End-to-End Hardware-Software Co-Design Framework for In-Memory Computing Systems: Case Study based on A Commercial 40nm eFlash-based IMC SoC Chip

ABSTRACT

Although promising for Artificial Intelligence (AI) applications, current In-Memory Computing (IMC) technology faces a variety of challenges before mass production. One of the biggest obstacles is the lack of platforms and toolchains to bridge the gap between algorithms and IMC chips. This work proposes an end-to-end hardware-software co-design framework and the related toolchains for IMC systems. We further propose several key techniques to improve the performance, including: (a) An 8-bit hardware-friendly Quantification-Aware Training (QAT) approach to quantify the algorithm model 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 chip. (c) An efficient mapping strategy based on the Integer Linear Programming (ILP) approach to increase the space utilization of the IMC array. Finally, based on a commercial 40nm eFlash-based IMC SoC chip, we have verified the feasibility and demonstrated the performance with voice recognition, speech noise reduction and person detection applications as case studies. Our results show that the accuracy is over 95.7% (in quiet environment) and 87.27% (in white noise environment), and the false recognition rate is below 1 time per 24 hours for voice recognition; the Perceptual Evaluation of Speech Quality (PESQ) for noise reduction is improved by 21.53%; and the person detection accuracy is up to 97.80%.

KEYWORDS

in-memory computing, eFlash memory, toolchains, hardware-software co-design framework

1 Introduction

Nowadays, accompanied with the rapid development of algorithm 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, semantic segmentation etc. [1-3]. However, with the continuous expansion of the network scale, the NN accelerators based on traditional von Neuman architecture face big challenges owing to the physical separation of the memory and processor units. The frequent data movement between the memory and the processor induces high delay and energy consumption, called "memory wall" bottleneck [4]. In-Memory Computing (IMC) is a promising solution for breaking the "memory wall", where computational tasks are proceeded 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 massive parallel processing tasks. Such a structure is attractive for NN inference accelerations with trained weight data stored in the memory [6], as it can efficiently perform the core NN computing operations, i.e., multiply-and-accumulate (MAC) operations.

To date, numerous memory media are proposed to implement the IMC structure. Traditional memories, like Dynamic Random-Access Memory (DRAM), can use a charge-sharing mechanism between cells to implement in-memory logic operations [7,8]. However, its 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 problem 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 precision (8-bit per cell), high density and process maturity, and is already in production for IMC implementation [15,16]. Therefore, in this work, we adopt a commercial 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], because it needs a comprehensive cross-layer understanding of the algorithm models, compilation tools, circuits and other related domain-specific knowledges. Consequently, the portability and deployment cost are seriously problematic, which are the main obstacles that 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 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 a commercial 40nm eFlash-based IMC SoC chip (named IMC-SoC thereafter), we propose an end-to-end hardware-software co-design framework and the related toolchains 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. 1 shows the complete working flow of our tool, and the main innovations of this paper are summarized as following:

- An optimized hardware-friendly 8-bit Quantification 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 the framework.

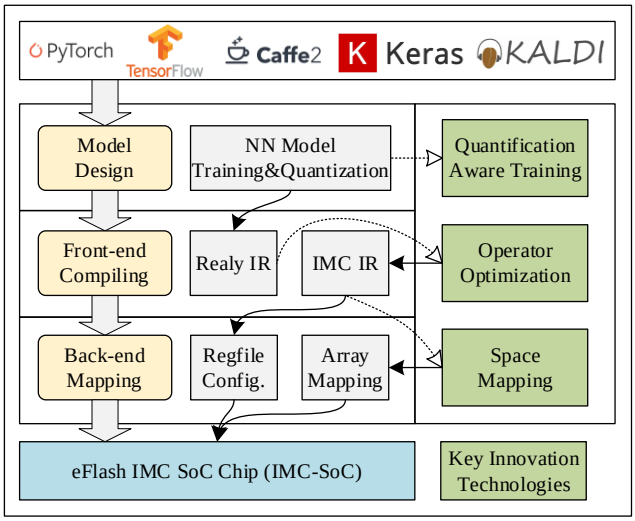


Fig. 1. Our proposed end-to-end hardware-software co-design framework and the key innovation technologies for IMC systems.

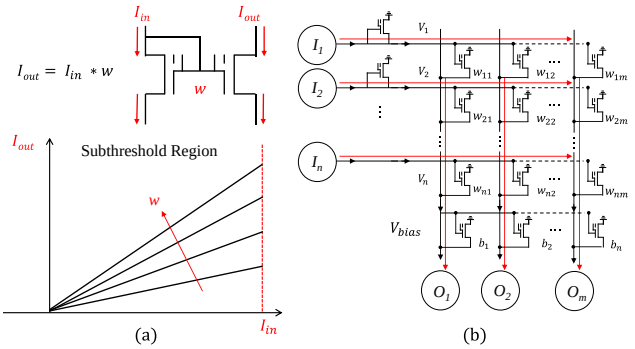


Fig. 2. (a) Computing principle of floating-gate transistors in subthreshold region. (b) Analog MAC operations realized in the eFlash array.

The remainder of this paper is organized as follows. Section 2 presents the structure of the IMC-SoC chip and our key techniques. Section 3 presents the experimental results. Finally, Section 4 concludes this work.

2 Our Proposed Framework

Fig. 1 depicts the architecture of our end-to-end hardware-software co-design platform for IMC systems. The input of the platform is from mature deep learning models, and the final output is the binary data flow that can be directly deployed on the IMC-SoC chip.

2.1 The Structure and Principle of IMC-SoC

The IMC-SoC chip used in this work is based on the 40nm eFlash memory, which can implement NN inference computation in extremely low-power consumption, since it uses the subthreshold operation mode (see Fig. 2(a)) for analog MAC operations. As shown in Fig. 2(b), the output current O_j in the j-th column can be expressed by the following formula [15]:

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

$$O_j = (\Sigma_i w_{ij} I_i) + I_{b_j} \quad (2)$$

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

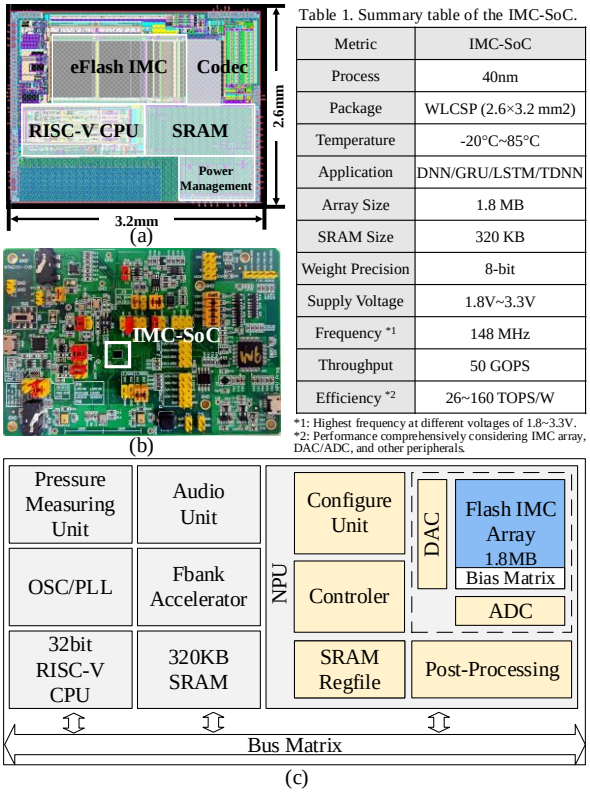


Table 1. Summary table of the IMC-SoC.

Metric	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 different voltages of 1.8~3.3V.
*2: Performance comprehensively considering IMC array, DAC/ADC, and other peripherals.

Fig. 3. The IMC-SoC chip. (a) Die photo. (b) The testing board. (c) The overall architecture of the IMC-SOC chip.

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.

With dedicated optimizations, each eFlash cell can store 8-bit weight and achieve 8-bit MAC precision, which can satisfy the accuracy demand of most edge NN applications. The IMC-SoC integrates this high-performance computing array with other control units, post-processing units, etc., to form a network processing unit (NPU) with computility of 50GOPS and energy efficiency of 26-160 TOPS/W. As shown in Fig. 3(c), the IMC-SoC embeds 32-bit RISC-V CPU core, 320KB on-chip SRAM, Fbank accelerators and audio unit for high performance audio processing, and various peripheral interfaces. Fig. 3(a) and Fig. 3(b) show the die photo and testing board, and Table 1 summarizes the attributes of the IMC-SoC chip. The detail of the chip is omitted here.

2.2 8-bit Quantification-Aware Training Method

Nowadays, deep learning models have advanced the frontiers in many fields, which are commonly trained with floating-point 32-bit (FP32) vectors. However, the SOTA IMC devices generally can support up to 8-bit precision. In this regard, quantization is the most used method to map the floating-point weights to fixed-point numbers with low bitwidths (e.g., 8-bit or 4-bit). In addition, faster inference and smaller memory footprint can be achieved with quantization methods [23,24].

In general, there are two types of quantization algorithms, including Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). PTQ requires no re-training and is more

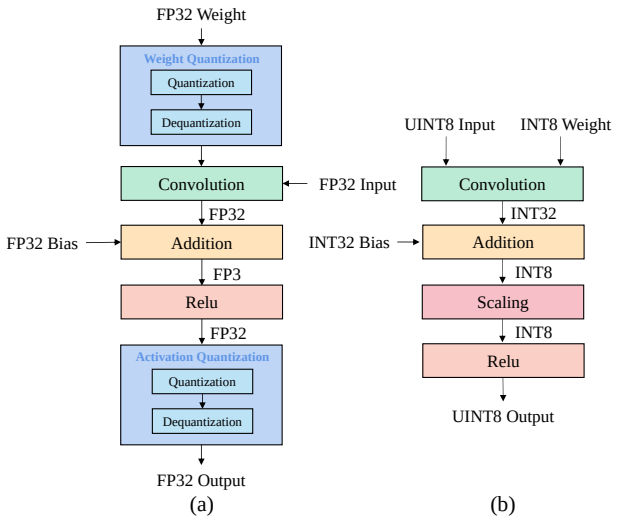


Fig. 4. The QAT method for (a) training and (b) inference flows.

lightweight, whereas it may introduce unacceptable errors incurred by low-bit quantization [25]. By contrast, the QAT method models the errors during the training process and offers more optimal solutions for bridging the gap for full-precision accuracy with low-bit precision. To exploit the hardware features of the IMC-SoC chip, we have made specific optimizations to the typical QAT method [26], as shown in Fig. 4. In our optimized QAT, to efficiently support the 8-bit precision of the IMC-SoC, firstly, we use the straight-through estimator [27] for approximation to avoid zero or undefined value while rounding for the gradients of the quantization operations. Secondly, we provide a suitable scaling factor G for NN inference given by Eq. (3), where G_{IN} , G_W and G_{Out} are the scaling factors for inputs, weights and outputs. The bias is quantified with the scaling factor G_{Bias} defined in Eq. (4):

$$G = \frac{G_{IN} G_W}{G_{Out}} \tag{3}$$

$$G_{Bias} = G_{IN} G_W \tag{4}$$

With the above optimizations, we can deploy various NN models on the IMC-SoC chip with negligible accuracy loss.

2.3 IMC Operator Optimization

In our IMC-SoC chip, the typical matrix formula for NN computing is defined as:

$$Y = \sigma\left(\frac{WX+b}{G}\right) \tag{5}$$

where W, X, b, and G are corresponding to 8-bit weights, 8-bit unsigned input data, bias parameter and gain value in a matrix operation respectively, σ is the activation function, Y represents the final 8-bit calculation results. Through this computational paradigm, multiple networks, such as DNN, TDNN and LSTM, can run on our IMC-SoC chip efficiently.

However, when executing 8-bit MAC operations, there may exist accuracy loss of low-bit data because of the process variations and noises in the eFlash cells [15]. Thus, while transferring relay Intermediate Representation (IR) [21] to customized IMC IR (as shown in Fig. 1), we have designed several operator optimization methods to reduce the impact of low-bit accuracy loss on overall MAC operations. Meanwhile, processing of the negative inputs and convolution acceleration are handled at this stage.

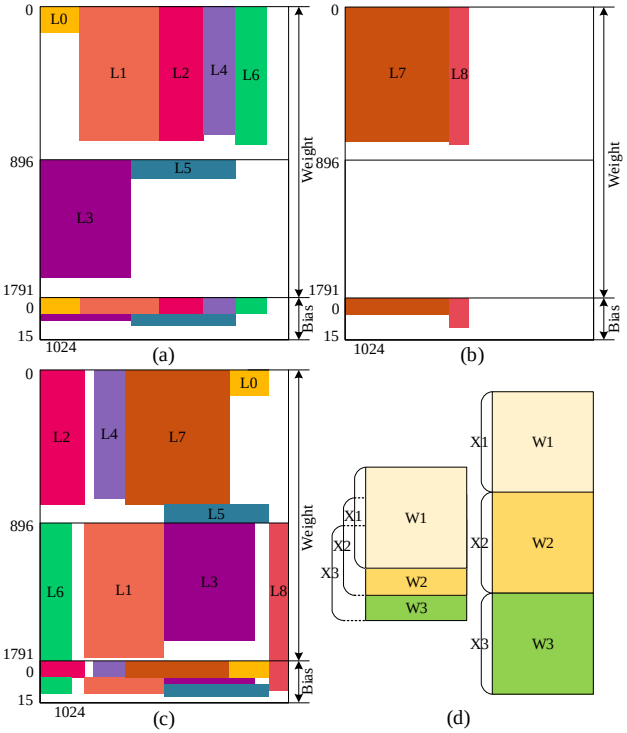


Fig. 5. Weights and bias distributions of the DCCRN with (a)-(b) traditional sequential mapping methods and (c) our proposed ILP mapping method. (d) Sketch map of the multi-point convolution optimization, in which the weight matrix is divided into multiple copies.

(a) Amplification Strategy of $W/X/b$

This method is based on the computing formula in Eq. (5). If W, X and b are amplified, the original theoretical calculation results can be obtained by adjusting the value of the G accordingly. On this premise, through magnifying the parameters by 2^n to shift the low-bit data to the high-bit region, the resulting accuracy will be greatly improved since the computation of the low-bit data are eliminated.

(b) Replication Strategy of W

This method replicates the weight data in the IMC memory array and performs multiple MAC operations with the same parameters. By averaging these results, the final error can be minimized by reducing the effects of process variations and noises.

(c) Positive and Negative Optimizations

This method aims for processing signed paraments. Since the array can only process unsigned MAC operations, special treatments are required for operators with negative parameters. The first method is to add a positive number to the negative inputs to ensure that all the inputs become non-negative, and let the bias subtract the product of the positive number and the weights of the current layer. The second method is to invert the negative part of the signed input and expand it with the positive part into a new unsigned input. Nevertheless, the former method may lead to excessive bias value, while the latter one may occupy more chip resource.

(d) Multi-point Convolution Optimizations

The sketch map of this method is shown in Fig. 5(d), where W and X represent the weight matrix and input vector in the convolution operations. In the multi-point mode, the weight matrix is divided into multiple copies according to the computing stride, and the

computations of $W1$, $W2$ and $W3$ that normally take three cycles can be done at once in this mode. By slicing and replicating the weight matrix in the array, multi-layer convolution operations can be accelerated with parallel computing.

In different cases, we can employ specified combination of the above optimization strategies to improve the performance of the network deployment on the IMC-SoC chip.

2.4 Integer Linear Programming Space Mapping

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-SoC memory array for completing the MAC operations. 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. For example, with regard to a typical multi-point Deep Complex Convolution Recurrent Network (DCCRN) for audio denoising, different allocation effects are shown in Fig. 5. Squares with different colors represent the weights and bias data in different layers of the DCCRN.

In traditional sequential mapping methods, weights are loaded into the whole memory array (1792×896 in our IMC-SoC chip) following the input order of the layers. When the weights cannot be loaded in the current free space, the system will wait to load the data until the existing matrix is processed. As shown in Fig. 3(a)-(b), the entire DCCRN needs to be separately loaded into the array by twice and the highest memory space utilization for these two loads is only 53.6%.

For further improving the efficiency of the hardware resource, we propose an Integer Linear Programming (ILP) space mapping strategy to convert the limitations of the bias and weight data into the constraints and objectives of the ILP problem. The constraints for each layer mainly include: (a) The coordinate of the weights cannot exceed the range of the memory array; (b) The weights and bias cannot be stored in different regions (across line 896); (c) The column addresses of the bias and weights are equal; (d) The weights should occupy as little range of the column as possible. By using Google's constraint programming satisfiability (CP-SAT) solver, we can generate the CP models of variables and constraints and obtain the optimal solutions of the weight mapping. As shown in Fig. 5(c), the whole weights and bias data of the DCCRN can be allocated into the memory array at once, and the memory space utilization can be raised to 71.7%.

3 Case Study and Results

3.1 Toolchain Setup

This section focuses on the complete workflow of our end-to-end framework. The toolchains in the framework, which are used to map NN models to the IMC-SoC chip, and generate corresponding configuration and programmed files, can be adopted to applications. As shown in Fig. 1, multiple mature deep learning frameworks, such as PyTorch, TensorFlow, Caffe2, Keras and Kaldi, etc., are supported by the front-end design. Firstly, through 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, specified combination of the operator optimization strategies (presented in section 2.3) are implemented to improve the computing precision and to generate the IMC-IR that is more suitable for the IMC-SoC. Afterwards, 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 2. 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

Table 3. Accuracy of voice recognition.

Scenario \ Environment		Pure	White Noise (10dB)	Pink Noise (10dB)
English Commands*1	(11 words, 75dB)	98.18%	87.27%	90.18%
Video Control	(12 words, 75dB)	98.70%	92.70%	97.70%
Smart Speaker*2	(30 words, 75dB)	98.12%	93.54%	97.50%
Desktop Control	(36 words, 75dB)	95.70%	89.60%	88.90%

*1 : The other words for recognition are Chinese.
*2 : False recognition rate of the wake word is below 1 time per 24 hours.

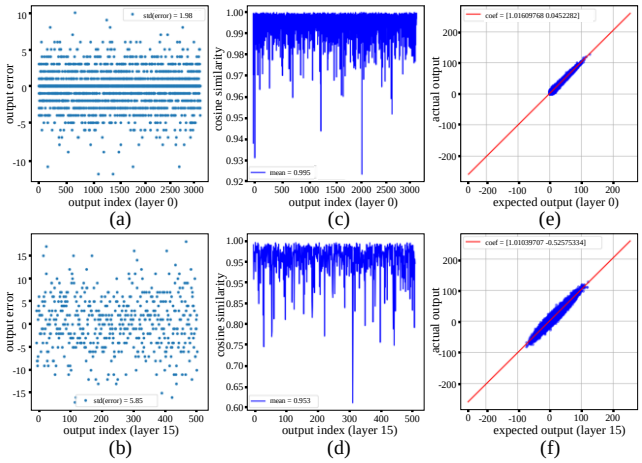


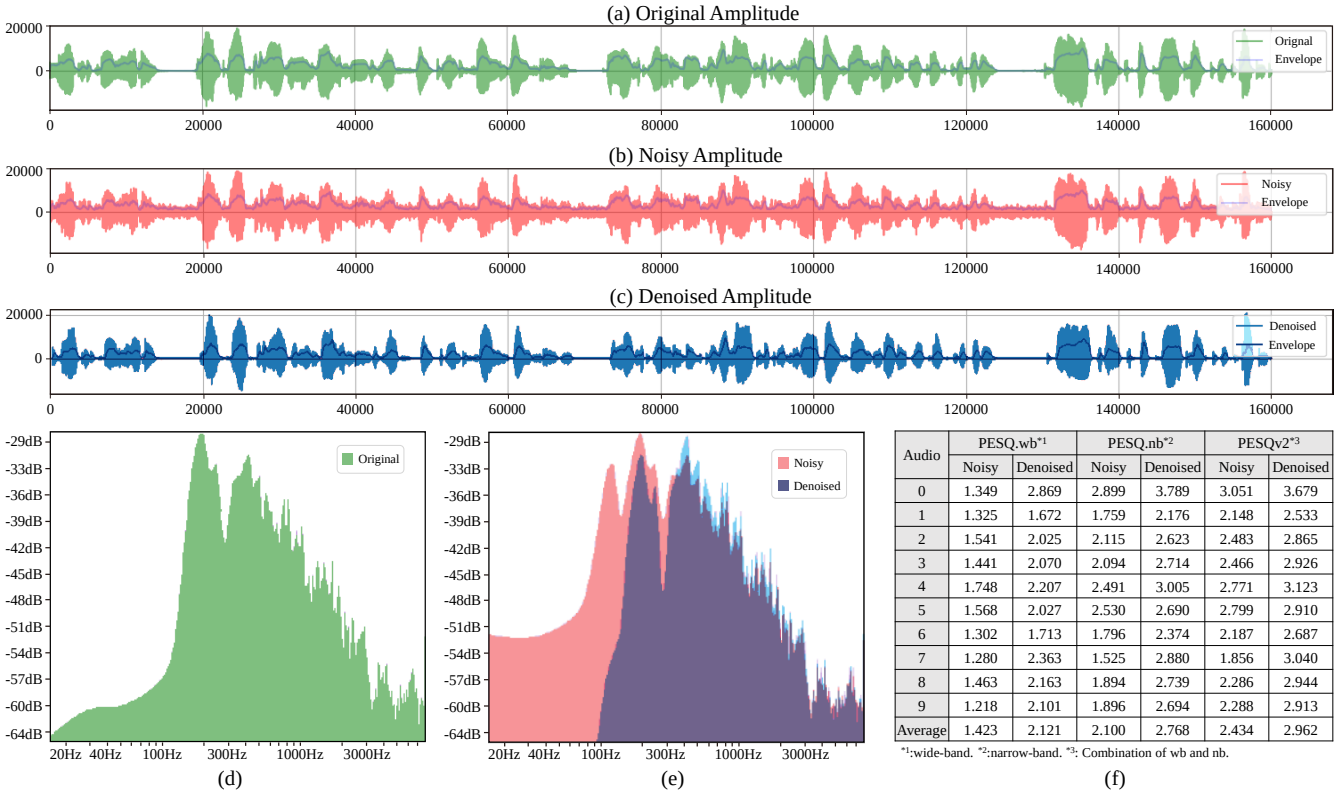
Fig. 6. Error analysis of the actual outputs from the IMC-SoC chip and expected output data in the first layer and last layer of the voice recognition network. (a)-(b) Output standard deviation of the errors and (c)-(d) cosine similarity of each column in the IMC-SoC array. (e)(f) Linear fitting curves of the actual and the expected outputs of the IMC-SoC chip.

3.2 Feasibility Verification

We take a voice recognition network (5 layers) as an example to verify the feasibility of our framework and the related tools. We analyze the theoretical weights from the NN model and the actual read-back programmed weights from the IMC-SoC chip. As shown in Table 2, the average standard deviation (std) and 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 meets the requirements to accurately deploy the NN weights on the IMC-SoC chip. Based on this framework and toolchain, we further verify the performance (mainly accuracy) with voice recognition, speech noise reduction and person detection as case studies.

3.3 Voice Recognition

As for end-side devices, voice recognition has gradually become an indispensable function. In this case, we deployed a 16-layer voice recognition model on the IMC-SoC chip. As shown in Table 3, we tested the phonetic keywords (75 dB) of four different scenarios. Without noisy interference, the worst recognition rate is above 95.70%. With the addition of white noise (about 10dB) and pink noise (about 10dB), the worst recognition rates are 87.27% and 88.90% respectively. In addition, the false recognition rate for



Audio	PESQ.wb*1		PESQ.nb*2		PESQv2*3	
	Noisy	Denoised	Noisy	Denoised	Noisy	Denoised
0	1.349	2.869	2.899	3.789	3.051	3.679
1	1.325	1.672	1.759	2.176	2.148	2.533
2	1.541	2.025	2.115	2.623	2.483	2.865
3	1.441	2.070	2.094	2.714	2.466	2.926
4	1.748	2.207	2.491	3.005	2.771	3.123
5	1.568	2.027	2.530	2.690	2.799	2.910
6	1.302	1.713	1.796	2.374	2.187	2.687
7	1.280	2.363	1.525	2.880	1.856	3.040
8	1.463	2.163	1.894	2.739	2.286	2.944
9	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.

(f)

Fig. 7. Amplitude diagram of the (a) original, (b) noisy and (c) denoised audios. Frequency diagram of the (d) original, (e) noisy (red) and denoised (blue) audios. (f) Comparison of the PESQ scores for ten test audios before and after denoise processing.

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.

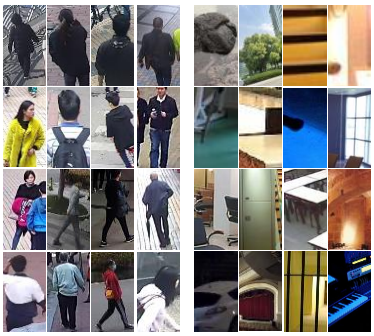
During the inference process, we obtained the actual outputs of each layer of the network through real-time precision analysis tools and compared it with the expected outputs from the software. Taking the first layer and the last layer of the network as examples (see Fig. 6), we analyzed the error between the actual values and the expected outputs from the IMC-SoC chip for each input frame of the voice. In Fig. 6(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. 6(c), which proves the accuracy for single-layer MAC operations with the IMC-SoC chip and our framework. However, during the processing of a multi-layer network, the errors will be accumulated layer-by-layer inevitably. We can find from the last-layer error analysis in Fig. 6(b) and Fig. (d), the accuracy of the system has a little decrease. Nevertheless, we can find from Fig. 6(e)-(f) that the slope of the linear fit curves is close to one. Therefore, the accuracy of the system can be further improved by adding noise reduction techniques.

3.4 Speech Noise Reduction

In the field of audio processing, noise reduction has been widely used and has been proved as an effective method. In this case, we deployed a DCCRN model with 9 full connected layers and one long short-term memory layer (as a case study) on our IMC-SoC chip to denoise different audios. 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. 7(a)-(c), the red noisy spectrum has 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. 7(d)-(e), we can clearly find that the amount of low-frequency noise was added to the original audio and was eliminated after the noise reduction processing. Furthermore, through the Perceptual Evaluation of Speech Quality (PESQ) parameter (which is a parameter used to measure the quality of the audio and the value is generally from -0.5 to 4.5, the bigger the better) [24,25], we can quantitatively evaluate the performance of our techniques. In Fig. 7(f), 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 in Fig. 7(f) combined the wide-band and narrow-band denoise effects) is improved by 0.524 (by 21.53%) compared to the noisy audio.

3.5 Person Detection

As can be seen from Table 1, 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. Take person detection as a case study, we designed a customized image classification network with ten convolution layers and one full connected layer to show the feasibility. For each input picture, we



Resize to 64×64

Image Classification Network (CNN)

Sample	Accuracy
Positive	97.99%
Negative	97.61%
Overall	97.80%

Fig. 8. Accuracy of person detection with customized image classification network implemented on the IMC-SoC chip.

resized the 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%.

4 Conclusion

We present an end-to-end hardware-software co-design framework and the related toolchains for IMC systems. Firstly, we adopt an 8-bit hardware-friendly QAT method to train high-accuracy NN models and to achieve fast inference speed and small memory footprint on the IMC chip. Secondly, we propose several operator optimizations to minimize the accuracy loss and to accelerate convolutional computation. Thirdly, to make better use of the memory space, we convert the weight mapping problem into ILP for solving and greatly improve the hardware utilization efficiency. Through 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 95.7% (in quiet environment) and 87.27% (in white noise environment), and the false recognition rate is below 1 time per 24 hours. Then, 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 the person detection accuracy can be up to 97.8%. Our work bridges the gap between NN algorithms and IMC chips, and can be extended for other IMC chips.

REFERENCES

[1] H. Zhang et al., "CP-SRAM: charge-pulsation SRAM marco for ultra-high energy-efficiency computing-in-memory," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022.

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

[3] A. Krizhevsky et al., "ImageNet classification with deep convolutional neural networks," *Communications of The ACM*, vol.60, no.6, pp. 84-90, 2017.

[4] K. Lee et al., "A Charge-Sharing based 8T SRAM In-Memory Computing for Edge DNN Acceleration," in *Proceedings of the 58th ACM/IEEE Design Automation Conference*, 2021.

[5] N. Verma et al., "In-Memory Computing: Advances and Prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 43-55, 2019.

[6] A. Sebastian et al., "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol.15, no.7 pp.529-544, 2020.

[7] S. Li et al., "Drisa: A dram-based reconfigurable in-situ accelerator," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.288-301, 2017.

[8] V. Seshadri et al., "Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology," in *50th Annual IEEE/ACM International Symposium on Microarchitecture*, 2017.

[9] K. Lee et al., "Bit parallel 6T SRAM in-memory computing with reconfigurable bit-precision," in *Proceedings of the 57th ACM/IEEE Design Automation Conference*, 2020.

[10] CJ. Jhang et al., "Challenges and trends of SRAM-based computing-in-memory for AI edge devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol.68, no.5, pp.1773-1786, 2021.

[11] DC. Kau et al., "A stackable cross point phase change memory," in *IEEE International Electron Devices Meeting*, pp.1-4, 2009.

[12] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol.557, no.7792, pp.641-646, 2020.

[13] S. Jung et al., "A crossbar array of magnetoresistive memory devices for in-memory computing," *Nature*, 2022.

[14] D. Ielmini et al., "In-memory computing with resistive switching devices," *Nature electronics*, vol.1, no.6, pp.333-343, 2018.

[15] X. Guo et al., "Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm NOR flash memory cells," *IEEE Custom Integrated Circuits Conference*, pp.1-4, 2017.

[16] L. Zhao et al., "A compute-in-memory architecture compatible with 3D NAND flash that parallelly activates multi-layers," in *58th ACM/IEEE Design Automation Conference*, pp.193-198, 2021.

[17] V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol.11, no.1, pp.1-13, 2020.

[18] A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol.44, no.3, pp.14-26, 2016.

[19] E. Eleftheriou et al., "Deep learning acceleration based on in-memory computing," *IBM Journal of Research and Development*, 2019.

[20] A. Drebes et al., "TC-CIM: Empowering tensor comprehensions for computing-in-memory," in *International Workshop on Polyhedral Compilation Techniques*, 2020.

[21] M. Li et al., "The deep learning compiler: A comprehensive survey," *IEEE Transactions on Parallel and Distributed Systems*, vol.32, no.3, pp.708-727, 2020.

[22] P. Barham et al., "Machine learning systems are stuck in a rut," *Proceedings of the Workshop on Hot Topics in Operating Systems*, no.7, pp.177-183, 2019.

[23] J. Yang et al., "Quantization networks," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[24] A Gholoami et al., "A Survey of Quantization Methods for Efficient Neural Network Inference," *Low-Power Computer Vision*, pp.291-326, 2021.

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

[26] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

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

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

[29] AW Rix et al., "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp.749-752, 2001.