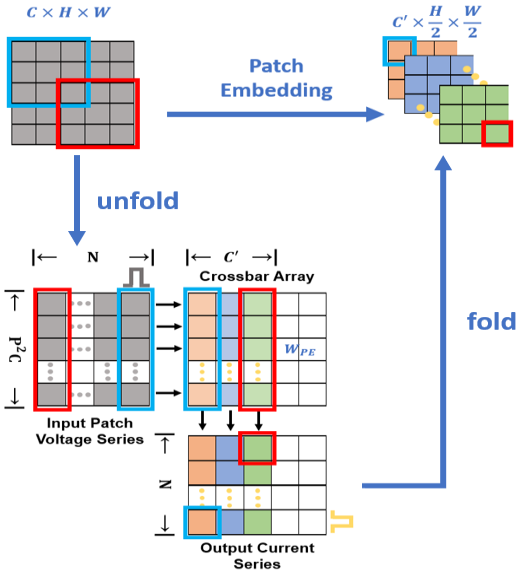

Fig. 5. Process to implement Patch Embedding on crossbar array.

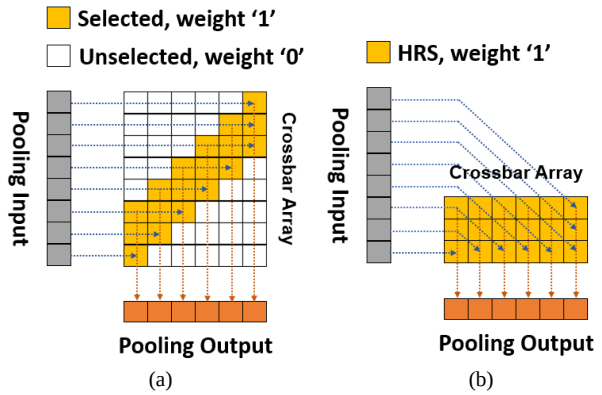

Fig. 6. Crossbar array implementation of Pooling.

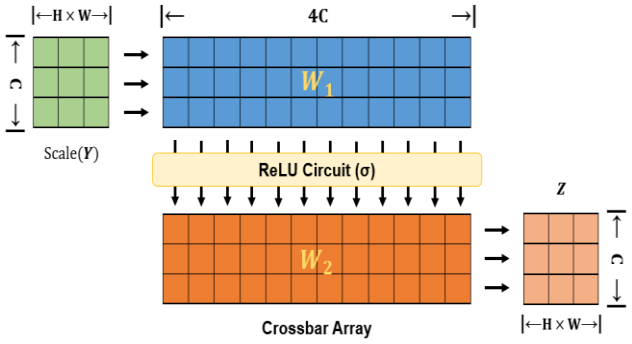

Fig. 7. Crossbar array implementation of Channel MLP. The input data flows through scaling, matrix multiplications with learnable weights, activation functions, and the process of extracting output data.

neighboring cells. Positive weight is represented by cells in the left column, while negative weight is indicated by cells in the right column. The difference in bit line current between two columns is detected to determine the real output current I_O.

$$I_O = {I_O}^+ - {I_O}^- = V_{IN} \times (G^+ - G^-) \quad (5)$$

B. Layer Implementation

1) Implementation of Patch Embedding

Like ViT, Patch Embedding (PE) is used to downscale the sample, to extract the features before each stage.

$$X = PE(I) = fold(unfold(I) \times W_{PE}) \quad (6)$$

where $I \epsilon (C, H, W)$ and $X \epsilon (C', \frac{H}{2}, \frac{W}{2})$, W_{PE} is the learnable weight, P is the path size, C is the input channel, N is the path number and C' is the embedding dimension. The patch size is set to 3 and the stride is fixed at 2 in this work. To implement the embedding process, the input is firstly unfolded to N patches with size (N, P^2C). Thereafter, each patch is converted to the input voltage of the crossbar array and fed into the word lines. After collecting the VMM results with size (N, C'), the output of embedding is folded to the feature maps as shown in Fig. 5.

2) The implementation of Pooling

An average Pooling takes the mean value of the kernel after the layer scaling, and can be represented as

$$Y = avgPool(Scale(X)) \quad (7)$$

where $avgPool(\cdot)$ stands for average pooling. The average pooling is the sum of the multiplication result of the elements and the matrix of ones, and then divided by the number of the elements. Since the stride of the average pooling is 1, the high reuse rate of elements makes it possible to process in parallel with crossbar array as shown in Fig. 6(a). To save the wasted area of unused devices, the array is reconstructed to Fig. 6(b) where HRS devices are engaged to reduce the static power consumption.

3) The implementation of Channel MLP

The channel MPL block consists of 2-layer MLP with a scaling and non-linear active function. The detailed structure is shown in Fig. 7 and the formula is

$$Z = \sigma(Scale(Y) \times W_1) \times W_2 \quad (8)$$

where $W_1 \in R^{C \times 4C}$ and $W_2 \in R^{4C \times C}$ are learnable parameters in MLP. $\sigma(\cdot)$ denotes the non-linear active function and ReLU is used in this work.

The weights of MLP layer are mapped on crossbar array directly. When the input voltage is applied to the word lines, the result of vector matrix multiplication is collected as output current from the bit lines. Besides, the ReLU circuit in II.B. helps to transfer the bit line current to the input voltage of the second layer MLP at the same time.

C. Crossbar Array Nonidealities

1) Device Non-idealities

Due to manufacturing imperfections, non-ideal memristor device characteristics such nonlinearity, conductance variation and programming failure are inevitable, and they can bring

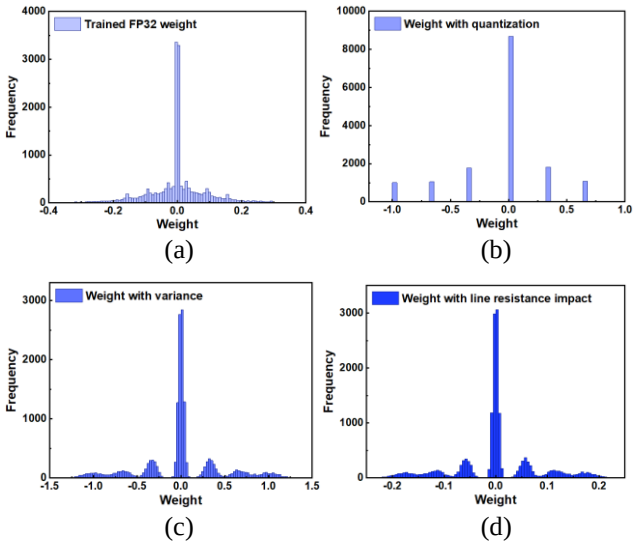

Fig. 8. An example of weights mapping with trained weight from channel MLP in stage-2 block-4. The figures show: (a) the trained FP32 weight, (b) the quantized weight, (c) the weight mapping with device nonidealities and (d) the weight mapping with crossbar nonidealities.

significant impact to the network performance [35]. Therefore, the non-idealities should be considered during the network design stage and compensated appropriately.

The proposed solution divides the devices into levels according to the resolution in the crossbar array as shown in Fig. 8. First, the ideal weights, w_a, are converted to an absolute value and the signs are recorded. Second, the largest *t%* tail weights are assigned to G_{LRS} directly and left maximum weight is marked as w_{max-t}. According to the following equation, the left weights are transferred to the conductance weights, w_c:

$$w_c = w_a \frac{G_{LRS}-G_{HRS}}{w_{max-t}} + G_{HRS} \quad (9)$$

The conductance range is then partitioned based on the intermediate value between each pair of neighboring levels. Each RRAM level is given a set of conductance weights based on the regions. The recorded signs are then combined, and variations for each weight are generated based on the levels, with the addition of a normal distribution to reflect the non-ideal behaviors at each level [36].

Each training cycle generates a random mask matrix that represents programming failure based on the measured programming failure rate in order to simulate the effects of writing failure. To obtain the programmed conductance value, the array conductance matrix and this programming failure mask are merged using the AND operation. It should be noticed that the failing devices have HRS status assigned to them.

2) *Crossbar Non-idealities*

Despite the mentioned error from RRAM unite cells, crossbar array structure also introduces non-idealities which are primarily caused by sneak path and line resistance [37]. The node voltage between the theoretical and real cases will change due to sneak paths and cumulative IR-drop on the connection

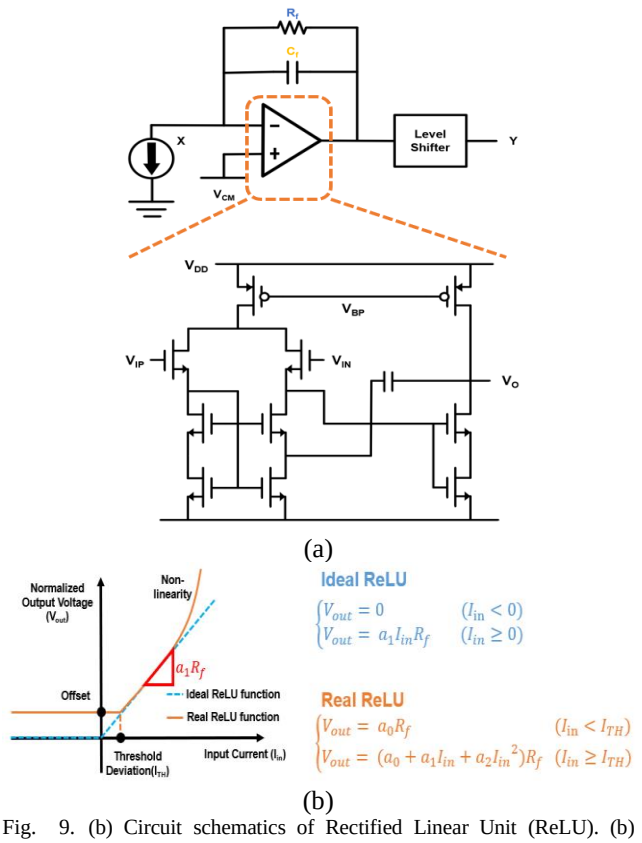

Fig. 9. (b) Circuit schematics of Rectified Linear Unit (ReLU). (b) Comparison of ideal model and actual model of ReLU circuit including non-ideal factors.

traces. As shown in Fig. 8, conductance overlapping and line resistance work together to obfuscate real conductance levels and decrease accuracy.

To estimate the IR-drop in the crossbar, iterative and linear approaches have been applied in various studies of a similar research [38-39]. The typical approach is to compute the crossbar voltage distribution using Kirchhoff's laws equations. The temporal complexity can be lowered to $O(m^2n^2)$ using iterative matrix techniques. Even so, it continues to take a long time, particularly as the crossbar size increases. In order to speed up simulation, a novel method was suggested to manage voltage and current degradations as scaling factor matrices that are applied to the output current and input voltage. This model achieves reduced current distortion while combining rapid computation with time complexity $O(mn)$. [40] provides more technical explanations.

3) *Peripheral Circuitry*

The crossbar array output is read out using ADC. In the digital domain, nonlinear activation functions like ReLU and Sigmoid are implemented. However, using an ADC will consume up power and circuit area, both of which are significant issues [34]. A transimpedance amplifier (TIA) is illustrated in Fig. 9(a) to generate partial sum of current for the subarrays, the current to voltage conversion, and ReLU activation function with the aim of enhancing the system efficiency. The ideal signal is expressed as

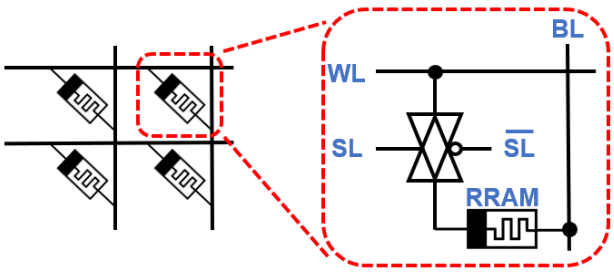

Fig. 10. The circuit model diagram of the 2T1R unit cell with Ta_2O_5 RRAM.

Table. III. Crossbar parameter summary

Nonidealities	Value
Number of levels	4
Average R (kΩ)	5 / 6.9 / 11.3 / 27.9
Device variation (σ/μ)	10%
Array size	64×64
Line resistance(Ω)	0.5
Output offset	5μA
Threshold deviation	5μA
Nonlinearity	1%

$$\begin{cases} V_{out} = 0 & (I_{in} < 0) \\ V_{out} = a_1 I_{in} R_f & (I_{in} \geq 0) \end{cases} \quad (10)$$

The TIA non-idealities, such as the non-linearity, threshold deviation, and input offset, as shown in Fig. 9(b), would result in computation error, lowering accuracy even further.

The ReLU function is modeled as a polynomial formula with second order effect to replicate the effects of the TIA non-idealities:

$$\begin{cases} V_{out} = a_0 R_f & (I_{in} < I_{TH}) \\ V_{out} = (a_0 + a_1 I_{in} + a_2 {I_{in}}^2) R_f & (I_{in} \geq I_{TH}) \end{cases} \quad (11)$$

where I_{in} represents the input current to the function, and V_{out} represents the output voltage, respectively. I_{TH} represents the deviation in the threshold, a_0 represents the output offset, a_1 represents the linear coefficient, which is set to 1 in this study and, a_2 is the nonlinear coefficient.

IV. Implementation and Results

To demonstrate the proposed software-hardware co-design architecture, Edge-PoolFormer-E16 model is developed for image classification task using CIFAR-10 dataset. The model consists of two stages, with embedding dimension 32 and 64, respectively. The first stage has 12 blocks and the second stage has 4 blocks followed by a MLP classifier. The memristor model is based on the TaOx 2T1R RRAM crossbar chip measurement results [41, 42]. The RRAM device structure and layer of materials are integrated in a 0.18-μm complementary

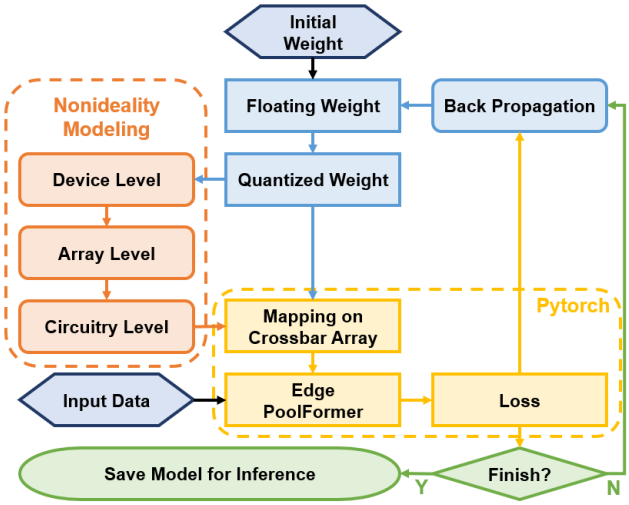

Fig. 11. The flowchart of the aware training. This approach aims to improve the model resilience and efficiency within real-world hardware limitations.

metal-oxide-semiconductor (CMOS) baseline wafer through a backend-of-line (BEOL) integration process. The circuit model of a 4-level RRAM device is used in the project as shown in Fig. 10. Table. III provides a summary of all the non-idealities. The non-idealities aware training [42] is carried out using these parameters, and the inference is assessed using trained weights that are mapped onto the devices. The weights are directly programmed into the devices and are updated by the software. The flowchart of the aware training used in this work is shown in Fig. 11. In the PyTorch platform, the fundamental training modules are implemented. During the forward pass, weights undergo quantization to specific levels corresponding to RRAM cells. Our modeling accounts for non-idealities during array mapping. In the backward pass, the floating-point weights are updated using local Python functions, leveraging training loss from PyTorch for this purpose.

A. *Weight Quantization*

Model quantization is the first step to map the trained weights on crossbar array. As a result, accuracy degradation is inevitable due to the limited resolution. The impact of the device resolution on the training accuracy is shown in Fig. 12. The accuracy loss for the conventional offline training is substantially more than that for the nonidealities aware training used on the Edge PoolFormer-E16 after mapping the weights to 2–8 bits resolution where <0.5% loss observed. Also, the normal offline training is more dependent on the device resolution than the nonidealities aware training in Edge PoolFormer. The device resolution is set to 2 bits in training during the following parts.

B. *Non-ideality Aware Training*

With a 6% degradation as the variation up to 20%, the influence of device variation as shown in Fig. 13(a), which shows a remarkable preservation of accuracy with the proposed method. Fig. 13(b) illustrates the effect of the program failure probability. Even though the non-idealities aware training may experience a 10% drop in accuracy compared to the software

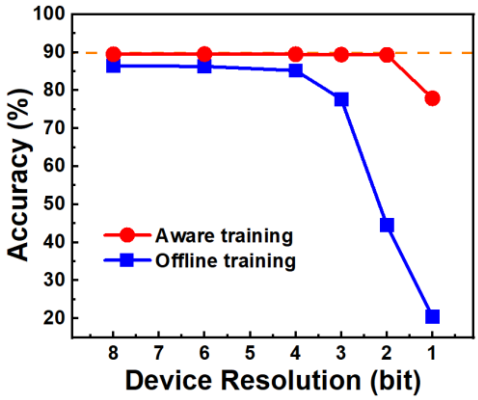

Fig. 12. The impact of the device resolution. The software accuracy is shown as dash line for reference. It shows the benefits of non-idealities aware training in mitigating accuracy loss due to limited resolution compared to conventional offline training methods.

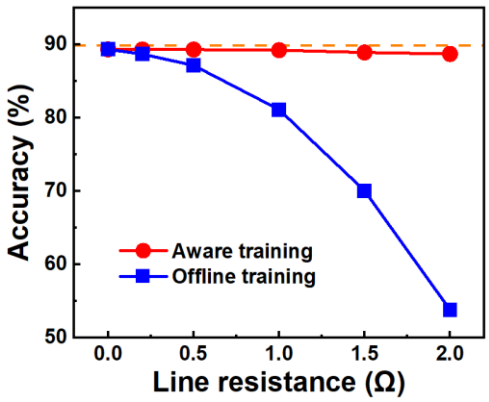

Fig. 14. The impact of the crossbar array line resistance. The software accuracy is shown as dash line for reference. The ability of the proposed aware training to maintain accuracy levels even under significantly increased line resistance conditions

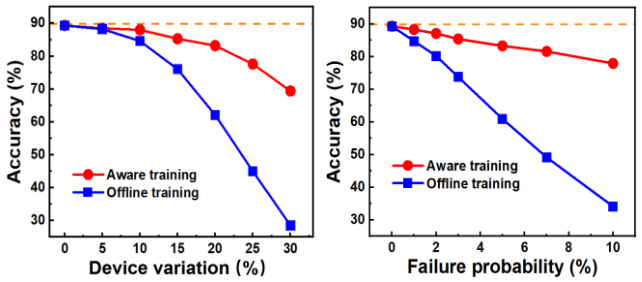

Fig 13. A comparison of the recognition accuracy between the non-idealities aware training method and the conventional offline training method in Edge PoolFormer, for (a) different device to device variation (defined as σ/μ) and (b) different program failure probability.

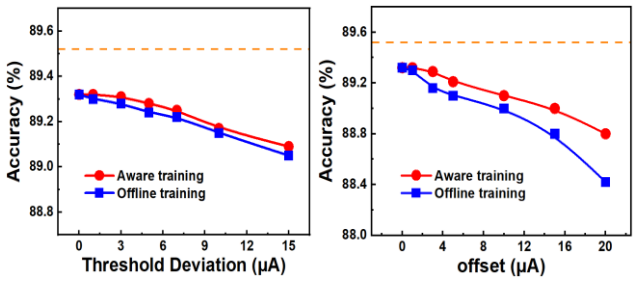

Fig 15. This diagram compares the results of the non-idealities aware training and the conventional offline training in Edge Poolformer, with (a) an increase in the output offsets of the ReLU circuits and (b) an increase in the deviation of the input threshold in the ReLU circuit. The non-linear component represents 1% of the total output.

result with a 10% programming failure, it is still more reliable than the conventional offline training method, which has a higher probability of programming failure.

The line resistance between adjacent cells will lower the voltage provided to the memristor cell. It is based on the size of the array, the resistance of the lines, and the average conductance. It is dependent on the average conductance, the line resistance and the array size. Referring to the distribution of the mapping weights, Fig. 8, the high weight states in the distribution of the mapping weights will be moved toward the center and overlapped depending on the cell location in the array, which will affect accuracy. The effect of the suggested training framework against the line resistance is shown in Fig. 14 along with the loss of accuracy against different line resistances. When line resistance is severely increased to 2Ω, the framework can still maintain accuracy compared to the conventional offline training on 64×64 array.

Section III.D introduces the TIA non-idealities, such as the non-linearity, the threshold deviation, and the output offset. Fig. 15 illustrates the robustness of the aware training against these non-idealities in comparison to the conventional offline training. The results reveal that Edge PoolFormer framework is not sensitive to the impact of designed ReLU circuit, which relaxes the standard requirements for the ReLU hardware implementation.

C. *Interaction Inference*

All of these non-idealities interact in the real crossbar, which magnifies the accuracy loss caused by each one individually. The impact of all non-idealities on Edge PoolFormer framework is examined simultaneously in the inference system. The training is carried out using the parameters from Table.III, and the trained weights used in the inference system have been loaded onto the hardware. The weight distribution following the training with non-idealities is shown in Fig. 16. The distribution clearly aggregates to four separate levels with a noticeable overlap due to device variance when compared to the analog weight by software training. The training loss and accuracy of the proposed Edge PoolFormer-E16 framework on CIFAR-10

Table. IV. RRAM-based frameworks on CIFAR-10 comparisons

	[43]	[44]	[45]	[46]	[47]	[48]	This work
Model	VGG-11	ResNet-18	VGG	VGG-8	ResNet-20	ResNet-20	PoolFormer
Device	HfO2	NR	HfO2/Al2O3	TaOx/HfOx	NR	NR	TaOx
# of Conductance Levels	NR	256	16	128	32	10	4
# cells per weight	2	2	2	2	2	2	2
Ron(Ω)	11k	3k	50k	100k	3k	25k	5k
On/Off Ratio	1.9	1000	16	10	1000	12	6
D2D Variation (σ/μ)	NR	20%	4%	3.70%	0.9%	NR	10%
# of parameters	7.66M	NR	6.6M	13M	0.27M	0.27M	0.24M
Line resistance (Ω)	NR	NR	0.1	NR	NR	NR	0.5
Crossbar Array	256×256	64×64	1152×128	128×128	32×32	NR	64×64
Failure	10%	NR	NR	NR	NR	NR	1%
Nonidealities Aware	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Software Accuracy (%)	85.2	87	85.17	NR	88.27	84.94	89.52
Inference Accuracy (%)	83	86.3	79.4	81	85.58	84.11	88.07

*NR：Not Reported

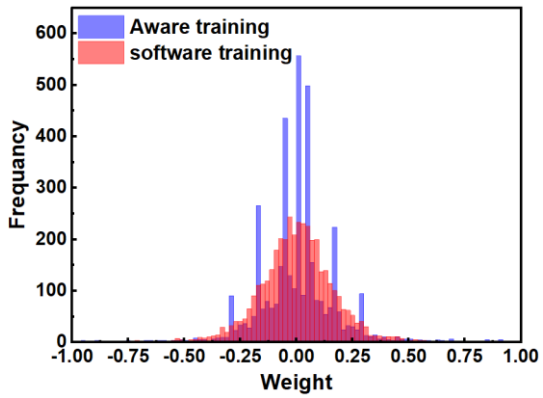

Fig. 16. A comparison of the weight distribution between the ideal software weights trained through normal training and the trained weights in the inference system after undergoing non-idealities aware training. Weights are extracted from the MLP classifier.

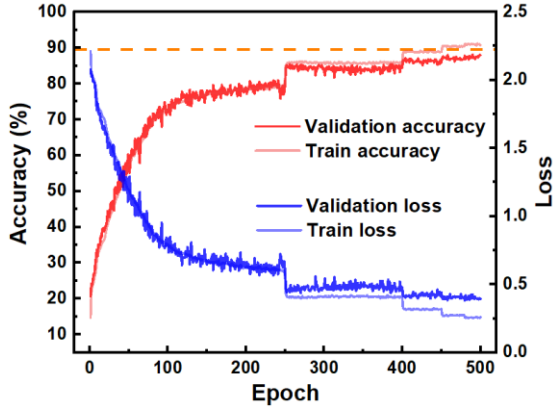

Fig. 17. Validation inference accuracy with aware training under incorporated crossbar array nonidealities.

is given in Fig. 17, where the validation accuracy has more oscillation and is close to the software accuracy after 500 epochs. The contrast between the framework with aware training and the conventional offline training is shown in Fig. 18. The inference system with typically trained weights has lost the identification ability because of all these non-idealities and will mistake the majority of figures for deer and bird. On the other hand, the inference with non-idealities aware training can classify most labels correctly, with an 88.07% overall accuracy for 64×64 array, 0.5 Ω line resistance and 10% device variation, around 1.5% loss compared to the software.

The results also reveal that the Edge PoolFormer framework implemented on memristor-based PIM system is a potential outstanding software and hardware co-design solution for edge AI applications on small dataset. In Table.IV, the performance of the suggested framework is outlined and contrasted with that of other previously published DNNs and Transformer that was implemented with a memristor crossbar. The results of the image classification task using the CIFAR-10 dataset are evaluated for a fair comparison. The highlighted framework showcases a notable achievement by leveraging a considerably smaller model and utilizing only a 4-level device, ultimately leading to enhanced inference accuracy. Notably, PoolFormer not only rivals the software and inference accuracies of established architectures like VGG and ResNet but also excels in adapting to diverse hardware environments. Its efficiency in utilizing conductance levels and parameters strikes a delicate balance between model intricacy and effectiveness, while demonstrating a keen awareness of hardware-related non-idealities. The documented minimal failure rate and the successful implementation framework underscore Edge PoolFormer practical viability, solidifying its position as a resilient and versatile model tailored specifically for the intersection of software efficacy and hardware-conscious design, particularly in the realm of resource constrained edge-AI applications.

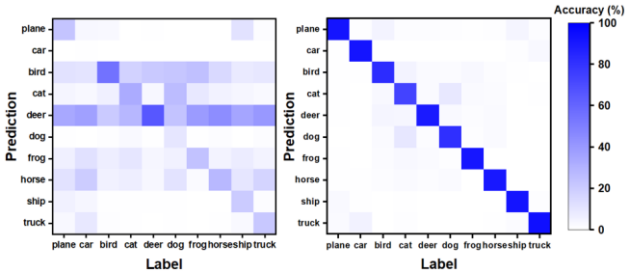

Fig 18. The confusion matrix of the classification correlation for the Edge PoolFormer inference system is shown in the form of a heat map. (a) The result from normal offline training and (b) The outcome of training that takes into account non-idealities. The recognition accuracy of the inference system with normal training is approximately 28.21%, while with training that accounts for non-idealities, it is 88.07%.

V. Conclusion

In this paper, a new light hardware-friendly Transformer like neural network structure, Edge PoolFormer, is presented. A modeling and training framework is introduced for edge device implemented on memristive crossbar is proposed. The structure and performance of the neural network are analyzed. Integrating the nonideal factors into a unified training process allows for comprehensive evaluation of framework performance. In addition, compared with conventional DNNs, this structure can realize satisfactory accuracy with fewer parameters, and the training framework can effectively alleviate the effects of these non-idealities to minimize degradation of the inference accuracy. Designed using a combination of Python and PyTorch, the suggested framework is tested using a 16-block network built on a measured 64×64 RRAM array with a 4-level weight resolution. 88.07% inference accuracy is achieved on CIFAR-10 image classification tasks with less than 1.5% accuracy degradation compared to the ideal model.

References

[1] A. Vaswani, et al.,"Attention is all you need," *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[2] N. Carion and F. Massa, et al., "End-to-end object detection with transformers," *European conference on computer vision (ECCV)*, 2020, pp. 213-229.

[3] A. Dosovitskiy and L. Beyer, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *International Conference on Learning Representations (ICLR)*, 2020.

[4] Z. Liu and Y. Lin, et al., "Swin transformer: Hierarchical vision transformer using shifted windows," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 10012-10022.

[5] L. Yuan and Y. Chen, et al., "Tokens-to-token vit: Training vision transformers from scratch on imagenet," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 558-567.

[6] P. Ramachandran and N. Parmar, et al., "Stand-alone self-attention in vision models," *Advances in Neural Information Processing Systems (NIPS)*, vol. 32, 2019.

[7] Y. -H. Chen, T. Krishna, J. S. Emer and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, Jan. 2017.

[8] N.P. Jouppi, *et al.*, "In-datacenter performance analysis of a tensor processing unit," *in Proc. 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, pp. 1-12, Jun. 2017.

[9] J. Lee, *et al.*, "UNPU: an energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits, vol.* 54, no. 1, pp: 173-185, Jan. 2019.

[10] H. Wang and Z. Wu, et al., "HAT: Hardware-aware transformers for efficient natural language processing," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 7675-7688.

[11] B. Graham and A. El-Nouby, et al., "Levit: a vision transformer in convnet's clothing for faster inference," in *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2021, pp. 12259-12269.

[12] A. Hassani and S. Walton, et al., "Escaping the big data paradigm with compact transformers," arXiv preprint arXiv:2104.05704, 2021.

[13] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "Metaformer is actually what you need for vision," In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10819-10829.

[14] X. Yang, B. Yan, H. Li and Y. Chen, "ReTransformer: ReRAM-based processing-in-memory architecture for transformer acceleration," *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1-9.

[15] M. Kang, H. Shin and L. -S. Kim, "A framework for accelerating transformer-based language model on ReRAM-based architecture," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* , vol. 41, no. 9, pp. 3026-3039, Sept. 2022

[16] H. Sun, Z. Zhu, Y. Cai, X. Chen, Y. Wang and H. Yang, "bit-efficient quantized and regularized training framework for processing-in-memory accelerators," *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 325-330

[17] C. Yang, X. Wang and Z. Zeng, "Full-circuit implementation of transformer network based on memristor," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 69, no. 4, pp. 1395-1407, April 2022.

[18] S. Jain, A. Sengupta, K. Roy and A. Raghunathan, "RxNN: A framework for evaluating deep neural networks on resistive crossbars," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 40, no. 2, pp. 326-338, Feb. 2021.

[19] J. Woo and S. Yu, "Impact of selector devices in analog RRAM-based crossbar arrays for inference and training of neuromorphic system," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 27, no. 9, pp. 2205-2212, Sept. 2019.

[20] T. Cao, Z. Zhang, W. L. Goh, C. Liu, Y. Zhu and Y. Gao, "A ternary weight mapping and charge-mode readout scheme for energy efficient FeRAM crossbar compute-in-memory system," *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, Hangzhou, China, 2023, pp. 1-5, doi: 10.1109/AICAS57966.2023.10168639.

[21] K. Bai, Q. An, L. Liu and Y. Yi, "A Training-Efficient Hybrid-Structured Deep Neural Network With Reconfigurable Memristive Synapses," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 28, no. 1, pp. 62-75, Jan. 2020, doi: 10.1109/TVLSI.2019.2942267.

[22] S. Jain and A. Raghunathan, "CxDNN: Hardware-software compensation methods for deep neural networks on resistive crossbar systems," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 6, pp. 1–23, Jan. 2020.

[23] S. Roy, S. Sridharan, S. Jain and A. Raghunathan, "TxSim: Modeling training of deep neural networks on resistive crossbar systems," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 29, no. 4, pp. 730-738, April 2021.

[24] A. Lu, X. Peng, W. Li, H. Jiang and S.Yu, "NeuroSim simulator for compute-in-memory hardware accelerator: validation and benchmark," *Front. Artif. Intell.*, vol. 4, 2021.

[25] X. Jianwei, Y. Rendong, C. Faquan and L. Peilin, "SFANC: Scalable and Flexible Architecture for Neuromorphic Computing," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 31, no. 11, pp. 1826-1838, Nov. 2023, doi: 10.1109/TVLSI.2023.3282239.

[26] T. Cao, Z. Zhang, W. L. Goh, C. Liu, Y. Zhu and Y. Gao, "ECG classification using binary CNN on RRAM crossbar with nonidealities-aware training, readout compensation and CWT preprocessing," *2023*

IEEE Biomedical Circuits and Systems Conference (BioCAS), Toronto, ON, Canada, 2023, pp. 1-5, doi: 10.1109/BioCAS58349.2023.10389002.

[27] A. M. S. Tosson, S. Yu, M. H. Anis and L. Wei, "A Study of the Effect of RRAM Reliability Soft Errors on the Performance of RRAM-Based Neuromorphic Systems," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 25, no. 11, pp. 3125-3137, Nov. 2017, doi: 10.1109/TVLSI.2017.2734819.

[28] M. Zhou, W. Xu, J. Kang and T. Rosing, "TransPIM: A memory-based acceleration via software-hardware co-design for transformer," *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 1071-1085.

[29] Alex Krizhevsky (2009). Learning Multiple Layers of Features from Tiny Images. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pages 770– 778.

[31] A. Hassani and S. Walton, et al., "Escaping the big data paradigm with compact transformers," arXiv preprint arXiv:2104.05704, 2021.

[32] ITRS, "International technology roadmap for semiconductors," 2013.

[33] W. Wang et al., "Efficient training of the memristive deep belief net immune to non-idealities of the synaptic devices," *Adv. Intell. Syst.*, Feb. 2022.

[34] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proc. IEEE*, vol. 106, no. 2, pp. 260-285, Feb. 2018.

[35] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional nural networks," *IEEE J. Emerging Sel. Top. Circuits Syst.*, vol. 9, no. 3, pp. 570-579, Sept. 2019.

[36] Y. Long, X. She and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1769-1774, 2019.

[37] T. Cao, C. Liu, Y. Gao and W. L. Goh, "Parasitic-aware modelling for neural networks implemented with memristor crossbar array," *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 122-126, 2021.

[38] B. Liu et al., "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014.

[39] M. E. Fouda, J. Lee, A. M. Eltawil and F. Kurdahi, "Overcoming crossbar nonidealities in binary neural networks through learning," *2018 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, Athens, Greece, 2018.

[40] T. Cao, C. Liu, Y. Gao and W. L. Goh, "Parasitic-aware modeling and neural network training scheme for energy-efficient processing-in-memory with resistive crossbar array," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 2, pp. 436-444, Jun. 2022.

[41] W. Wang, et al, "Enabling neuromorphic computing: BEOL integration of CMOS RRAM chip and programmable performance," *2019 IEEE International System-on-Chip Conference (SOCC)*, Singapore, pp. 354-358, 2019.

[42] T. Cao et al., "A non-idealities aware software–hardware co-design framework for edge-AI deep neural network implemented on memristive crossbar," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 12, no. 4, pp. 934-943, Dec. 2022.

[43] L. Xia, M. Liu, X. Ning, K. Chakrabarty and Y. Wang, "Fault-tolerant training enabled by on-line fault detection for RRAM-based neural computing systems," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1611-1624, Sept. 2019.

[44] Y. Xiao, Q. Xu and B. Yuan, "Tolerating device-to-device variation for memristive crossbar-based neuromorphic computing systems: a new bayesian perspective," *2023 International Joint Conference on Neural Networks (IJCNN)*, Gold Coast, Australia, 2023, pp. 1-7, doi: 10.1109/IJCNN54540.2023.10191448.

[45] X. Fei, Y. Zhang and W. Zheng, "XB-SIM*: A simulation framework for modeling and exploration of ReRAM-based CNN acceleration design," *Tsinghua Sci. Technol.*, vol. 26, no. 3, pp. 322-334, June 2021.

[46] X. Peng, S. Huang, H. Jiang, A. Lu and S. Yu, "DNN+NeuroSim V2.0: an end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 40, no. 11, pp. 2306-2319, Nov. 2021.

[47] Y. Bi, Q. Xu, H. Geng, S. Chen, and Y. Kang, "AD2VNCS: adversarial defense and device variation-tolerance in memristive crossbar-based neuromorphic computing systems," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 29, no. 1, pp. 8-27, Jan. 2024. doi: 10.1109/TCAD.2023.3600231.

[48] S. K. Roy, A. Patil and N. R. Shanbhag, "Fundamental limits on the computational accuracy of resistive crossbar-based in-memory architectures," *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, Austin, TX, USA, pp. 384-388, 2022.

Tiancheng Cao (Student Member, IEEE) received his B.Eng. (Highest Distinction) and Ph.D. degrees in Electrical and Electronic Engineering from Nanyang Technological University (NTU), Singapore, in 2021 and 2024, respectively. He was awarded the NTU Science and Engineering Undergraduate Scholarship and the prestigious Nanyang President's Graduate Scholarship for his doctoral research. During his Ph.D., he was affiliated as a student researcher with the Institute of Microelectronics (IME) at the Agency for Science, Technology and Research (A*STAR), Singapore.

He is currently a research fellow at the Centre for System Intelligence and Efficiency (CSIE), NTU, Singapore. His research interests span neuromorphic computing, edge computing for the Internet of Medical Things (IoMT), and applications of translational medicine.

Weihao Yu earned his PhD from National University of Singapore in 2024, his Master of Engineering from Sun Yat-sen University in 2019, and his Bachelor of Science from South China Normal University in 2017. His research interests include deep learning, computer vision, and natural language processing.

Yuan Gao (Member, IEEE) received the B.E and M.E degrees in electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2000 and 2002, respectively, and the Ph.D. degree in electrical engineering from the National University of Singapore, Singapore, in 2008.

Since 2007, he has been with the Institute of Microelectronics (IME), Agency for Science, Technology and Research (A*STAR), Singapore. He is currently a principal investigator and principal scientist in the Integrated Circuit Design and Systems (ICDS) Department, where he is leading the next generation intelligent sensor interface IC development. He has authored or coauthored 3 book chapters, more than 120 peer-reviewed international journal and conference papers and has more than 10 US patents granted or filed. He has co-supervised 8 PhD students and he is an accredited A*STAR PhD Scholar supervisor. He received A*STAR Graduate Academy Star Mentor Award in 2023 and IEEE Solid State Circuit Society Outstanding Reviewer Award in 2023. His primary research areas include energy efficient analog and mixed-signal IC design in the emerging areas such as AI hardware, intelligent sensor interface, biomedical microsystem and energy harvesting.

Dr. Gao was TPC member of the IEEE International Solid-State Circuits Conference (ISSCC) between 2015 – 2020 and served as Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS−I: REGULAR PAPERS between 2020 – 2022. Currently he is an Associate Editor of the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS.

Chen Liu received his B. E. degree in Peking University, Beijing, China, in 2014, and the Ph.D. degree in School of integrated circuits, Peking University, Beijing, China, in 2019. He joined the Institute of Microelectronics (IME), Agency for Science, Technology and Research (A*STAR), Singapore since 2019, and currently is a scientist and a principal investigator in Sensors, Actuators & Microsystems Department.

His research interests include the emerging nonvolatile memory devices, piezoelectric and ferroelectric materials and radio frequency MEMS devices.

Tan-Tan Zhang received his B.Sc. degree in Automation from the Fuzhou University, Fuzhou, China, in 2008, and the M.Sc. and Ph.D. degrees in electrical and electronic engineering (now electrical and computer engineering) from the University of Macau, Macao, China, in 2010 and 2018, respectively. Since 2018, he has been with the Institute of Microelectronics (IME), Agency for Science, Technology and Research (A*STAR), Singapore. He was with the State Key Laboratory of Analog and Mixed-Signal VLSI as research assistant from 2010 to 2018. He took short-term internship with SICK AG, Waldkirch, Germany in 2007.

His current research interests include ultra-low power circuits and systems for biomedical applications and energy-efficient sensor interfaces. He was a recipient of the Student Travel Granted Award of IEEE International Solid-State Circuits Conference in 2017.

Prof. Yan (Fellow, IEEE) is currently the Managing Director of Kunlun 2050 Research and Chief Scientist of Kunlun Tech & Skywork AI, and the former Group Chief Scientist of Sea Group. Prof. Yan Shuicheng is a Fellow of Singapore's Academy of Engineering, AAAI, ACM, IEEE, and IAPR. His research areas include computer vision, machine learning, and multimedia analysis. Till now, Prof Yan has published over 800 papers at top international journals and conferences, with an H-index of 140+. He has also been named among the annual World's Highly Cited Researchers nine times. Prof. Yan's team received ten-time winners or honorable-mention prizes at two core competitions, Pascal VOC and ImageNet (ILSVRC), deemed the “World Cup” in the computer vision community. Besides, his team won more than ten best papers and best student paper awards, particularly a grand slam at the ACM Multimedia, the top-tiered conference in multimedia, including the Best Paper Awards thrice, Best Student Paper Awards twice, and Best Demo Award once.

Wang Ling Goh (Senior Member, IEEE) received both her Bachelor of Engineering Degree in Electrical and Electronic Engineering and Doctor of Philosophy in Microelectronics from the Department of Electrical and Electronic Engineering at the Queen’s University of Belfast in United Kingdom in 1990 and 1995, respectively.

Dr Goh joined the School of Electrical and Electronic Engineering (EEE) as a lecturer and became an Associate Professor in 2004. Prior to her current appointment as Associate Dean (Academic) at the Graduate College, she had held many academic positions such as Deputy Director (Undergraduate) of the Renaissance Engineering Programme (Jun 2019 – Jul 2021), Coordinator for the Final Year Projects at School of EEE (Sep 2018 – Jul 2020), Programme Coordinator of B.Eng. (EEE) (Jun 2014 – May 2017), Member of NTU Teaching Council (Oct 2012 – Jun 2013), Associate Dean (Outreach & External Relations) at the College of Engineering (Jan 2010 – Dec 2012), Assistant Chair of Students (Jul 2008 – Dec 2009) and Assistant Head of Division at School of EEE (Sep 2006 – Jun 2008).

Dr Goh participates actively as General Chair or Advisory/Technical Committee Member in various international conferences. She was the General Conference Chair of the 2016 International Symposium on Integrated Circuits (ISIC 2016), held in Singapore, from December 12-14 in 2016. Dr Goh’s research interests include digital/mixed-signal Integrated Circuit (IC), and biomedical and neuromorphic circuits.