An End-to-End In-Memory Computing System based on A 40nm eFlash-based IMC SoC: Circuits, Toolchains, and Systems Co-Design Framework

Tianshuo Bai, Wanru Mao, Guangyao Wang, Hanjie Liu, Aifei Zhang, Jianchao Hu, Shihang Fu, Shuaikai Liu, Xitong Yang, Biao Pan, Wei Xing, Carol Guo, Derek Wang, Wang Kang, *Senior Member, IEEE*

***Abstract*—Despite its promising potential for Artificial Intelligence (AI) applications, current In-Memory Computing (IMC) technology faces a variety of challenges before mass production. One of the major challenges we face is the absence of efficient toolchains for deploying canonical networks on IMC chips. To address this issue, we propose a co-designed framework that integrates circuit, toolchain, and system elements specifically for IMC. More specifically, our framework consists of several key techniques to improve the key performance including (a) an 8-bit hardware-friendly Quantization-Aware Training (QAT) approach to quantify the deep learning network from floating-point data to fixed-point data, (b) a novel operator optimization technique to increase the computing precision when running the algorithm models on the IMC chips, and (c) an efficient mapping strategy based on the Integer Linear Programming (ILP) approach to improve the computation resource utilization of the IMC array. We assess our method on our 40nm eFlash-based IMC SoC chip with voice recognition, speech noise reduction, and person detection tasks. Our experimental results show an accuracy over 94.60% in a quiet environment and 87.27% in a white noise environment and a false recognition rate below 1 time per 24 hours for voice recognition, a 21.53% improvement for the Perceptual Evaluation of Speech Quality (PESQ) for noise reduction, and a 97.80% accuracy in person detection.**

***Index Terms*—In-memory computing, eFlash, toolchains, circuit-toolchain-system co-design framework**

I. Introduction

NOWADAYS, accompanied with the rapid development of algorithms and hardware, Neural Networks (NNs) have attracted extensive attention for their outstanding performance and have been applied in various applications, such as image recognition, voice detection, and semantic segmentation [1-3]. However, with the continuous expansion of the network scale, NN accelerators based on traditional von Neumann architecture face significant challenges owing to the physical separation of memory and processor units. The frequent data movement between the memory and the processor induces high delay and energy consumption, known as the “memory wall” bottleneck [4]. In-Memory Computing (IMC) is a promising solution for breaking the “memory wall”, where computational tasks are processed in the memory unit without data movement [5]. In addition to alleviating the latency and energy consumption caused by excessive data transmission, the high-density computational memory array structure has great potential for massively parallel processing tasks. Such a structure is attractive for NN inference accelerations with trained weight data stored in memory [6], as it can efficiently perform the core NN computing operations, i.e., multiply-and-accumulate (MAC) operations.

To date, numerous memory media have been proposed to implement the IMC structure. Traditional memories, such as Dynamic Random-Access Memory (DRAM), can use a charge-sharing mechanism between cells to implement in-memory logic operations [7,8]. However, their volatile memory cells need to be refreshed after each operation, introducing extra energy. SRAM-based IMC technology can achieve superior speed and scalability with advanced process technology [9], whereas it is volatile and still has problems with high area cost for multi-bit precision operations [10]. Phase Change Memory (PCM) [11], Resistive RAM (ReRAM) [12] and Magnetic RAM (MRAM) [13] are emerging nonvolatile memories and can theoretically realize large-scale crossbar array structure and high throughput. They are under intensive research and development, yet the process is not mature enough for product [14]. Embedded Flash (eFlash) memory is attractive for its high storage precision (could contain multi-bit information per cell), high integration density and process maturity, and is already in production for IMC implementation [15,16]. Therefore, in this work, we adopt a 40nm eFlash-based IMC SoC chip as our end-side device for NN deployment.

Nevertheless, adopting IMC chips for NN accelerators generally requires specific manual implementation [17,19]. To deploy NNs properly, programmers have to choose from multiple options that are logically equivalent but differ dramatically in terms of subgraph splitting, memory reuse, pipelining, and other factors that depend on the software-hardware codesign. This requires a comprehensive cross-layer understanding of the algorithm models, compilation tools, circuits, and other related domain-specific knowledge. Consequently, the portability and deployment cost are seriously problematic, which are the main obstacles that

Manuscript received ?, 2023, revised ??. This work was supported by the Natural Science Foundation of China (62274008), Beijing MSTC Nova Program (Z211100002121014 and Z201100006820042), and Beijing Natural Science Foundation (L223004). Corresponding author: W. Kang.

T. Bai, W. Mao, G. Wang, H. Liu, B. Pan, W. Xing and W. Kang are with the School of Integrated Circuit Science and Engineering, Beihang University, Beijing,100191, China. (E-mail: wang.kang@buaa.edu.cn)

A. Zhang, J. Hu, S. Fu, S. Liu, X. Yang, C. Guo and D. Wang are with the Zhicun Research Lab, Beijing, 100083, China.

hinder the large-scale commercialization of IMC chips [20]. In state-of-the-art (SOTA) deep learning design tools, such as TensorFlow and TVM, they take the NN model definitions as the inputs and generate the code implementations on general-purpose computing hardware, like CPUs, DSPs, and GPUs [21]. Nonetheless, the unique principles and architectures of IMC (especially analog IMC) make these tools generally unsuitable for IMC chips, which have a strong coupling with algorithm models. For example, in low-level deployment, making good utilization of the relatively limited IMC array space and allocating computation parameters in the IMC array efficiently are crucial problems to be solved, which, however, are not solved in SOTA tools [22].

In this work, based on our 40nm eFlash-based IMC SoC (named IMC-SoC thereafter) chip, we propose a circuit-toolchain-system co-design framework for IMC systems. The tools can extend from the user-level NN models to the low-level drivers that directly control the IMC chips. Fig. 4 shows the complete working flow of our tool, and the main innovations of this paper are summarized as follows:

- An optimized hardware-friendly 8-bit Quantization Aware Training (QAT) method exploiting the characteristics of the eFlash cells is proposed to quantify the weights of the NN models from floating-point data to fixed-point data.
- An operator optimization technique involving parameter amplification, weight replication, and multi-point acceleration is proposed to improve the running precision of the NN models when deployed on the IMC-SoC chip.
- An efficient memory space mapping strategy is proposed based on the Integer Linear Programming (ILP) approach to increase the space utilization of the memory array.
- With our end-to-end framework and the related tools, we deployed voice recognition, speech noise reduction, and person detection models on the IMC-SoC chip as case studies to show the feasibility and efficiency of this framework.

The remainder of this paper is organized as follows. Section II presents the structure and working principle of the IMC-SoC chip. Section III lists the key innovation optimizing strategy in our software-hardware co-design toolchain framework. Section IV presents the experimental results. Finally, Section V concludes this work.

II. Structure and Principle of Our IMC-SoC Chip

This section focuses on the fundamental structure and working principle of the IMC-SoC chip. Firstly, we describe the basic architecture of the Flash cell and array and illustrate the analog MAC calculation method with the eFlash IMC array. We then introduce the overall structure of the IMC-SoC chip and analyze its workflow combined with dataflow in detail. Finally, the layout and essential performance of the IMC-SoC are shown and summarized.

A. eFlash In-Memory Computing Array Operating Principle

The IMC-SoC chip used in this work is based on the 40nm eFlash memory, which can implement NN inference computation with extremely low power consumption. The basic building block of the eFlash IMC array is the Flash cell, as shown in Fig. 1(a). Each cell contains two floating-gate transistors sharing the common source line (SL) and controlled by different word lines (WL). The Flash cell works well in the subthreshold mode with a nearly-exponential dependence of the drain current I_{DS} of the memory cell on the gate voltage V_{GS} (Fig. 1(c)) [23]. The slope could stay relatively constant in a broad range of memory states, enabling the gate-coupled circuit to operate in a broad range of memory states, minimally to ~10pA, which ensures the eFlash array to operate with ultra-low power consumption. With dedicated optimizations, each cell can store 8-bit weight and achieve 8-bit MAC precision, which can satisfy the accuracy demand on most of the edge NN applications. Fig. 1(e) shows the relation between the readout current and weights of the eFlash cell, which exhibits high linearity and indicates that every single cell can support 8-bit information well. However, due to the intrinsic physical noises within the analog in-memory operation, low-bit data will suffer more disturbance. In this regard, we designed customized optimizing methods for reducing the impact of the noises, which will be stated in Section III.

The eFlash IMC array has a crossbar-like structure, as shown in Fig. 1(d), which has demonstrated great efficiency benefits in realizing MAC operation [24]. Since all cells in the same row share the same coupling gate voltage, the output current O_j in the j-th column can be expressed by the following formula [15]:

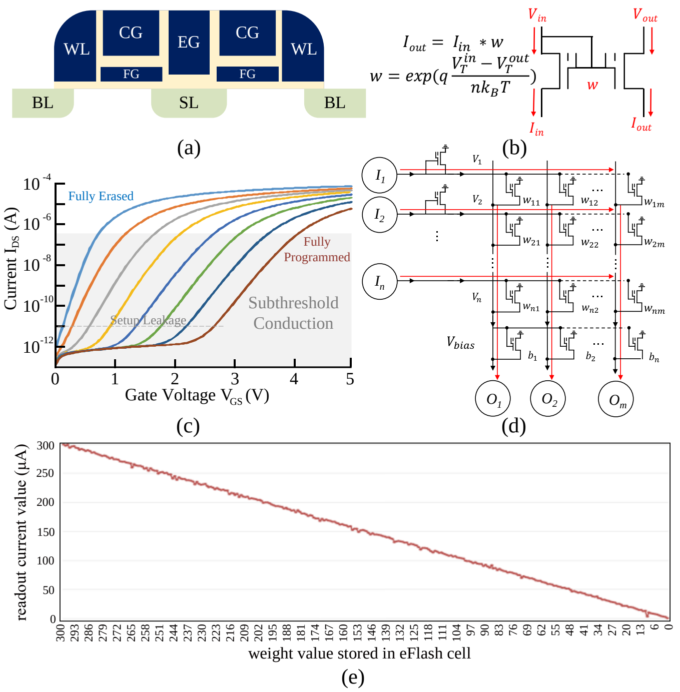
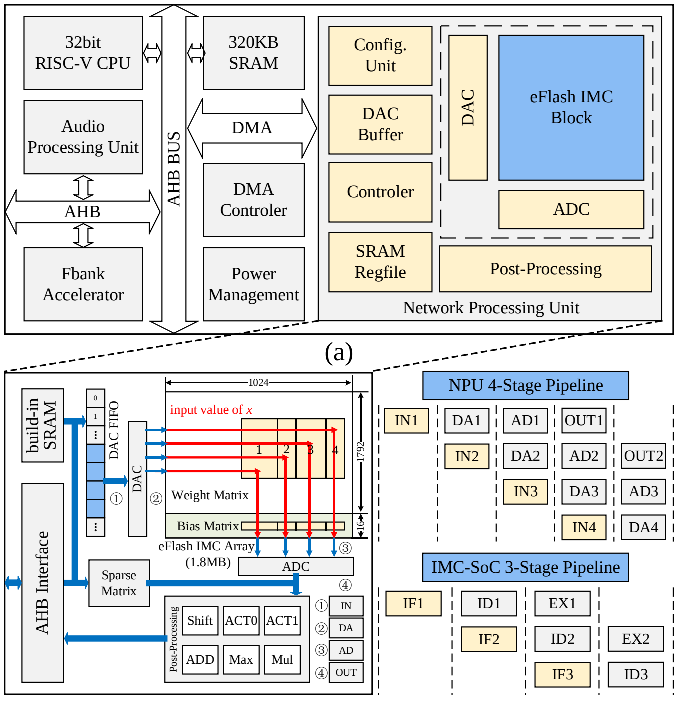


Fig. 1. (a) Schematic cross-section of the Flash cell. (b) The equivalent circuit of the floating-gate transistors. (c) Drain current of the Flash cell, at $V_{DS} = 1V$, as a function of the gate voltage for several memory states. Gray-shaded region shows the subthreshold conduction region. (d) Analog MAC operations realized in the eFlash array. (e) Readout current results of eFlash device stored with different weight value.

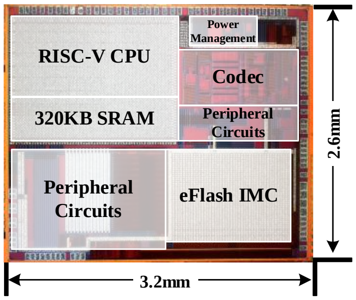


(a)

(b)

(c)

Fig. 2. (a) The overall architecture of the IMC-SOC chip. (b) Major working flow of the eFlash IMC block. (c) Working pipeline stage of the NPU and the IMC-SoC chip.



Summary of the IMC-SoC	
Process	40nm
Package	WLCSP (2.6×3.2 mm²)
Temperature	-20°C~85°C
Application	DNN/GRU/LSTM/TDNN
Array Size	1.8 MB
SRAM Size	320 KB
Weight Precision	8-bit
Supply Voltage	1.8V~3.3V
Frequency *1	148 MHz
Throughput	50 GOPS
Efficiency *2	26~160 TOPS/W

*1: Highest frequency at voltages of 1.8~3.3V.
*2: Performance comprehensively considering IMC array, DAC/ADC, and other peripherals.

Fig. 3. Die photo and the summary table of the IMC-SoC chip.

$$w_{ij} = exp\left\{q\frac{V_{th}^{j} - V_{th}^{(i,j)}}{nk_BT}\right\} \tag{1}$$

$$O_j = (\sum_i w_{ij}I_i) + I_{b_j} \tag{2}$$

where I_i is the input current applied to the cells in the i-th row, w_{ij} is the current-independent coefficient represented by the difference of the threshold voltage V_{th} of the eFlash cells and the peripheral transistors, I_{b_j} is the current imposed by the external circuitry on the output column wires corresponding to a bias value. Hence, Kirchhoff's law enables the calculation of the MAC operation. We deployed an analog neural network on a chip by implementing the eFlash IMC array in the subthreshold region, which obtained >10^3 speed and efficiency improvement compared with the 28nm TrueNorth chip from IBM [25].

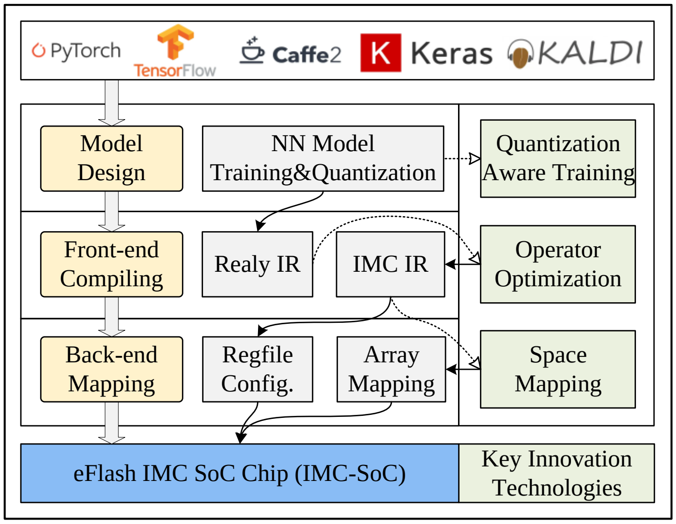


Fig. 4. Our proposed end-to-end circuit-toolchain-system co-design framework and the key innovation technologies for the IMC systems.

B. IMC-SoC Working Principle

Fig. 2(a) shows an overall architecture of the IMC-SoC chip, which contains a Network Processing Unit (NPU) integrated with the former high-performance eFlash IMC array, a 32-bit RISC-V CPU core, Direct Memory Access (DMA) module, 320KB on-chip SRAM, Fbank accelerators for high performance audio processing, and various peripheral devices. With the N307 RISC-V processor core, the IMC-SoC chip adopts a three-stage variable-length pipeline design and can achieve standout energy efficiency and overall cost. The NPU core can interact with other modules in the SoC via the DMA module and perform neural network computing tasks with high parallelism. Fig. 3 shows the die photo and the core attributes of the IMC-SoC chip.

As denoted by the blue arrow in Fig. 2(b), external data is transmitted through the AHB interface to the DAC FIFO (or read from the build-in register file SRAM) for preparation, which is meant to send 448-byte data to the DAC at one time. Afterwards, digital inputs are converted into analog data and entered into the corresponding rows. In the core in-memory computing process, the weight and bias data are loaded to the specified position of the eFlash array in advance, where the input vector gets multiplied. Finally, the analog results accumulated in each column outflow through the ADC and are handled by the post-processing module. Additionally, a digital computing unit for sparse MAC operation is also adopted to corporately calculate high-bit data to improve precision under certain conditions. Depending on this working principle, the NPU-level pipeline can be divided into four stages (see Fig. 2(b)(c)), which can maximally make use of each module to complete the continuous data processing. The NPU module can realize a computational capability of 50GOPS and an energy efficiency of 26-160 TOPS/W.

III. Framework of Our Toolchain

This section primarily discusses the workflow of our framework and the key technologies adopted. Fig. 4 illustrates the architecture of our end-to-end framework for the IMC. The overall input is from canonical deep learning models, and the final underlying output is the binary data flow that can be directly deployed on the IMC-SoC chip. Through our innovative workflow, we deploy deep learning models with higher accuracy, faster speed, better compatibility, and larger memory utilization.

A. 8-bit Quantization-Aware Training Method

Deep learning models have achieved enormous success in many fields. They are commonly trained with floating-point 32-bit (FP32) vectors. However, the SOTA IMC devices generally can support only up to 8-bit precision. To resolve this issue, quantization is commonly adopted to map the floating-point weights to fixed-point numbers with low bit widths (e.g., 8-bit or 4-bit). As a consequence, faster inference and smaller memory footprint can also be achieved with quantization methods [26,27]. In general, there are two types of quantization algorithms: Post-Training Quantization and Quantization-Aware Training (QAT). The former method requires no re-training and is more lightweight, whereas it may introduce unacceptable errors incurred by low-bit quantization [28]. In contrast, the QAT method models the errors during the training process and offers better solutions by bridging the gap between full-precision accuracy with low-bit precision.

However, the standard QAT method that quantized only the weights into low bitwise is not hardware-friendly enough. In our IMC-SoC chip, the typical matrix formula for NN computing is defined as:

$$Y = \sigma\left(\frac{WX + b}{G}\right) \tag{3}$$

where W, b, X, and G correspond to 8-bit signed weights and bias parameters, 8-bit unsigned input data, and scaling factor in a matrix operation respectively. σ is the activation function, and Y represents the final 8-bit unsigned calculation results.

To exploit the hardware features of the IMC-SoC chip, specific optimizations have been made based on the typical QAT method [29], as shown in Fig. 5. The original scaling factor G for NN inference is given by Eq. (4), where G_{IN} ,G_W and G_{Out} are the scaling multiples for inputs, weights, and outputs. The bias is quantified with the scaling factor G_{Bias} defined in Eq. (5):

$$G = \frac{G_{IN}G_W}{G_{Out}} \tag{4}$$

$$G_{Bias} = G_{IN}G_W \tag{5}$$

In our optimized QAT, different from usual floating-point representation, we limit G_{IN}, G_W and G_{OUT} to the powers-of-two, that makes it feasible to scale data through simple shifting operation and greatly reduces the complexity of mapping from floating-point to integer. More especially, considering the bell-shaped distribution characteristic of DNN weights [30], we refine the quantization range of weights into [-128, 127] symmetrically with 0, which reduces the computation cost of handling the zero point. Meanwhile, to efficiently support the 8-bit precision of the IMC array, we use the straight-through estimator [31] for approximation to avoid zero or undefined values while rounding for the gradients of the quantization operations. With the above optimizations, we can deploy deep learning models on the IMC-SoC chip with negligible accuracy loss.

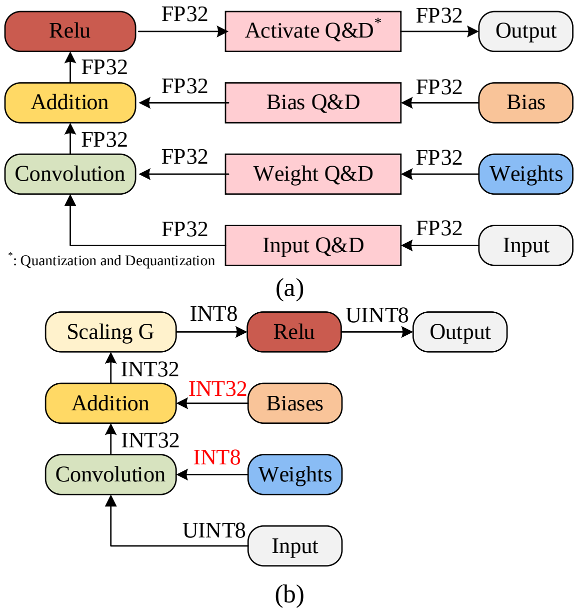


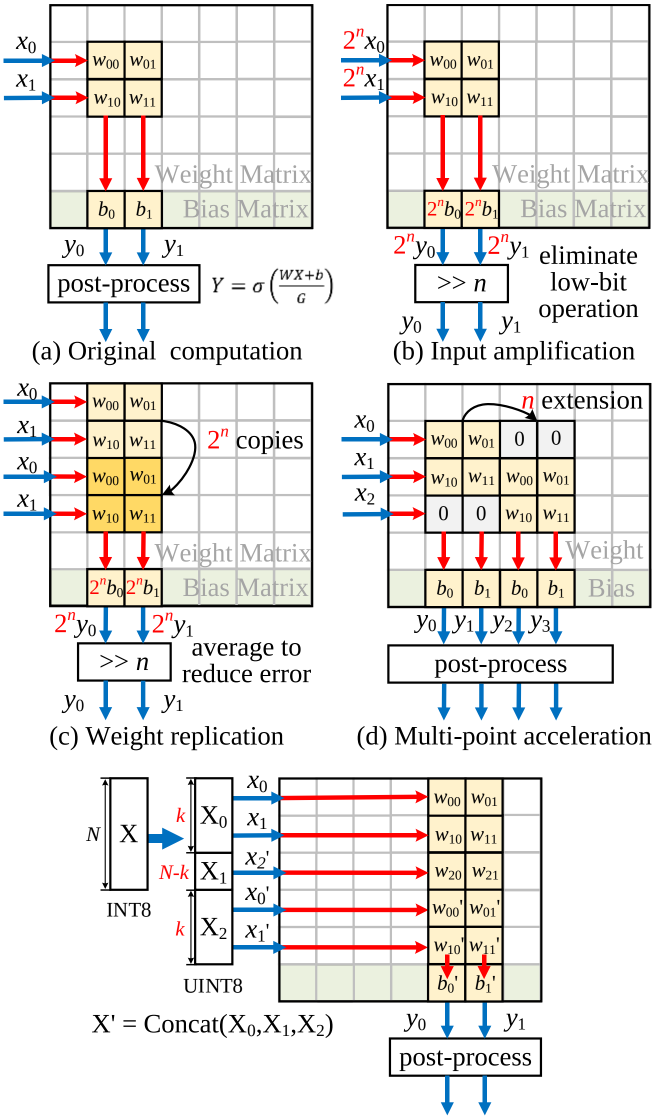
Fig. 5. The QAT method for (a) training and (b) inference flows. FP32 data needs to be quantified to INT8/UINT8 type.

B. IMC Operator Optimization

Through the computational paradigm Eq. (3), various typical networks, such as time delay neural network (TDNN) and long short-term memory (LSTM) network, can run on our IMC-SoC chip. However, when executing 8-bit MAC operations, there may exist unavoidable accuracy loss of low-bit data because of the process variations and noises in the eFlash cells [15]. Thus, while transferring relay Intermediate Representation (Relay-IR) [21] to customized IMC-IR (as shown in Fig. 4), several operator optimization methods have been designed to reduce the impact of low-bit accuracy loss on overall MACs. Besides, negative input handling and convolution acceleration can also be attained at this stage. A sample demonstration of the optimization methods is shown in Fig. 6. Additionally, these optimization options are configurable and users can flexibly enable or disable the corresponding optimization strategies based on the accuracy requirements and computing needs of the application scenarios.

1) **Amplification Strategy of W/X/b**

   Based on the computing formula in Eq. (5), when W, X, and b are amplified, the original theoretical calculation results can be obtained by adjusting the value of G accordingly. On this premise, through magnifying the



(a) Original computation

(b) Input amplification

(c) Weight replication

(d) Multi-point acceleration

(e) Positive and Negative optimization

Fig. 6. Demonstration of the IMC operator optimizations. (a) Original computing method for weight matrix ***w*** and input vector ***x***. Optimized computing method with (b) input vector amplification, (c) weight matrix replication, (d) multi-point acceleration mode and (e) positive and negative optimization.

computing parameters by 2^n before loading them into the IMC memory array, the low-bit information with relatively bigger error would be removed after the post-processing module, which eliminates the impact of low-bit data computation errors on the overall results. For more flexibility, the amplification of X and W can be independently controlled. Fig. 6(b) depicts the input amplification strategy. The weight optimization works in an analogous way by magnifying the weight data. In practical applications, most dense weight matrix is stored in the low 6 bits. Therefore, the amplification strategy usually sets the factor *n* as 1 or 2. Theoretically, larger factor *n* could acquire higher accuracy but may cause the overflow of bias.

2) **Replication Strategy of W**

As shown in Fig. 6(c), this method replicates the weight data in the row direction of the IMC memory array, which can make 2^n times copies. By appending corresponding duplicated input X, extra MAC operations with the same parameters are performed repeatedly. In this way, the final error introduced by the analog processing and noises can be minimized by taking averaging results of multiple computations with the same argument. This method could enhance the accuracy without changing the input information and causing overflow of input value, which is well suited for small-scale networks. For a single convolution layer [3×3, 64], this method (replicate 2/4 times) can reduce the MAC computation error rate by 5.8%~9.5%.

3) **High-Bit Sparse Matrix Mode**

This method aims to perform the high-bit portion data with a digital processor that could trade a bit lower efficiency for higher accuracy. Before executing analog operation in NPU, sparse matrix mode could be configured as analog-only, digital-analog hybrid, or digital-only, which determines the numbers of bits that the calculated data need to be left-shifted. For analog-only mode, all the weight data are stored in the IMC array to execute MAC operations. For the digital-analog hybrid mode, the weight value could be optionally left-shifted by 1~7 bits. As shown in Fig. 2(b), high-bit digital data flows into the digital sparse matrix processing unit, and the working results are combined with the IMC calculation result of low-8-bit weights. For digital-only mode, the weights do not need to be stored in the array and are processed directly in the sparse matrix unit. Generally, the high-bit weights are sparse (>99% zeros in top 2 bits), thus the sparse units would only incur <10% extra power and area overhead. This mode is often combined with the amplification strategies, where the left-shifted high bits are sent to the digital processors to ensure no information loss. By combining both methods, with 1~2 bits shifted, the MAC computation error rate can be reduced by 5.3%~9.2% respectively, without occupying extra array resources.

4) **Multi-point Convolution Optimizations**

The sketch map of this method is shown in Fig. 6(d), where w_{ij} and x_i represent the data of the weight matrix and input vector in the convolution operations. In the multi-point working mode, the weight matrix is expanded by *n* times in the column direction, which enables higher input parallelism. For instance, for convolution operations with a 2×7 input vector, 2×2 weight matrix, and 1 stride, it is originally necessary to slide 6 times to fulfill the computation. Instead, by turning on the 2× multi-point acceleration mode, the weight matrix is extended to 2×3 with zero line and the sliding step is raised to 2, which reduces the computing round from 6 to 3. When enabling 2/3/4× multi-point acceleration mode, the actual computation time of one single convolution layer [3×3, 64] can be reduced by 1.8/2.8/3.6× (of an available upper-bound of 2/3/4×). However, this optimization would sacrifice more array space and need to be turned on or off flexibly during task allocation.

5) **Positive and Negative (PN) Optimization**
This method aims for handling signed input vector. Since the IMC memory array can only process MAC operations with nonnegative input value, special treatments are required for operators with negative inputs. The simplest approach is to add one positive value to the INT8 input vector, ensuring that each input data is nonnegative. However, this method may lead to excessive information loss due to the overflow of the input data and could result in an overly large bias value, potentially exceeding the maximum limit.

Hence, we propose a dynamic PN optimization method that could extract information from the INT8 inputs and transform them into the UINT8 representations. As shown in Fig. 6(e), for the input vector $X[0\colon N]$, we set a PN length k ($k \in [0\colon N]$) to split the inputs into three parts and recombine them into a new vector X':

$$X_1 = Relu(X[0\colon k])$$

$$X_2 = X[k\colon N] + 128$$

$$X_3 = Relu(-X[0\colon k])$$

$$X' = Concat(X_1, X_2, X_3)$$

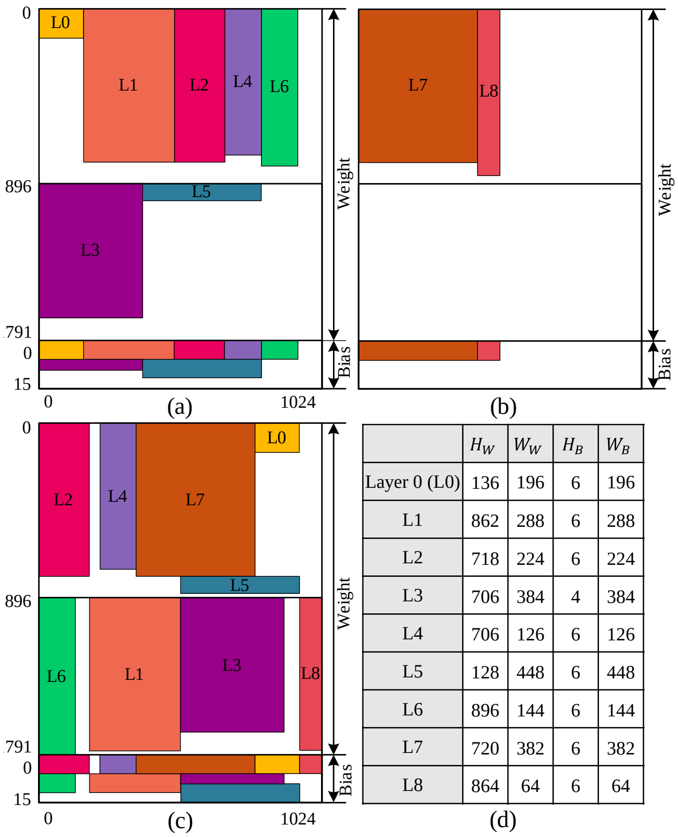
where X' contains both positive and negative values in a non-negative form. Concurrently, the toolchain will adjust the corresponding weights W and bias parameters based on the setting of k to ensure the final outputs match the theoretical value:

$$W' = [W, -W[0\colon N]]$$

$$bias' = bias - 128 \times sum(W[k\colon N])$$

By dynamically configuring k, we could avoid excessive information loss in X' and keep $bias'$ within the allowed range. Interestingly, when $k = 0$, our PN optimization method is equivalent to the originally mentioned method. The choice of k would not only affect the effects of IMC computation but also the array space, which needs comprehensive balance based on the scale of network parameters.

Different optimization methods have specific purposes. In practical applications, we could implement different combinations of the optimizations according to the characteristics of the NN models, which depend on aspects like weight parameter size, input data type, accuracy requirement, speed requirement, etc. In general, the optimization pattern of ‘Replication + PN + Multi-point’ could meet most of the demands. For high-accuracy cases, we could typically choose the ‘Amplification + High-bit Sparse’ modes, utilizing the digital processor to further improve the computing precision. For high-speed cases, we could allocate more array space to the multi-point acceleration mode, selectively weakening or even disabling the replication strategy and further reducing the number of computing iterations. Nevertheless, the combined optimization would bring about more complex effects. Varying across different networks, there still needs comprehensive debugging during engineering verification.



	H_W	W_W	H_B	W_B
Layer 0 (L0)	136	196	6	196
L1	862	288	6	288
L2	718	224	6	224
L3	706	384	4	384
L4	706	126	6	126
L5	128	448	6	448
L6	896	144	6	144
L7	720	382	6	382
L8	864	64	6	64

Fig. 7. Weights and bias distributions of the audio denoising NN with (a)-(b) traditional sequential mapping methods and (c) our proposed ILP mapping method. (d) Dimensional information of each layer of the audio network.

C. Integer Linear Programming Space Mapping

Deep neural networks rely extensively on MAC operations, which naturally make them suitable for IMC array structures. In the crossbar array, the activation of NN is performed along the rows of the memory array, multiplied by synaptic weights according to Ohm’s law and summed over each column according to Kirchhoff’s current law [32]. This allows the crossbar array to perform MAC operations efficiently and quickly at the storage location of data. Thus, at the stage of back-end mapping, it is an essential technique to automatically map the weight parameters from specific NN models to the IMC memory array for completing the MAC operations. This technique enables the IMC-SoC chip to execute various NN models without manual intervention or reconfiguration. However, due to the different scales between the IMC memory array and the weight parameters of each network layer, the utilization of the memory space would greatly affect the overall computing energy and latency. Therefore, it is important to find an optimal or near-optimal way to allocate the weight parameters in the memory array. For example, a typical NN for audio denoising requiring different allocations is shown in Fig. 7. Here, squares with different colors represent the weights and bias parameters in different layers of the neural network, where the scales of each data block are shown in Fig. 7(d).

TABLE I
ERROR ANALYSIS OF PROGRAMMED AND EXPECTED WEIGHTS FOR A 5-LAYER NETWORK

Layer	Mean(std(error))	Mean(error)	Fitted Slope	Fitted Intercept	Cosine Similarity
0	3.02	-0.05	0.98	0.05	0.9902
1	1.77	0.26	1	0.26	0.9958
2	1.74	0.15	0.99	0.15	0.9943
3	2.44	-0.05	1	-0.04	0.9966
4	2.22	0.07	1	0.11	0.9955

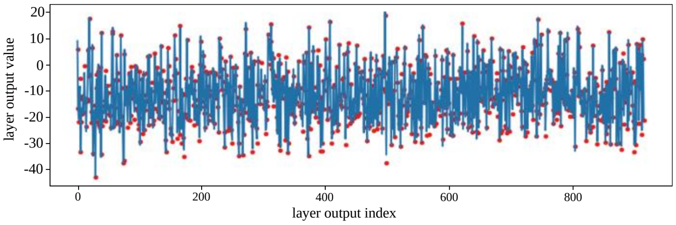


Fig. 8. Output results of the continuous 5-layer network ideal value (blue line) and IMC-SoC chip readout value (red point).

In traditional sequential mapping methods, weights are loaded into the whole memory array (1792×896 in our IMC-SoC chip), which means that the weights are mapped to the memory array in the same order as they appear in the NN model, without considering the size or shape of the weight blocks. When the weight block cannot be loaded into the current free space, the system will keep waiting until the existing matrix is processed. This would cause significant idle time and large energy consumption, as the system cannot perform any computation until the required weights are loaded. As shown in Fig. 7(a)-(b), the entire denoise needs to be separately loaded into the array twice and the highest memory space utilization for these two loads is only 53.6%.

To further improve the efficiency of the hardware resource, we propose an Integer Linear Programming (ILP) space mapping strategy to convert the limitations of the computing parameter's location into the constraints and objectives of ILP problem. The constraints for each layer mainly include:

1) The coordinate of the weights cannot exceed the range of the memory array.
2) The weights and bias cannot be stored in different regions (across line 896).
3) The column addresses of the bias and weights are equal.
4) The weights should occupy as little range of the column as possible.

By treating these boundary constraints as integer constraint problems, a candidate set of all feasible solutions can be constructed by the linear programming method. After obtaining the set of all solutions, a solver needs to be constructed to find the optimal or feasible solution. Here, in addition to constraining the parameter range, we also need to define the objective function of the problem. As for the weight allocation problem, our goal is to minimize the number of memory array sections that are taken by the input NN weight blocks. The solver will therewith return the solution status. For solvable cases, the solver will select one of the schemes with the smallest occupancy and return the memory addresses of each weight block. As shown in Fig. 7(c), the whole weights and bias parameters of the audio NN can be allocated into the memory array at once, and the memory space utilization can be raised to 71.7%.

TABLE II
ACCURACY OF VOICE RECOGNITION IN DIFFERENT SCENARIOS WITH/WITHOUT NOISE

Scenario	Pure	White Noise (10dB)	Pink Noise (10dB)
English Commands*[1] (11 words*[2], 75dB)	98.18%	87.27%	90.18%
Video Control (12 words, 75dB)	98.70%	92.70%	97.70%
Smart Speaker*[3] (30 words, 75dB)	98.12%	93.54%	97.50%
Desktop Control (40 words, 75dB)	95.70%	89.60%	88.90%
Command Recognition (100 words, 75 dB)	94.60%	89.04%	85.04%

*[1]: The others for recognition are Chinese words.
*[2]: The words here refer to the vocabulary for specific scenario. Each word was tested repeatedly for over 100 times.
*[3]: False recognition rate of the wake word is below 1 time per 24 hours.

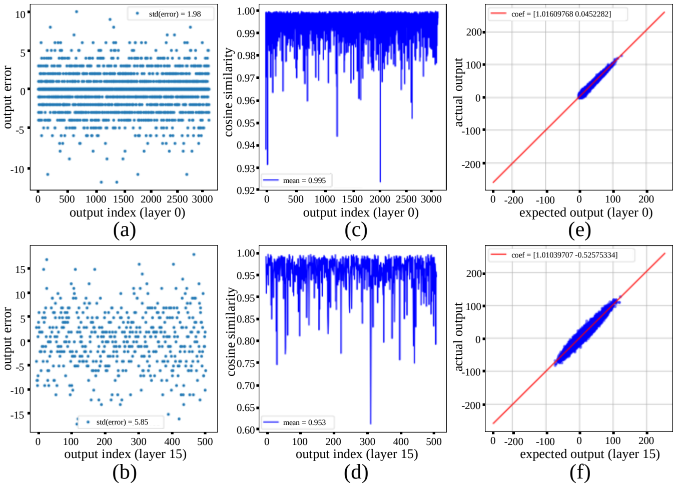


Fig. 9. Error analysis of the actual outputs from the IMC-SoC chip and expected output data in the first layer and last layer of the voice recognition network. (a)-(b) Output standard deviation of the errors and (c)-(d) cosine similarity of each column in the IMC-SoC array. (e)-(f) Linear fitting curves of the actual and the expected outputs of the IMC-SoC chip.

IV. CASE STUDY AND RESULTS

This section primarily presents our case studies and the implementation results. First, we introduce the setup of the toolchain. Then, we conduct the feasibility test on the adopted IMC-SoC chip. Finally, based on our IMC-SoC chip and toolchain system, we deploy voice recognition, noise reduction and person detection as case studies respectively, in order to verify the deployment effects of our end-to-end IMC-SoC system and demonstrate the corresponding data results.

A. Toolchain Setup

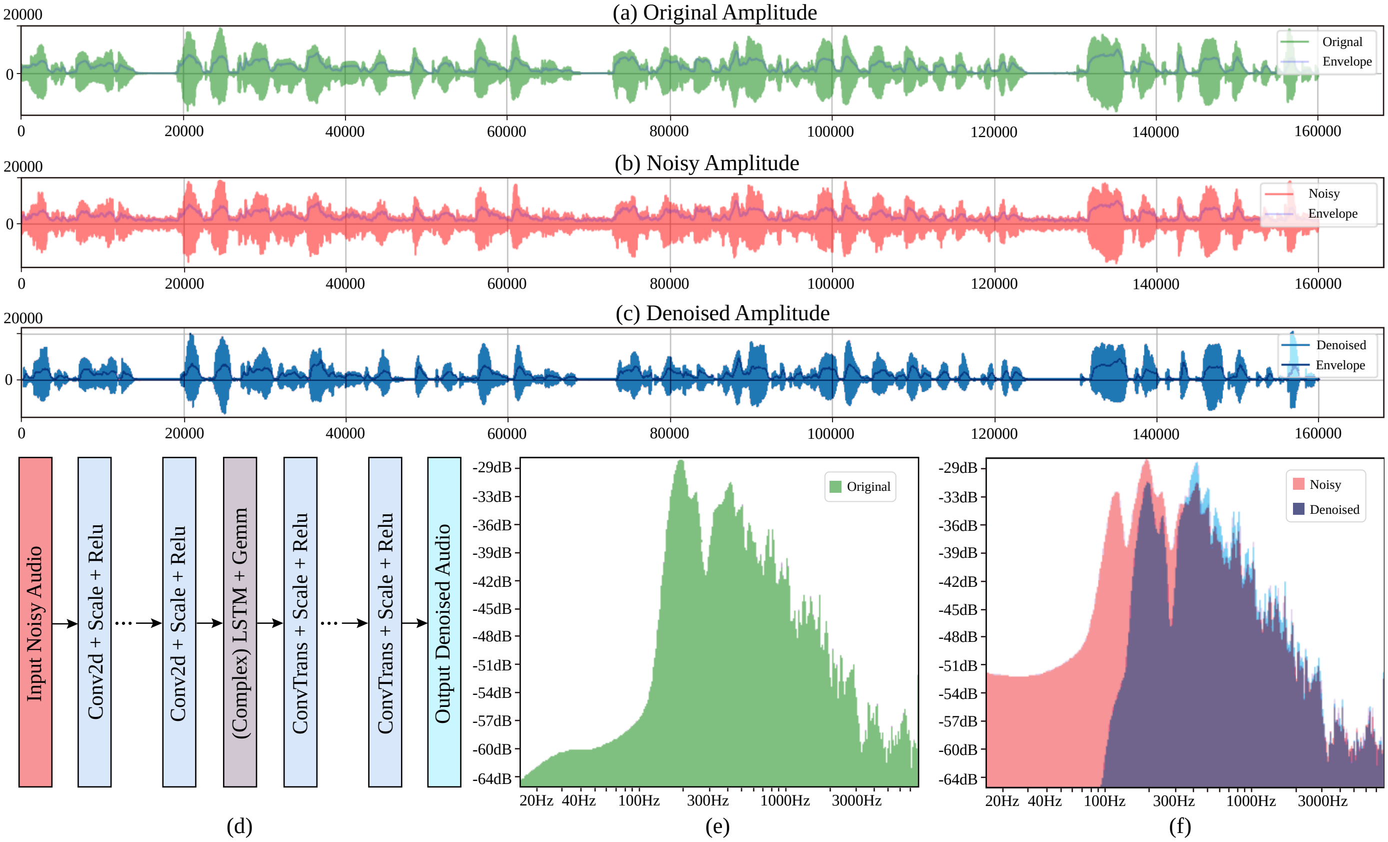


Fig. 10. Amplitude diagram of the (a) original, (b) noisy and (c) denoised audios. (d) Optimized CIM-friendly DCCRN architecture for the audio denoising. Frequency diagram of the (e) original, (f) noisy (red) and denoised (blue) audios.

This section provides a comprehensive description of the complete workflow of our end-to-end IMC system. In a deep learning project, the most pressing concern is about the deployment of the NN model. The toolchains in this framework enable the mapping of NN models to the IMC-SoC chip and the generation of corresponding configuration files for various applications. As illustrated in Fig. 4, the front-end design supports multiple mature deep learning frameworks, such as PyTorch, TensorFlow, Caffe2, Keras and Kaldi. First, by generalizing the interface functions, our toolchain reads the quantified NN models (.pb file) and generates the unified Relay-IR of the computational graph. At this stage, the combination of the operator optimization strategies (presented in Section III.B) is applied to improve the computational performance and generate the customized IMC-IR that is more suitable for the IMC-SoC chip. Then, the IMC-IR and the related parameters are compiled into profiles that are integrated into the SDK and downloaded to the IMC-SoC chip.

TABLE III
COMPARISON OF THE PESQ SCORES FOR TEN TEST AUDIOS BEFORE AND AFTER DENOISE PROCESSING

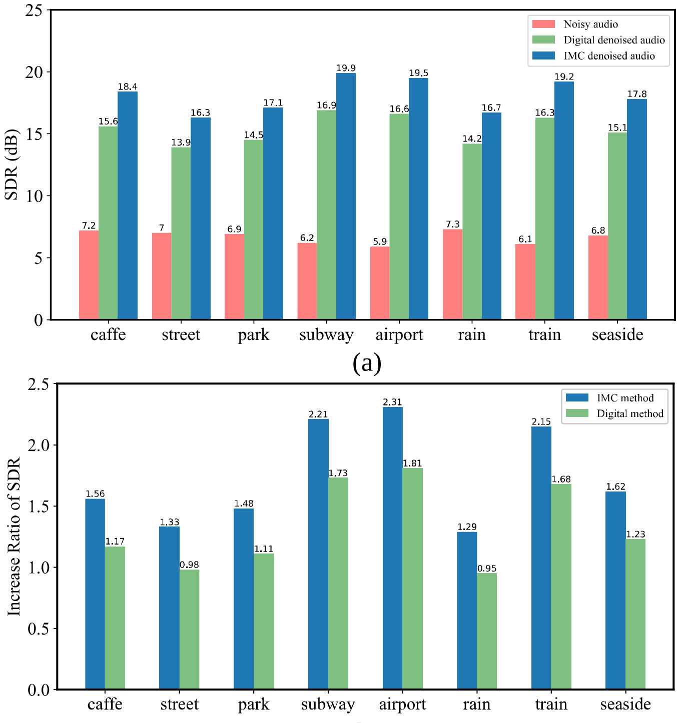
Audio	PESQ.wb[*1]		PESQ.nb[*2]		PESQv2[*3]	
	Noisy	Denoised	Noisy	Denoised	Noisy	Denoised
1	1.349	2.869	2.899	3.789	3.051	3.679
2	1.325	1.672	1.759	2.176	2.148	2.533
3	1.541	2.025	2.115	2.623	2.483	2.865
4	1.441	2.070	2.094	2.714	2.466	2.926
5	1.748	2.207	2.491	3.005	2.771	3.123
6	1.568	2.027	2.530	2.690	2.799	2.910
7	1.302	1.713	1.796	2.374	2.187	2.687
8	1.280	2.363	1.525	2.880	1.856	3.040
9	1.463	2.163	1.894	2.739	2.286	2.944
10	1.218	2.101	1.896	2.694	2.288	2.913
Average	1.423	2.121	2.100	2.768	2.434	2.962

[*1]: wide-band. [*2]: narrow-band. [*3]: Combination of wb and nb.

B. Feasibility Validation

To verify the feasibility of our framework and the related tools, we took a 5-layer voice recognition network as a verifying example. To ensure the consistency of the processed data, we compared the theoretical weights from the NN model with the actual read-back programmed weights from the IMC-SoC chip. As shown in TABLE I, the average standard deviation (std) and the mean value of the error for each array column are much smaller than the data width [-128,127]. The average cosine similarity (indicating the consistency) between the theoretical model weights and the actual read-back values (from the chip) of each layer is above 99%, which satisfies the requirements to accurately deploy the NN weights on the IMC-SoC chip.

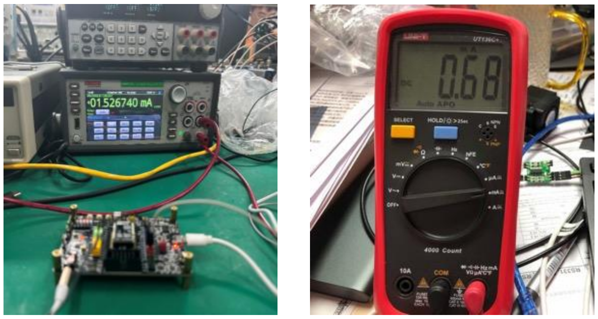
Based on this framework and toolchain, we further evaluate the performance with voice recognition, noise reduction and person detection as case studies using the fabricated IMC-SoC test chip. The corresponding NN models were pre-loaded into



(a)

(b)

Fig. 11. (a) SDR and (b) the increase ratio of SDR after implementing noise reduction by traditional digital scheme and IMC method in eight common environments.



(a) (b)

Fig. 12. (a) Power measurement results of audio noise reduction and (b) words recognition (100 words) task.

the memory array through the programmer. The input signals (e.g., human voices, environmental sounds, person images, etc.) were collected in advance and stored on the PC host, then transferred into the chip test board via USB-SPI converter. After the IMC-SoC chip finished the inference computation, the results were sent back through the SPI bus to the USB-SPI converter, and displayed on the test host via the USB virtual serial port. Detailed experimental results will be discussed in the following sections.

C. Voice Recognition

For end-side devices, voice recognition has become ubiquitous, which serves as a great study case with a practical purpose. we deployed a 16-layer TDNN voice recognition model on the IMC-SoC chip. During the inference process, we obtained the actual outputs of each layer of the network through real-time precision analysis tools and compared them with the expected outputs from the simulation. Taking the first layer and the last layer of the network as examples (see Fig. 9), we analyzed the error between the actual values and the expected outputs from the IMC-SoC chip for each input frame of the voice. In Fig. 9(a), after the processing of the first layer, the error distributions in the scatter plot are mostly concentrated around “0” and the standard deviation is only 1.98. The average cosine similarity between the actual output and the theoretical values reaches 99.5%, as shown in Fig. 9(c), which proves the accuracy of single-layer MAC operations with the IMC-SoC chip and our framework. However, during the processing of a multi-layer network, the errors will accumulate layer-by-layer inevitably. We can find from the last-layer error analysis in Fig. 9(b) and Fig. 9(d), the accuracy of the system has a little decrease. Nevertheless, we can find from Fig. 9(e)-(f) that the slope of the linear fit curves is close to one.

As shown in TABLE II, we tested the phonetic keywords (75 dB) in different scenarios. Without noisy interference, the worst recognition rate is above 94.60%. With the addition of white noise (about 10dB) and pink noise (about 10dB), the worst recognition rates are 87.27% and 85.04% respectively. In addition, the false recognition rate for wake-words is less than once per 24 hours, which will effectively reduce the battery drains of the system caused by false wake-up in the standby mode. TABLE IV presents the comparison of the computility and power consumption between the traditional digital scheme and our IMC-SoC. Despite the memory array can only store 1.8MB parameters, under the efficient mapping and acceleration strategy of our toolchain, the IMC-SoC could finally provide 4~20x higher computability and merely 0.07~0.5x power consumption compared with the digital scheme. For example, in the 100 words recognition scenario, the average operating current of the IMC-SoC is only 0.7mA. In practical applications like smart earphones and other edge devices, our end-to-end IMC system could significantly extend the battery life and improve the response speed for users. This validates the huge potential and practical value of IMC design for applications with high energy efficiency requirements.

D. Noise Reduction

In the field of audio processing, noise reduction has been widely used and has been proved as an effective method. For this case study, we deployed an optimized CIM-friendly DCCRN model with complex CNN layers and one LSTM layer (as a case study) on our IMC-SoC chip to denoise different audios (Fig. 10(d)), where the both CNN and recurrent neural network structures can handle complex-valued operation [33]. To observe the noise reduction effect, we collected ten raw audios as benchmarks and added noises artificially. Afterwards, we used the system with the IMC-SoC chip and our framework for the denoising task. We can observe the denoising effect from the amplitude and frequency spectrums of the noisy and the denoised audios. As shown in Fig. 10 (a)-(c), the red noisy spectrum undertakes more low-amplitude interference noise than the green original audio, while the blue denoised spectrum is largely restored. From the comparison of the frequency spectrum shown in Fig. 10(e)-(f), it could be clearly found that the amount of low-frequency

TABLE IV
COMPARISON OF THE IMC PERFORMANCE WITH DIGITAL METHOD IN DIFFERENT APPLICATION SCENARIOS

Application	Traditional Digital Scheme		IMC Scheme	
	Computility	Power Consumption	Computility	Power Consumption
40 words Recognition	30Mops	2mA	400Mops	0.5mA
100 words Recognition	100Mops	10mA	400Mops	0.7mA
Voice Wake-up	20Mops	0.6mA	400Mops	0.3mA
Noise Reduction	150Mops	15mA	1024Mops	1.5mA

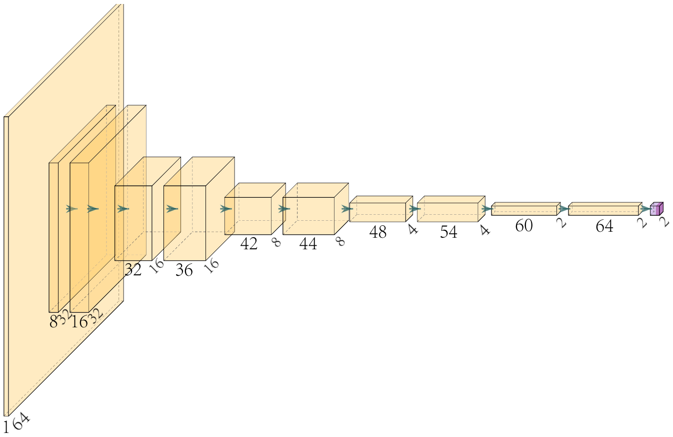
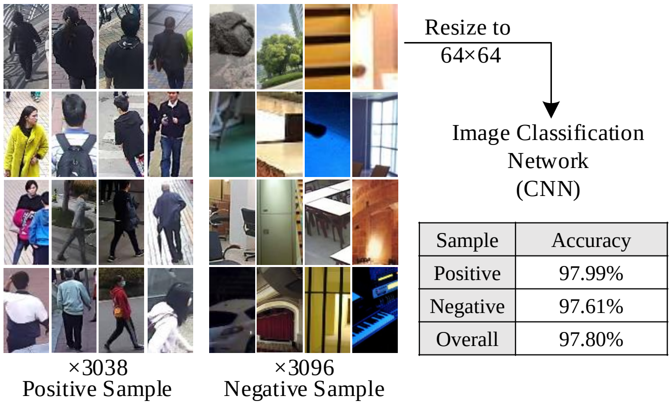


Fig. 13. Structure of our customized person detection network. The model consists of ten convolutional layers and one linear layer, which is used to extract features and classify the image into positive and negative (with or without person).



Sample	Accuracy
Positive	97.99%
Negative	97.61%
Overall	97.80%

Fig. 14. Accuracy of person detection with customized image classification network implemented on the IMC-SoC chip.

noise was added to the original audio and was eliminated after the noise reduction processing. The spectral profiles visually demonstrate that our approach can effectively suppress low-frequency noise while preserving the original speech components. Additionally, the performance of the audio can be quantitively assessed using the Perceptual Evaluation of Speech Quality (PESQ) parameter, considering both symmetric disturbance d_{SYM} and asymmetric disturbance d_{ASYM} of the audio signal. The PESQ score is calculated according to Eq. (6) and generally ranges from 1.0 (bad) to 4.5 (no distortion). Further details on the evaluation framework could be referred to [34,35]).

$$PESQ = 4.5 - k_{SYM} d_{SYM} - k_{ASYM} d_{ASYM} \qquad (6)$$

In TABLE III, we listed the PESQ scores on all test audios and calculated the average scores. We can find that after using the noise reduction network deployed on the IMC-SoC chip, the average PESQ score (here PESQv2 combined the wide-band and narrow-band denoise effects) is improved by 0.524 (by 21.53%) compared to the noisy audio.

Furthermore, we conducted tests on the IMC function to reduce audio noise in common environments such as cafes, streets, parks, etc., and compared the noise reduction results with traditional digital schemes. To evaluate the denoising effects of audio, we use the Signal-to-Distortion Ratio (SDR) to measure the quality of the audio. The calculation method is shown in Eq. (7), which computes the ratio of the power of the input signal to the power of the difference between input and reconstructed signals [36].

$$SDR_{(dB)} = 10\, log_{10}\left(\frac{\left\| s_{target} \right\|^2}{\left\| e_{interf} + e_{noise} + e_{artif} \right\|^2}\right) \qquad (7)$$

As shown in Fig.11, the SDR of the denoised audio with our IMC system is on average 17.6% higher than that of digital measurement. The largest SDR promotion ratio in the airport environment reaches 231%. At the same time, although the computational complexity of the IMC solution is six times higher, the operating power consumption is reduced by ten times to only 1.5mA (see TABLE IV). In summary, the proposed method delivers great energy efficiency improvement on the end-side devices. Compared with SOTA GPU solutions, our end-to-end IMC system can achieve ~10× energy efficiency [37].

E. Person Detection

As can be seen from Fig. 3, the array size (only 1.8MB) of the IMC-SoC chip is limited and is rather difficult to complete large-scale image processing tasks. Through exploiting our framework, we can maximally improve the utilization efficiency of the hardware and reduce the accuracy loss of the algorithm. Taking person detection as a case study, we designed a customized image classification network with ten convolution layers and one fully connected layer to show the feasibility, as shown in Fig. 13. For each input picture, we resized its input matrix size into 64×64 for processing on the IMC-SoC chip. Towards a testing collection dataset consists of 3038 pictures with person and 3096 pictures without person, the average detection accuracy is up to 97.80%.

V. CONCLUSION

We present an end-to-end circuit-toolchain-system co-design framework for IMC systems. First, we adopt an 8-bit hardware-friendly QAT method to train high-accuracy NN models and to achieve fast inference speed and a small memory footprint on the IMC chip. Second, we propose several operator optimizations to minimize the accuracy loss and accelerate convolutional computation. Third, to make better use of the memory space, we convert the weight mapping problem into an ILP to improve the hardware utilization efficiency. By using a commercial eFlash-based IMC-SoC chip, for the first time, we verify the feasibility of the framework with three case studies. For voice recognition, the accuracy can be over 94.60% (in a quiet environment) and 87.27% (in a white noise environment), and the false recognition rate is below 1 time per 24 hours. After deploying a speech noise reduction network on the IMC-SoC chip, low-frequency noise can be significantly eliminated and the PESQ score is increased by an average of 0.524 (by 21.53%). For image processing, our framework can maximize the utilization of the chip and deliver a detection accuracy up to 97.8%. Our work proposes a reliable framework for the efficient implementation of NN algorithms on IMC chips with excellent performance and consistency.

REFERENCES

[1] H. Zhang et al., "CP-SRAM: charge-pulsation SRAM macro for ultra-high energy-efficiency computing-in-memory," in *DAC*, New York, NY, USA, July 2022, pp.109-114.

[2] OI. Abiodun et al., "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol.4, no.11, Nov. 2018, doi: 10.1016 /j.heliyon.2018.e00938.

[3] A. Krizhevsky et al., "ImageNet classification with deep convolutional neural networks," in *Commun. ACM*, New York, USA, June, 2017, pp. 84-90.

[4] K. Lee et al., "A Charge-Sharing based 8T SRAM In-Memory Computing for Edge DNN Acceleration," in *DAC*, San Francisco, CA, USA, 2021, pp.739-744.

[5] N. Verma et al., "In-Memory Computing: Advances and Prospects," *IEEE MSSC*, vol. 11, no. 3, pp. 43-55, summer 2019, doi: 10.1109/MSSC.2019.2922889.

[6] A. Sebastian et al., "Memory devices and applications for in-memory computing," *Nat.Nanotechnol.*, vol.15, no.7 pp.529-544, 2020, doi: 10.1038/s41565-020-0655-z.

[7] S. Li et al., "Drisa: A dram-based reconfigurable in-situ accelerator," in *MICRO-50'17*, New York, NY, USA, 2017, pp.288-301.

[8] V. Seshadri et al., "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *MICRO-50'17*, New York, NY, 2017, pp.273-287.

[9] K. Lee et al., "Bit parallel 6T SRAM in-memory computing with reconfigurable bit-precision," in *DAC*, Virtual Event, USA, 2020, pp.1-6.

[10]CJ. Jhang et al., "Challenges and trends of SRAM-based computing-in-memory for AI edge devices," *IEEE TCASI*, vol.68, no.5, pp.1773-1786, 2021, doi: 10.1109/TCSI.2021.3064189.

[11]DC. Kau et al., "A stackable cross point phase change memory," in *IEEE IEDM*, Baltimore, MD, USA, 2009, pp. 1-4.

[12]P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol.557, no.7792, pp.641-646, Jan 2020, doi: 10.1038/s41586-020-1942-4.

[13]S. Jung et al., "A crossbar array of magnetoresistive memory devices for in-memory computing," *Nature*, vol.601, pp.211-216, 2022, doi: 10.1038/s41586-021-04196-6.

[14]D. Ielmini et al., "In-memory computing with resistive switching devices," *Nat Electron*, vol.1, no.6, pp.333-343, 2018, doi: 10.1038/s41928-018-0092-2.

[15]X. Guo et al., "Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm NOR flash memory cells," *in IEEE CICC*, Austin, TX, USA, 2017, pp.1-4.

[16]L. Zhao et al., "A compute-in-memory architecture compatible with 3D NAND flash that parallelly activates multi-layers," in *DAC*, San Francisco, CA, USA, 2021, pp. 193-198.

[17]V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," *Nat Commun*., vol.11, no.1, pp.1-13, 2020, doi: 10.1038/s41467-020-16108-9

[18]A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *ISCA*, Seoul, Korea (South), 2016, pp. 14-26.

[19]E. Eleftheriou et al., "Deep learning acceleration based on in-memory computing," *IBM JRD*, vol.64, no.6, pp.7:1-7:16, 2019, doi: 10.1147/JRD.2019.2947008.

[20]A. Drebes et al., "TC-CIM: Empowering tensor comprehensions for computing-in-memory," in *IMPACT*, Bologna, Italy, 2020.

[21]M. Li et al., "The deep learning compiler: A comprehensive survey," *IEEE TPDS*, vol. huj32, no. 3, pp. 708-727, March 2021, doi: 10.1109/TPDS.2020.3030548.

[22]P. Barham et al., "Machine learning systems are stuck in a rut," in *HotOS*, New York, NY, USA, 2019, pp.177-183.

[23]F. Merrikh et al., "High-Performance Mixed-Signal Neurocomputing With Nanoscale Floating-Gate Memory Cell Arrays," *IEEE TNNLS*, vol. 29, no. 10, pp. 4782-4790, Oct 2018, doi: 10.1109/TNNLS.2017.2778940.

[24]R. Han *et al*., "A Novel Convolution Computing Paradigm Based on NOR Flash Array With High Computing Speed and Energy Efficiency," in *IEEE TCASI*, vol. 66, no. 5, pp. 1692-1703, May 2019, doi: 10.1109/TCSI.2018.2885574.

[25]X. Guo *et al*., "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology," in *IEDM*, San Francisco, CA, USA, 2017, pp. 6.5.1-6.5.4

[26]J. Yang et al., "Quantization networks," in CVPR, Long Beach, CA, USA, 2019, pp. 7300-7308.

[27]A Gholoami et al., "A Survey of Quantization Methods for Efficient Neural Network Inference," in *Low-Power Computer Vision*, 1st ed., Chapman and Hall//CRC, 2021, pp.291-326.

[28]M. Nagel et al., "A white paper on neural network quantization," 2021, *arXiv preprint arXiv:2106.08295*.

[29]B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *CVPR*, June 2018, pp. 2704-2713.

[30]Y. Li et al., "Additive Powers-of-Two Quantization: An Efficient Non-uniform Discretization for Neural Networks" In International Conference on Learning Representations. Addis Ababa, Ethiopia, Apr, 2020

[31]Y. Bengio et al., "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv preprint arXiv:1308.3432*.

[32]C. Mackin, et al. "Optimised weight programming for analogue memory-based deep neural networks," *Nature*, vol.13, no.1, pp.3765, 2020, doi: 10.1038/s41467-022-31405-1

[33]Hu. Yanxin, et al. "DCCRN: Deep complex convolution recurrent network for phase-aware speech enhancement," 2020, arXiv preprint arXiv:2008.00264.

[34]I.T. Union, "Wideband extension to recommendation p. 862 for the assessment of wideband telephone networks and speech codecs." in *International Telecommunication Union, Recommendation P*, vol. 862, 2007

[35]AW. Rix et al., "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *ICASSP*, Salt Lake City, UT, USA, 2001, pp. 749-752.

[36]E. Vincent et al., "Performance measurement in blind audio source separation," in IEEE transactions on audio, speech, and language processing, vol.14, 2006

[37]NVIDIA Corporation. "NVIDIA A100 Tensor Core GPU." https://www.nvidia.cn/data-center/a100 (Accessed September 27, 2023).