Compressed CNN for Inferring Rapid RF Fingerprints using Memristor Crossbar Array

Zhaoxi Li*, Jianbin Huang†, Michael B. Jiang‡, and Kang Jun Bai§
*University of Maryland Baltimore County, Baltimore, MD
†San Francisco State University, San Francisco, CA
‡University of Illinois Urbana-Champaign, Champaign, IL
§Air Force Research Laboratory Information Directorate, Rome, NY
Email: *joshl1@umbc.edu, †jhuang19@sfsu.edu, ‡mbjiang2@illinois.edu, §kang.jun.bai@us.af.mil

***Abstract*—In response to the growing demand of authenticating wireless devices in a large pool of internet of things (IoT) network, a convolutional neural network (CNN) has proposed to identify Wi-Fi transmitters based on their radio frequency (RF) fingerprints – raw in-phase and quadrature (I/Q) components – with decent accuracy. Meanwhile, the in-memory operator using memristor crossbar array for analog vector-matrix multiplication (VMM) has demonstrated its high energy efficiency and small form factor when implemented deep learning models liked CNN. In this work, we showcase a compressed CNN using average pooling methodology to reduce model parameters and required hardware resources for the inference operation. Specifically, by linearizing pooling layers with averaging approach, trained convolutional and pooling layers can be integrated into one single matrix – via matrix-matrix multiplication – which can be realized by a memristor crossbar array for efficient and rapid inference operation. Prototype and proof-of-concept demo were made for authentication of Wi-Fi transmitters based upon their raw RF fingerprints. With merely 2 crossbar arrays made of 3-bit memristors alongside the temporal encoding mode, our model demonstrates 91% accuracy, making such a neural network a good candidate when implemented in power-limited mobile edge devices.**

***Index Terms*—compressed convolutional neural network, averaging pooling, in-memory computing, memristor, radio frequency fingerprints.**

1. Introduction

Internet of Things (IoT) network has immense potential for a variety of applications, such as medical diagnosis, environmental monitoring, smart cities, etc. [1]. Accordingly, authenticating IoT devices becomes a critical measure to ensure the security of large IoT deployment [2]. In this manner, radio frequency (RF) fingerprints of wireless transmitters can be used as an identification feature, as each has unique circuit artifacts.

State-of-the-art literature has demonstrated the success of convolutional neural networks (CNNs) in authenticating Wi-Fi transmitters based on their RF fingerprints by efficiently capturing spatial features within signals [3]. In addition to CNNs, recurrent neural networks (RNNs) [4] and their variants, including long short-term memory (LSTM) [5] and gated recurrent units (GRUs) [6], are well suited for processing the sequential nature of RF signals, enabling them to handle temporal dependencies over time. Afterward, the echo state network (ESN), a specialized RNN, also shows promise in authenticating RF signal by utilizing a sparsely connected reservoir with fixed internal states to enable efficient processing of temporal dynamics while significantly simplifying the training operation [7]–[9]. However, as large-scale IoT deployments require rapid and efficient inference, it becomes important to consider architectures that strike a balance between computational efficiency and accuracy.

In recent years, in-memory computing (IMC) interfacing with the resistive random-access memory (RRAM) has demonstrated its high energy efficiency and small form factor when building deep learning models [10]. Benefited by the ohmic and parallel natures of RRAM crossbar array, vector-matrix multiplication (VMM) is executed in analog domain instantaneously without the limitation of memory wall, and yet, the architecture of analog VMM could be challenging in implementing CNN directly [11].

In this work, a pre-trained CNN is simplified and compressed for VMM operations by leveraging the linear characteristics of convolution and pooling. When implemented in inference, fewer VMM operations are needed to accomplish cognitive tasks, *e.g.*, authentication of RF signals. In short, our compressed CNN significantly reduces the model size and computational latency while increasing the energy efficiency, making such a model a good candidate to be widely deployed for edge computing.

2. Compressed CNN with Average Pooling

In this experiment, RF fingerprints were extracted from raw in-phase and quadrature (I/Q) components of 16 Wi-Fi transmitters with a sampling frequency of $5\ \mathrm{MS/s}$ as depicted in [3], in which 128 pairs of IQ samples were used at a time to differentiate 16 transmitters. A conventional CNN, as illustrated in Fig. 1, has performed well to identify RF

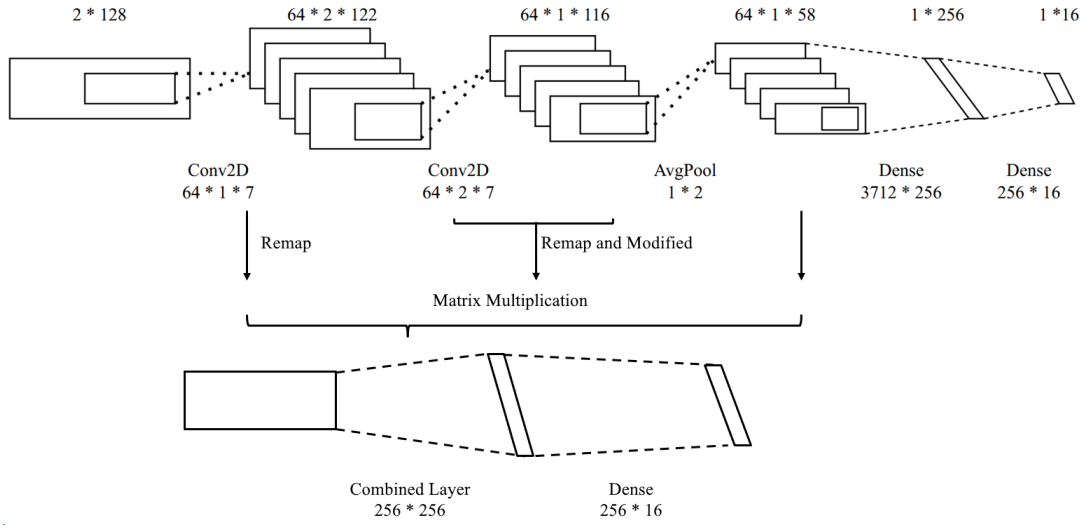

Figure 1. An overview of combining convolutions and average pooling into a single VMM operation.

signals, especially in static channel condition. The baseline here was made of 64 $[1 \times 7]$ filters for separated I and Q channels as the first layer, 64 $[2 \times 7]$ filters for concatenated feature maps as the second layer, followed by an average pooling layer with a stride of 2. The resulting outcomes were then flatten into an one-dimensional (1D) array and readout by a fully-connected (FC) classifier. This baseline model was simulated on a general purpose computer with 32-bit floating point number and reached an accuracy of 99% when tested with 16 transmitters.

To further simplified the trained CNN for efficiency, the matrix convolution was first turned into a series of two-dimensional (2D) VMM and then integrated into a single matrix using the associative properties of linear operation. Thereby, when used in inference, only VMM operations are needed. Detailed compression method with the RF identification task is discussed in the following.

Fig. 2 demonstrates the conversion from matrix convolution into 2D VMM. To be specific, inputs with separated I and Q channels are first reshaped into a $[1 \times 256]$ row vector while each $[1 \times 7]$ convolutional filter is turned into a column vector. This column vector is then placed based on the spatial locations of input elements for each sliding window within the matrix convolution. To better support the computation, zeros are filled into the filter vector as weight values in convolutional filters stair down as depicted in Fig. 2. By doing so, the number of column is calculated as $2 \cdot (128 - 7 + 1)$, and thus, the convolution with one filter results in a 2D VMM between a row vector of $[1 \times 256]$ and a weight matrix of $[256 \times 244]$. To accommodate multiple output channels, weight matrices of individual convolutional filter can be concatenated horizontally. With 64 $[1 \times 7]$ convolutional filters, the size of weight matrix is expanded to $[256 \times 15,616]$.

The second convolutional layer contains 64 $[2 \times 7]$ filters for concatenated features. As denoted in Fig. 3, each input row vector is $[1 \times 15,616]$ from the previous layer, while

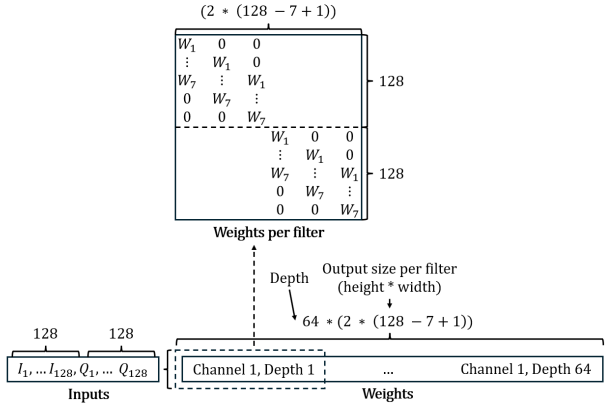

Figure 2. Convolution with 2D VMM featuring multiple output channels.

each convolutional filter has matrix size of $[244 \times 116]$, where the number of columns is calculated as $(122 - 7 + 1)$. With a similar strategy when accommodating more output channels, multiple input vectors can be concatenated vertically while weight matrices are concatenated horizontally, and thus, the resulting weight matrix has a shape of $[15,616 \times 7,424]$.

Organizing the weight matrix is essential to ensure the alignment between inputs and outputs across layers, in which each row of the weight matrix must correspond to its corresponding input and is filled systematically. To be specific, the weight matrix is initialized hierarchically, processing each filter for input channels followed by the output one. Reshaping is applied to input and output channels separately to ensure proper alignment. Such a method guarantees the reshaping process for input channels does not interfere with that of the output one.

Afterward, an average pooling layer with a stride of 2 is applied to calculate the average value of 2 adjacent columns of preceding convolutional layer in groups, and thus, the

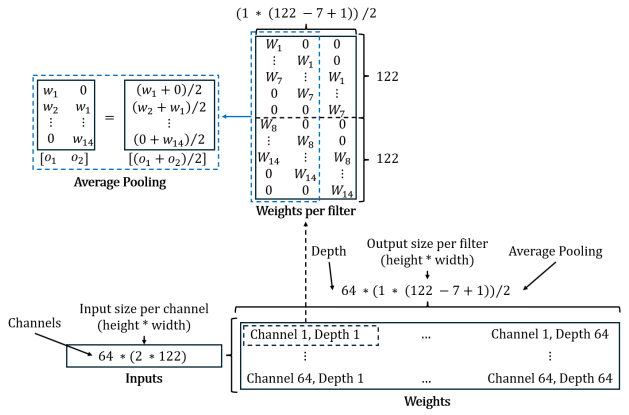

Figure 3. Convolution with 2D layer featuring multiple input channels and pooling.

size of resulting weight matrix is reduced by half as of $[15,616 \times 3,712]$. Until here, only linear operation is used to convert matrix convolution into VMMs and reduce the size of resulting outcome. In the inference operation, the entire convolution alongside the subsequent average pooling and flatten layer (with 256 neurons in this experiment) can be compressed into a single matrix, in the size of $[256 \times 256]$, using the associative properties of linear operation. For instance:

- Matrix-matrix multiplication between two convolutional layers is calculated as $[256 \times 15,616][15,616 \times 7,424] = [256 \times 7,424]$,
- Size reduction through the average pooling as of $[256 \times 3,712]$,
- Matrix-matrix multiplication between preceding and flatten layer is calculated as $[256 \times 3,712][3,712 \times 256] = [256 \times 256]$.

Together with a FC classifier, the entire inference operation is compressed into two VMMs – $[256 \times 256]$ and $[256 \times 16]$ – with a nonlinear activation to authenticate 16 RF signals. Such a compression methodology dramatically reduces the model size, required hardware resources, computational latency, and power consumption when implemented in edge devices.

3. Circuit & System Implementation

RRAM crossbar array for IMC promises an energy-efficient and compact analog VMM, in a way by leveraging Ohm's Law to execute operations [12]. By compressing the CNN inference operation with the aforementioned algorithm, the process of building the neural network onto RRAM crossbar array becomes significantly more efficient and unambiguous. Here, to differentiate 16 RF signals, 2 RRAM crossbar arrays of sizes 256×256 and 256×16 were used to build up our compressed CNN. To further reduce power consumption, input tensors and weight values were quantized into 4-bit and 3-bit, respectively.

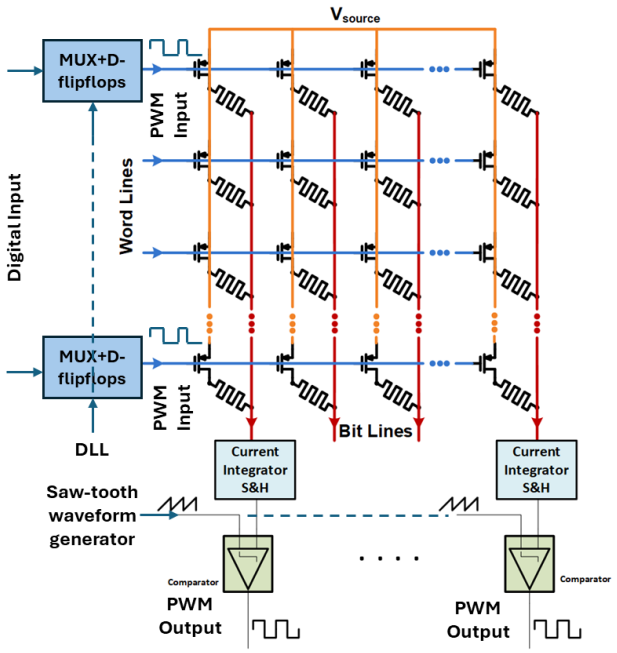

Figure 4. An overview of analog VMM operator using RRAM crossbar array with PWM signal as computing variables. MUX and DFF at each WL are to cover digital inputs into PWM signals. Current integrator and comparator at each BL are to cover the resulted analog current back to PWM signals for the following stage. A single DLL and sawtooth waveform generator are shared by the entire system for efficiency.

In this system, the computing variable is represented by the duty cycle of constant-amplitude pulses at a fixed clock frequency [13]. As showcased in Fig. 4, the accumulated current at each bit-line (BL) produces the weighted sum by multiplying an array of input pulse-width modulated (PWM) signals and their corresponding memristor's conductance, which is then normalized and integrated over a clock period via a capacitor by a current amplifier (served as a sample and hold, S/H). This sampled voltage is eventually compared to a reference signal made by the global saw-tooth waveform generator, and the duty cycle of comparator's output pulses is proportional to the VMM result.

3.1. Write Circuitry

The write circuitry consists of local and global modules: (1) a selector made of multiplexer (MUX) and a phase detector (PD) made of D-type flip flops (DFFs) as the local circuit at each word-line (WL) that interfaces with the digital input, and (2) a global delay locked loop (DLL) shared by all WLs with a set of 15 delayed clocks [13].

As depict in Fig. 5, the selector picks 1 of the 15 equally delayed clocks in according to the quantized 4-bit digital input from the select-line (SL). By comparing the delayed clock and the master reference clock via PD, the time period between individual rising edge of separated signals are measured, turning the digital input into an analog PWM signal.

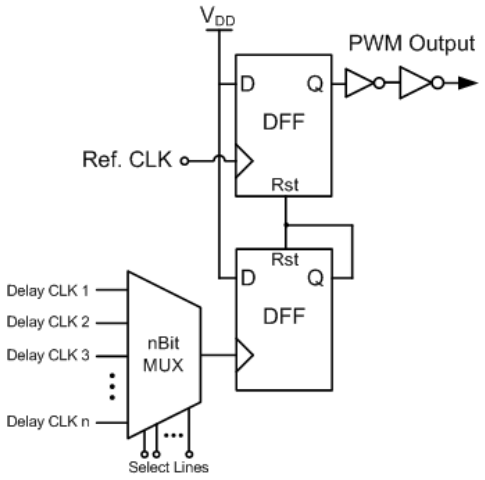

Figure 5. Simplify schematic of write circuitry.

3.2. RRAM with Bi-directional Current

Here, individual RRAM is made to encode either positive or negative weight values, in a way by controlling the direction of current flow through the RRAM [14], removing the necessity and power/area overhead from peripherals. As illustrated in Fig. 6, the RRAM is designated as positive weight when the reference voltage, V_H, is enabled and set to be 400 mV while the voltage at BL is pinned at 200 mV. By contrast, the RRAM is designated as negative weight when V_H is disenabled while V_L is pinned to ground. In short, a forward-biased current through the RRAM reads a positive weight value while a reverse-biased current reads a negative one. For other RRAMs that are read as zero, both V_H and V_L are disenabled for power efficiency and to eliminate sneak paths [15]. By doing so, currents resulted by either positive or negative weight values from each individual RRAM subtract each other along the BL, such that $I_{total} = \Sigma(I_{positive}) - \Sigma(I_{negative})$, and thus, individual RRAM, rather than a pair of 1-transistor-1-RRAM (1T1R)

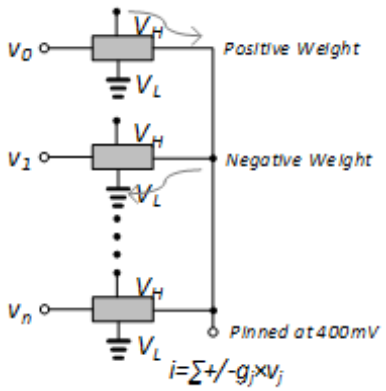

Figure 6. Illustration of RRAM with either positive or negative weight value in terms of current direction.

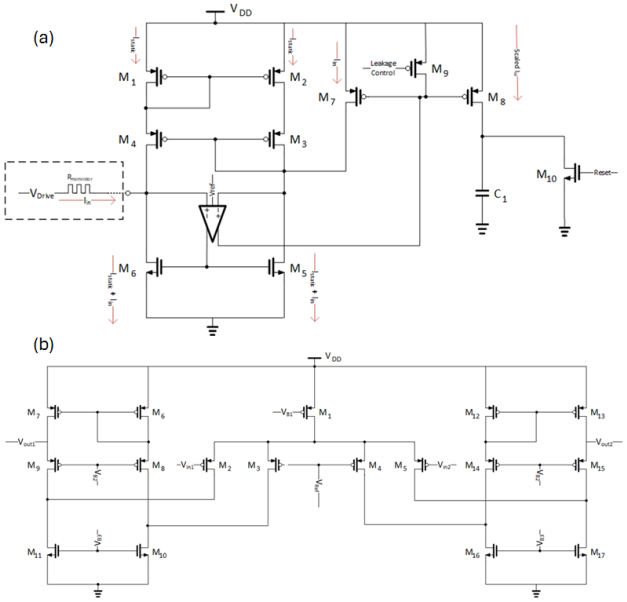

Figure 7. Circuit implementation of (a) current amplifier and S/H, and (b) op-amp used in current amplifier.

with current subtractor, can be deployed to support the computation of deep learning workloads.

3.3. Read Circuitry

The read circuitry also consists of local and global modules: (1) a current amplifier and a comparator at each BL as the local circuit, and (2) a sawtooth waveform generator shared by all BLs as the global circuit.

The design methodology of current amplifier is depicted in Fig. 7(a). Transistors M_1 and M_2 form a current mirror with 120 μA static current. The operational amplifier (op-amp) pins the potential of BL at a fixed voltage in according to V_{REF} (*e.g.,* 400 mV in this design). The accumulated current from BL and the static current are then integrated and mirrored by transistors M_5 to M_6. Once the current mirror is balanced, accumulated current from BL is duplicated to M_7, and then scaled down by a factor that equals to the number of WLs. Such a normalized current eventually charges up a sensing capacitor, C_1, over 200 ns, in which the resultant voltage at C_1 is proportional to the VMM from the corresponding BL and turned into PWM signals for the next stage by comparing to a sawtooth waveform.

To force the driving voltage of M_5 and M_6 is pinned to V_{ref}, the op-amp used current amplifier is made of two supplemental single-stage folded cascode op-amps, as showcased in Fig. 7(b). As they share the same V_{ref}, supplemental op-amps can be integrated in the layout for area efficiency.

4. Experimental Results

The mathematical model of our compressed CNN was first simulated in software and the performance metrics

TABLE 1. COMPARISON ON RF AUTHENTICATION WITH THE STATE-OF-THE-ART CNN MODELS

	[3]	[16]	[17]	[18]	This Work	
					Baseline	Combined
Input Size	2x256	2x128				
Structure	8x Conv. 128x[1x7] Conv. 128x[1x5] MaxPool Dense (256) Dense (64) Dense (16)	Conv. 128x[3x3] MaxPool 5x Conv. 256x[3x3] MaxPool Dense (256) Dense (64) Dense (17)	Conv. 50x[1x7] Conv. 50x[2x7] Dense (256) Dense (80) Dense (16)	Conv. 50x[1x3] MaxPool Conv. 50x[2x3] MaxPool Dense (256) Dense (4)	Conv. 64x[1x7] Conv. 64x[2x7] AvgPool Dense (256) Dense (16)	VMM [256x256] Dense (16)
# of Classes	16	17	16	4	16	
# of Parameters	1,102,096	2,936,721	1,542,362	400,534	1,012,160	69,120
# of MACs	111,566,383	109,102,144	5,651,960	1,337,824	7,715,584	69,120
Accuracy	98.70%	93.40%	96.00%	98.00%	99.20%	99.20%

are summarized in Table 1. The number of parameters and multiple-and-accumulate (MAC) operations were calculated with PyTorch-OpCounter [19]. It can be observed that our compressed CNN achieves significant reduction in model parameters and required MAC operations, with 14.64x and 111.63x improvement respectively as compared to our baseline model while maintaining an identical accuracy of 99.20%. As compared to the next-best model from the state-of-the-art, our compressed CNN offers up to 15.94x and 1614.10x reduction on model parameters and MAC operations respectively.

Our model was then simulated in hardware using two RRAM crossbar arrays in the Cadence Virtuoso platform with a custom $65\ \mathrm{nm}$ CMOS/RRAM technology node, sizing with 256×256 and 256×16. The resistance of each RRAM was quantized into 3-bit, ranging from $10\ \mathrm{k}\Omega$ to $25\ \mathrm{k}\Omega$ with a high impedance state at $150\ \mathrm{k}\Omega$ [20]. The simulated system also included dedicated registers to control the direction of current flow at each RRAM. The reference voltage at each BL was pinned at $200\ \mathrm{mV}$ while the current reduction factor of the read circuitry was set according to the number of WLs, *i.e.,* 256 for both layers. The supply voltage was kept at $1.2\ \mathrm{V}$ and the frequency for input quantization was fixed at $50\ \mathrm{MHz}$. With 8,000 RF data streams, our hardware model achieved an overall accuracy of 91.60%, yielding 7.6 percentage points reduction as compared to our software model.

5. Conclusion

In this work, we showcase a compressed CNN to authenticate wireless transmitters based on their RF fingerprints – down-converted IQ components. In our model, an average pooling, instead of the conventional max pooling, is used without nonlinear activation in between. Such a linearized CNN can be compressed into simple VMMs by the linear associative law. In addition to the model compression, we also demonstrate the high computational efficiency when implemented with IMC technology using RRAM crossbar arrays. With the introduced fully analog circuitry, the RRAM crossbar array has efficiently performed analog VMM using PWM signals, offloading the necessity of power-hungry data conversion between analog and digital. When tested with RF data streams, our software model demonstrates as high as 15.94x and 1614.10x reduction on model parameters and required MAC operations respectively. When simulated in hardware, our prototype expresses a reasonable accuracy of 91.6% even with aggressive quantization. Our methodology highlights that the linearized and compressed CNN with fully analog RRAM crossbar arrays can be easily deployed in resource-constrained environments, such as IoT networks and edge platforms, to efficiently perform artificial intelligence and machine learning workloads.

Acknowledgment

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, wither expressed or implied, of the Air Force Research Laboratory (AFRL) or the U.S. Government. This material has been cleared for public release, unlimited distribution (PA Case Number: AFRL-2025-xxxx, xx-xxx 2025).

References

[1] S. Sinha, "State of iot 2023: Number of connected iot devices growing 16

[2] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[3] K. Sankhe, M. Belgiovine, F. Zhou, L. Angioloni, F. Restuccia, S. D'Oro, T. Melodia, S. Ioannidis, and K. Chowdhury, "No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 165–178, 2019.

[4] M. I. Jordan, *Attractor Dynamics and Parallelism in a Connectionist Sequential Machine*. IEEE Press, 1990, pp. 112–127.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[7] K. Bai, C. Thiem, N. McDonald, L. Loomis, and Y. Yi, "Toward intelligence in communication networks: A deep learning identification strategy for radio frequency fingerprints," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 204–209.

[8] K. Bai, L. Liu, and Y. Yi, "Spatial-temporal hybrid neural network with computing-in-memory architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2850–2862, 2021.

[9] K. J. Bai, C. Thiem, J. Lombardi, Y. Liang, and Y. Yi, "Design strategies and applications of reservoir computing: Recent trends and prospects [feature]," *IEEE Circuits and Systems Magazine*, vol. 23, no. 4, pp. 10–33, 2024.

[10] X. Yang, B. Taylor, A. Wu, Y. Chen, and L. O. Chua, "Research progress on memristor: From synapses to computing systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, pp. 1845–1857, 2022.

[11] H. Jia, M. Ozatay, Y. Tang, H. Valavi, R. Pathak, J. Lee, and N. Verma, "Scalable and programmable neural network inference accelerator based on in-memory computing," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 1, pp. 198–211, 2022.

[12] Y. Huang, V. Ravichandran, W. Zhao, and Q. Xia, "Towards energy-efficient computing hardware based on memristive nanodevices," *IEEE Nanotechnology Magazine*, vol. 17, no. 5, pp. 30–38, 2023.

[13] H. Jiang, K. Yamada, Z. Ren, T. Kwok, F. Luo, Q. Yang, X. Zhang, J. J. Yang, Q. Xia, Y. Chen, H. Li, Q. Wu, and M. Barnell, "Pulse-width modulation based dot-product engine for neuromorphic computing system using memristor crossbar array," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–4.

[14] K. J. Bai, H. Jiang, Z. Qin, and C. Thiem, "Temporal-encoded 6t-rram with bidirectional control for future neuromorphic systems," in *2024 25th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2024, pp. 1–6.

[15] K. J. Bai and H. Jiang, "Compute-in-memory with 6t-rram memristive circuit for next-gen neuromorphic hardware," in *2024 Neuro Inspired Computational Elements Conference (NICE)*. IEEE, 2024, pp. 1–5.

[16] Y. Shi, K. Davaslioglu, Y. E. Sagduyu, W. C. Headley, M. Fowler, and G. Green, "Deep learning for rf signal classification in unknown and dynamic spectrum environments," 2019.

[17] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "Oracle: Optimized radio classification through convolutional neural networks," 2018.

[18] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.

[19] Lyken17. (2022) pytorch-opcounter: A python library for counting flops and parameters in pytorch models. [Online]. Available: https://github.com/Lyken17/pytorch-OpCounter

[20] K. J. Bai, D. Titcombe, J. Lombardi, C. Thiem, and N. Cady, "Moving towards game-changing technology: Fabrication and application of hfo2 rram for in-memory computing," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–7.