Title:	KeyRAM: A 0.34uJ/decision 18k decisions/s Recurrent Attention In-memory Processor for Keyword Spotting
Archived version	Accepted manuscript: the content is identical to the published paper, but without the final typesetting by the publisher
Published version DOI :	10.1109/CICC48029.2020.9075923
Journal homepage	http://ieee-cicc.org/
Authors (contact)	Hassan Dbouk (hdbouk2@illinois.edu) Sujan K. Gonugondla (gonugon2@illinois.edu) Charbel Sakr (sakr2@illinois.edu) Naresh R. Shanbhag (shanbhag@illinois.edu)
Affiliation	University of Illinois at Urbana Champaign

Article begins on next page

KeyRAM: A 0.34 uJ/decision 18 k decisions/s Recurrent Attention In-memory Processor for Keyword Spotting

Hassan Dbouk, Sujan K. Gonugondla, Charbel Sakr, and Naresh R. Shanbhag
Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign, Urbana, USA.

***Abstract*—This paper presents a 0.34 uJ/decision deep learning-based classifier for keyword spotting (KWS) in 65 nm CMOS with all weights stored on-chip. This work adapts a Recurrent Attention Model (RAM) algorithm for the KWS task, and employs an in-memory computing (IMC) architecture to achieve up to 9× savings in energy/decision and more than 23× savings in EDP of decisions over a state-of-the art IMC IC for KWS using the Google Speech dataset while achieving the highest reported decision throughput of 18.32 k decisions/s.**

***Index Terms*—machine learning, keyword spotting, recurrent attention networks, in-memory computing**

I. Introduction

Speech has emerged as a natural mode for humans to interact with intelligent Edge devices including smart phones and personal digital assistants [1]. Preceding a speech recognizer with an 'always-on' keyword spotting (KWS) system (Fig. 1) enables such devices to continually sense, detect, and classify speech segments under stringent energy, computational and storage constraints. Recently, various deep learning based KWS classifier algorithms such as deep/convolutional neural networks (DNNs/CNNs) and recurrent NNs (RNNs) have been shown to achieve high ($> 90\%$) accuracies [2] but at the expense of very high computational costs (Fig. 2) making them unsuitable for sub-μJ/decision Edge platforms. Previous KWS IC implementations have been in digital [3]–[5] and for simple datasets, e.g., TIMIT. Recently, [6] implements a binarized RNN-based KWS on an in-memory computing (IMC) architecture for the more complex Google Speech [7] dataset achieving an energy-efficiency of 3.36 μJ/decision.

This work employs an algorithm-hardware co-design approach to realize KWS for Edge devices with $< 1\,\mu$J/decision. To the best our knowledge, this is the first work to propose using a Recurrent Attention Model (RAM) [8] for KWS *and* the first IC implementation of RAM. The use of RAM for the KWS task reduces the computational complexity of inference compared to state-of-the-art neural network-based algorithms (Fig. 2) at iso-accuracy. The proposed RAM algorithm is mapped onto a multi-bit multi-bank sparsity-aware IMC IC that stores all model weights on-chip to further increase the energy efficiency. As a result, up to 9$\times$ savings in energy/decision and $> 23\times$ savings in EDP of decisions over state-of-the art IMC ICs for KWS is achieved, while realizing the highest reported decision throughput of 18.32×10^3decisions/s.

This work was sponsored by the AFRL and DARPA under agreement FA-8650-18-2-7866 as part of the FRANC program.

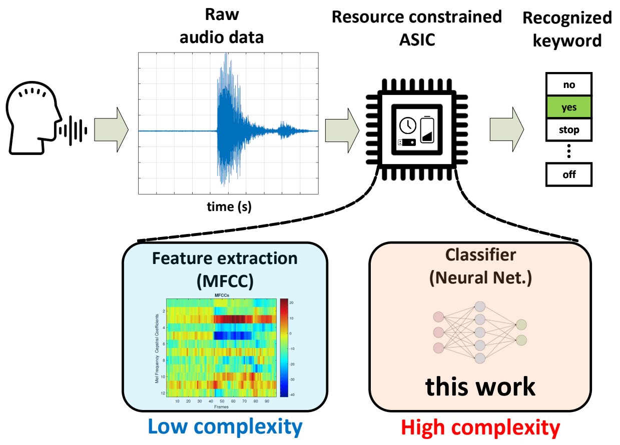
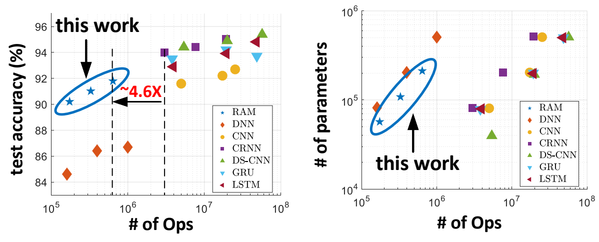


Fig. 1. A typical keyword spotting (KWS) pipeline processes raw audio in two stages: feature extraction and classification. The classification stage, which is implemented via neural networks, dominates the complexity of the KWS system.

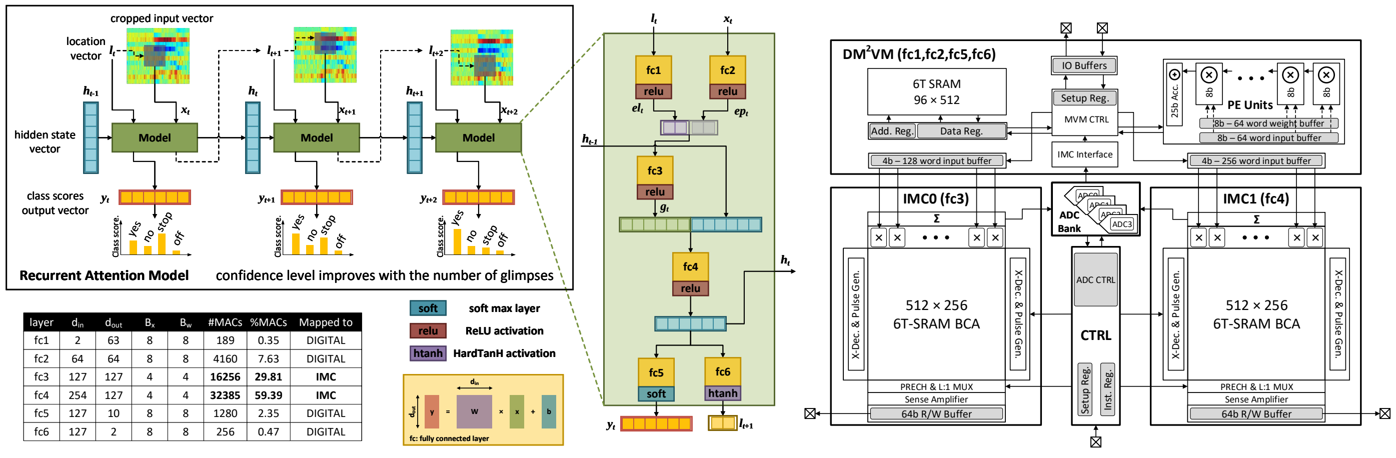


(a) (b)

Fig. 2. The accuracy-complexity-storage trade-off in the recurrent attention model (RAM) compared to other neural network classification algorithms for the KWS task with 12 classes using the Google Speech dataset: (a) test accuracy, and (b) number of parameters vs. number of operations.

II. Recurrent Attention Model for KWS

The proposed RAM for KWS employs a RNN in a feedback loop to selectively process input subsets (glimpses) in the feature space before classification thereby reducing the computational complexity by $\sim 4.6\times$ at similar accuracies (Fig. 2a) for the same dataset (Google Speech [7]). The RAM algorithm



layer	d_{in}	d_{out}	B_x	B_w	#MACs	%MACs	Mapped to
fc1	2	63	8	8	189	0.35	DIGITAL
fc2	64	64	8	8	4160	7.63	DIGITAL
fc3	127	127	4	4	**16256**	**29.81**	**IMC**
fc4	254	127	4	4	**32385**	**59.39**	**IMC**
fc5	127	10	8	8	1280	2.35	DIGITAL
fc6	127	2	8	8	256	0.47	DIGITAL

Fig. 3. The proposed RAM-based system for KWS: (a) algorithm, and (b) chip architecture.

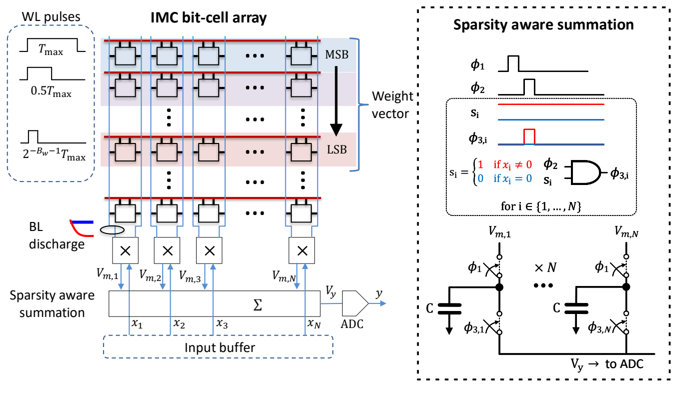


Fig. 4. The multi-bit IMC bank: (a) architecture, and (b) sparsity aware summation scheme that improves the IMC's dot product accuracy for sparse input vectors.

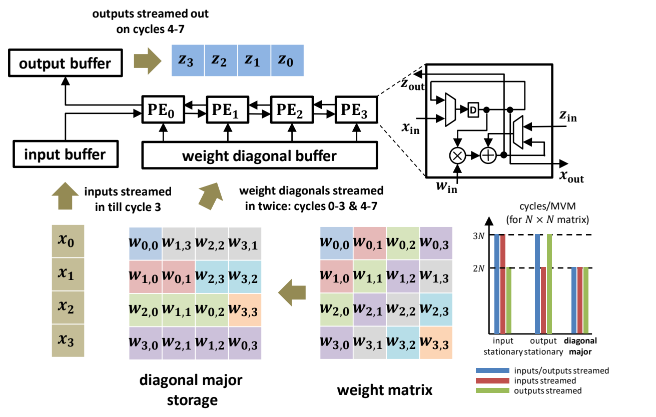


Fig. 5. The principle of the diagonal major MVM kernel (DM^{2}VM) for a simple 4×4 FC layer.

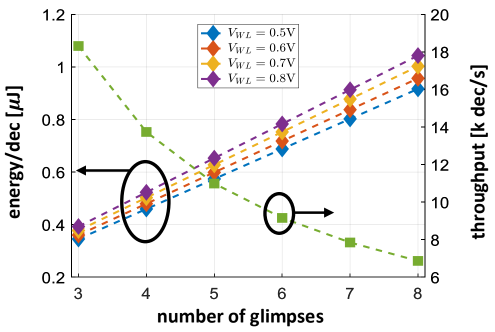
was originally proposed for image classification [8] where inputs are 2D images. To enable audio classification using RAM, we use Mel-frequency Cepstral Coefficient (MFCC) features as RAM inputs. Furthermore, while images have spatial invariance in both dimensions, audio features exhibit only temporal invariance, therefore we propose a location vector l_t that points to the time index of the glimpse.

The proposed RAM-based algorithm for KWS (Fig. 3a) employs 6 fully connected layers (fc1 to fc6) to track the informative features across multiple glimpses t via a hidden state vector h_t. At glimpse t, RAM combines h_{t-1} with the input patch x_t at location l_t to compute output class scores y_t indicating confidence levels, and updates the next glimpse location l_{t+1} and the corresponding hidden state vector h_{t+1}.

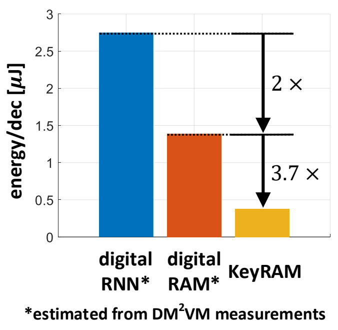
Precision analysis indicates that layers fc3 (127×127) and fc4 (254×127), which account for 89.2% of the total computational complexity, can be executed with 4-b precision while the rest need 8-bits. Therefore, fc3 and fc4 are mapped on to an IMC sub-system since it is best suited for low-precision matrix-vector multiplies (MVMs), and the rest (fc1, fc2, fc5, and fc6) are computed in high (8-b) precision digital. Thus, the proposed RAM for KWS not only requires fewer operations but is well-matched to a multi-bit IMC implementation. Moreover, unlike standard algorithms [2], a RAM-based KWS also enables a dynamic trade-off between energy and accuracy.

III. Chip Architecture

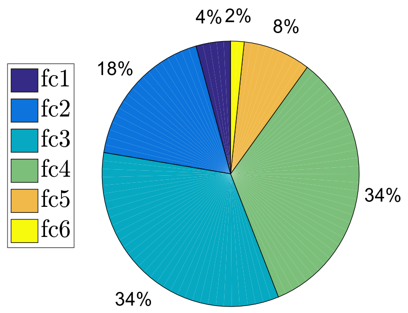
The proposed KeyRAM architecture (Figure 3b) stores all weights on-chip and comprises of: 1) two IMC blocks (IMC0 and IMC1) to implement multi-bit MVMs via temporal folding into a sequence of dot products. The IMC block is based on the single-bank IMC architecture in [9] which implements single dot-product per read cycle. Each IMC block in KeyRAM consists of a standard 6T SRAM 512×256 bit-cell array (BCA) with per-column multipliers and a cross-column adder. The two blocks share four 6-b ADCs and implement fc3 and fc4 respectively; 2) a diagonal major MVM kernel (DM^{2}VM) to efficiently implement fc1, fc2, fc5 and fc6 in digital; and 3) a



(a)



(b)



(c)

Fig. 7. Measurement results showing: (a) decision throughput and energy for varying V_{WL} and number of glimpses, (b) energy savings of KeyRAM compared to an all-digital implementation of RAM and RNN at iso-model size, and (c) energy breakdown of KeyRAM across different layers at $V_{\mathrm{WL}} = 0.7$V using three glimpses.

digital control block (CTRL) for timing synchronization. The IC is configurable and can perform 2–to–10 way classification.

A. IMC Block

The proposed multi-bit IMC architecture (Figure 4) realizes sparse dot products efficiently. The IMC (Fig. 4a) stores B_w-bit weights in the BCA in a column major format and computes dot-products between these weights and inputs stored in buffers in three steps: 1) stored digital weights are converted into analog via concurrently turned on pulse-width modulated word-lines (WLs), such that the voltage discharge on each bit-line (BL) is proportional to the multi-bit weight in the memory; 2) the BL discharges are multiplied with the corresponding input data from buffers via charge redistribution; and 3) the multiplier outputs are selectively summed across the columns via charge sharing across BLs resulting in the final dot product which is converted to digital via two 6-b single-slope ADCs operating at a sample rate of $10\,$M Sample/s. The ADC outputs are scaled and shifted to calibrate for offsets followed by a ReLU non-linearity. The accuracy of the IMC computations can be traded off with energy by controlling the word-line voltage V_{WL}. The IMC interface and MVM controller route the ADC outputs to appropriate input buffers in the DM^{2}VM.

The use of a ReLU activation function increases the input activations sparsity to $\sim 50\% - 70\%$. Sparse input vectors present a challenge for the charge sharing summation scheme used in [9] as the multiplier output voltage spread shrinks. A *sparsity-aware summing* method (Fig. 4b) is proposed in which the per-column multiplier output voltages are selectively charge shared based on whether the corresponding input element is zero or not. In the process, the output swing is preserved thereby improving the ADC accuracy.

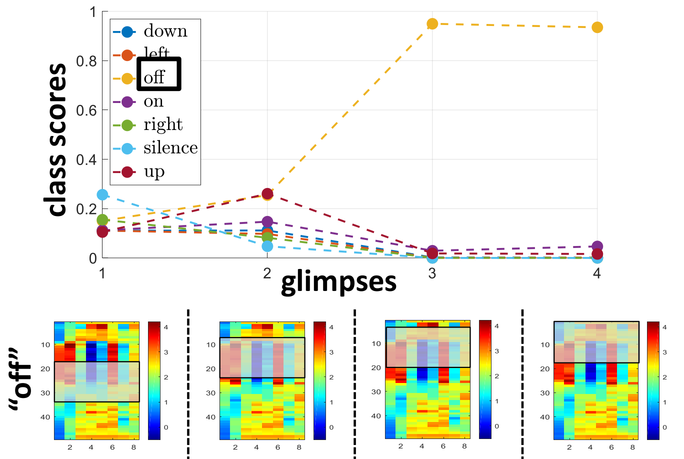


Fig. 6. Measured classifier confidence across glimpses for one input sample with the true class label 'off'. The IC correctly identifies the location in the input MFCC feature map which is highly correlated to the class label 'off'.

B. DM^{2}VM Block

The DM^{2}VM kernel (Fig. 5) digitally computes all MVM operations in fc1 (2×63), fc2 (64×64), fc5 (127×10), fc6 (127×2) with 8-b inputs and 8-b weights. The MVM dot-products are computed via 64 processing elements (PEs), each implementing an 8-b×8-b multiply-accumulate (MAC) operation. The 25-b accumulated dot-product outputs are truncated to 8-b per algorithmic requirements. The DM^{2}VM processor is designed to minimize idle cycles when inputs/outputs are streamed in/out since the diagonal major architecture is able to complete an $N \times M$ MVM in a fixed number $N + M$ of cycles irrespective of whether $N > M$ or vice-versa. In contrast, an input (output) stationary architecture requires between $2N + M$ ($N + 2M$) and $N + M$ cycles, respectively, based on whether the input/output/both are streamed or not. The DM^{2}VM fetches weight diagonals from its SRAM where weights are stored in diagonal major format, begins computation as soon as the first input is streamed in, and stops exactly when the final output is streamed out without any stalls. This makes the DM^{2}VM well-matched to the diverse set of MVM dimensions utilized by the RAM algorithm.

IV. Measurements Results

Measurements on the IC are performed using the Google Speech dataset [7]. The RAM was trained for 7-way classification using $11\,$k data samples each corresponding to a $1\,$s keyword sampled at a $16\,$kHz. The inputs to the classifier on

TABLE I
COMPARISON TABLE.

	ISSCC'17 [3]	CICC'18 [4]	ESCCIRC'18 [5]	VLSI'19 [6]	This Work
Technology	65 nm	65 nm	65 nm	65 nm	65 nm
Algorithm	DNN	LSTM	LSTM	Binarized-RNN	RAM
Dataset	TIDIGITS	TIMIT	TIMIT	Google Speech	Google Speech
# of Classes	11	39	4[a]	10	7
Test Accuracy [%]	98.35	80.4	–	90.2	90.38
On-chip Storage [kB]	747.52	82	32	18	38
Area [mm^2]	9.61	1.57	1.035	6.2	4.13
Energy/Decision [μJ]	6.4[d]	9.54[d]	0.06	3.36	$0.34 - 1.043$[b] ($0.57 - 1.62$)[c]
Decisions/s	26.8[d]	1.3 k[d]	83.3[d]	7.8 k	**$6.86\,\mathrm{k} - 18.32\,\mathrm{k}$**[b]
# of MACs/Decision	–	–	$5.8\,\mathrm{k} - 27.2\,\mathrm{k}$	–	$273\,\mathrm{k} - 730\,\mathrm{k}$[b]
Energy-Delay Product [pJ.s]	239 k[d]	7.3 k[d]	720	430	**$18{-}152$**[b] ($31 - 236$)[c]
Supply Voltage [V]	$0.6 - 1.2$	$0.75 - 1.24$	0.575	$0.9 - 1.1$	1
Energy Efficiency [TOPS/W]	–	3.08	–	11.7	1.6 (0.91)[c]

[a]4 binary classifiers [b]with changing V_{WL} and # of glimpses [c]with CTRL energy included [d]estimated from reported data

the chip are 8-channel MFCCs extracted within $40\,\mathrm{ms}$ windows with $20\,\mathrm{ms}$ overlap resulting in a 8×49-dimensional feature vector. The input patch dimension at each glimpse is 8×8 and the locator is a one-dimensional scalar.

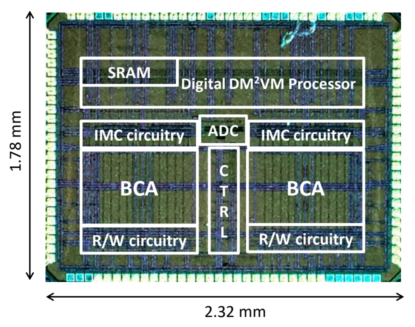
The measured classification accuracy is 90.38% at $V_{\mathrm{WL}} = 0.7\,\mathrm{V}$ with 4 glimpses. The classification accuracy improves with the number of glimpses in proportion to the classifier's confidence (Fig. 6). The decision energy and latency linearly increase with the number of glimpses (Fig. 7a). Combined with a tunable WL pulse amplitude V_{WL}, the RAM-based KWS IC incorporates dynamic energy-accuracy trade-offs, e.g., the energy/glimpse varies from $0.11\,\mu$J-to-$0.13\,\mu$J as V_{WL} varies from $0.5\,\mathrm{V}$-to-$0.8\,\mathrm{V}$. The measured energy breakdown (Fig. 7c) shows that fc3 and fc4 which account for 89% of computations consumes 68% of the total energy consumption. These savings are attributed to the use of IMC. Comparison with a digital architecture implementing a standard RNN with the same model size in Figure 7b shows a $7.3\times$ savings in decision energy of which $2\times$ and $3.7\times$ is attributed to the use of the modified RAM algorithm and the IMC, respectively. The CTRL energy ($0.08\,\mu$J/glimpse) can be amortized with larger problem sizes.

Table I compares KeyRAM with state-of-the-art digital [3]–[5] and in-memory [6] KWS IC implementations. KeyRAM achieves between $3\times$-to-$9\times$ reduction in energy/decision compared to the IMC [6]. In addition, $> 23\times$ reduction in the decision energy-delay product (EDP) compared to other KWS implementations. The proposed art achieves the highest reported throughput at $18.32\,\mathrm{k}$ decisions/s.

Figure 8 shows the die micrograph of the 65nm CMOS IC along with its summary.

V. CONCLUSION

This paper presents KeyRAM, a classifier IC for keyword spotting for edge applications. KeyRAM achieves an energy efficiency of $< 0.5\,\mu$J/decision for a multi-class Google Speech dataset employing an algorithm-hardware co-design approach thereby meeting the requirements of real-time Edge applications. The concepts presented in this paper can be extended to other applications such as video inference where data access costs dominate even more.



Technology	65nm
Die Size	1.78mm × 2.32 mm
Memory Capacity	38kB
Nominal Supply	1.0V
CTRL Frequency	1GHz
Throughput	6.86k – 18.32k
Energy/dec	0.34μJ-1.043μJ
Algorithm	RAM

Fig. 8. Die micrograph and chip summary.

REFERENCES

[1] I. McGraw *et al.*, "Personalized speech recognition on mobile devices," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5955–5959.

[2] Y. Zhang *et al.*, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[3] M. Price *et al.*, "A scalable speech recognizer with deep-neural-network acoustic models and voice-activated power gating," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 244–245.

[4] F. Conti *et al.*, "Chipmunk: A systolically scalable 0.9 mm 2, 3.08 GOP/s/mW@ 1.2 mW accelerator for near-sensor recurrent neural network inference," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2018, pp. 1–4.

[5] J. S. Giraldo *et al.*, "Laika: A 5uW programmable LSTM accelerator for always-on keyword spotting in 65nm CMOS," in *IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, 2018, pp. 166–169.

[6] R. Guo *et al.*, "A 5.1 pJ/neuron 127.3 us/inference RNN-based speech recognition processor using 16 computing-in-memory SRAM macros in 65nm CMOS," in *Symposium on VLSI Circuits*. IEEE, 2019, pp. C120–C121.

[7] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[8] V. Mnih *et al.*, "Recurrent models of visual attention," in *Advances in neural information processing systems*, 2014, pp. 2204–2212.

[9] M. Kang *et al.*, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.