In-memory principal component analysis by crosspoint array of resistive switching memory

Pieriuli Mannocci, Graduate Student Member, IEEE, Andrea Baroni, Enrico Melacarne, Cristian Zambelli, Member, IEEE, Piero Olivo, Eduardo Pérez, Christian Wenger, and Daniele Ielmini, Fellow, IEEE

Abstract—In-memory computing (IMC) is one of the most promising candidates for data-intensive computing accelerators of machine learning (ML). A key ML algorithm for dimensionality reduction and classification is the principal component analysis (PCA), which heavily relies on matrix-vector multiplications (MVM) for which classic von Neumann architectures are not optimized. Here, we provide the experimental demonstration of a new IMC-based PCA algorithm based on power iteration and deflation executed in a 4-kbit array of resistive switching random-access memory (RRAM). The classification accuracy of Wisconsin Breast Cancer dataset reaches 95.43%, close to floating-point implementation. Our simulations indicate a 250× improvement in energy efficiency compared to commercial graphic processing units (GPUs), thus supporting IMC for energy-efficient ML in modern data-intensive computing.

Index Terms—In-memory computing, resistive random access memory, hardware accelerator, principal component analysis

I. INTRODUCTION

Machine learning (ML) is becoming ubiquitous in our daily life, stimulating the development of new computing hardware to reduce energy consumption and maximize throughput [1]. Digital computing cannot meet these requirements, due to energy and latency cost of von Neumann architecture [2]. In-memory computing (IMC) has risen as one of the most promising candidates for next-generation computing given its high throughput, low energy and good scaling thanks to high-density crosspoint arrays of resistive memories [3]. Owing to its inherent parallelism, IMC is well-suited to accelerate algebraic operations such as matrix-vector multiplication (MVM), which is the cornerstone of many ML tasks. A key algorithm in ML is the principal component analysis (PCA), capable of reducing data dimensionality thus enabling clustering and feature extraction [4]. PCA identifies the principal components (PCs), namely the vector basis which maximizes data variance (Fig. 1a). Adopting the PCs as a new reference basis can reveal data clusters and enable classification (Fig. 1b). PCA algorithms include eigendecomposition (ED) [4], singular value decomposition (SVD) [5] and non-linear iterative partial least squares (NIPALS) [6], [7]. However, the digital implementation of these algorithms is not efficient, due to the need for massive MVM and extensive data movement. This work presents a new IMC-based PCA algorithm, consisting of a deflated power iteration to extract the PCs. The algorithm is experimentally validated on a 4-kbit resistive switching random access memory (RRAM) array revealing software-equivalent accuracy in classifying the Wisconsin Breast Cancer dataset. The simulated energy efficiency outperforms commercial graphic processing units (GPUs) by a factor of 250.

II. IMC-BASED PCA ALGORITHM

The ED approach to PCA is illustrated in Fig. 2. First, the dataset $\hat{X} \in \mathbb{R}^{n \times h}$ is centered by subtracting the variable-wise mean value, thus resulting in the zero-mean dataset $X$. Then, the empirical covariance matrix $X^T X$ is computed and decomposed into eigenvectors and eigenvalues, according to:

$$
X^T X = V \Lambda V^{-1} \tag{1}
$$

Fig. 1. Principal Component Analysis. From a multi-dimensional dataset with observation of many variables (a), PCA extracts the directions of greater variance, a projection upon which (b) allows to highlight clusters and untangle correlations.

Fig. 2. Eigendecomposition approach to PCA. The dataset is centered to remove bias, then its covariance matrix is computed and decomposed to find eigenvectors (principal components) and eigenvalues (variance contribution). A subset of the PCs is used as reference frame for projection.
where \( V \) is the matrix of eigenvectors and \( A \) is the (diagonal) matrix of eigenvalues [8]. A subset of \( p \) eigenvectors corresponding to the largest \( p \) eigenvalues is then chosen, with the aim of enclosing as much variance as possible while at the same time reducing the number of variables. Typically, eigenvectors associated to eigenvalues larger than 1 are retained [9], while those associated to eigenvalues smaller than 1 are discarded, such that the new reference basis is given by \( P = V_{1:p} \). Finally, the dataset is recast in the new basis as \( Y = XP \).

The first eigenpair \( \{ \lambda_1, e_1 \} \) can be readily computed by means of a power iteration on the empirical covariance matrix \( X^TX \). The remaining eigenpairs are then obtained by a modified version of the numerical Hotelling deflation algorithm (HDA) [10], where the subspace associated with a given eigenvector is removed from the covariance matrix while preserving the remaining eigenpairs. HDA is illustrated in Fig. 3. Given a matrix \( A_1 \) with eigenvalues \( \{ \lambda_1, \lambda_2, \ldots, \lambda_n \} \) and eigenvectors \( \{ e_1, e_2, \ldots, e_n \} \), the deflated matrix \( A_2 = A_1 - \lambda_1 e_1 e_1^T \) has eigenvalues \( \{ 0, \lambda_2, \ldots, \lambda_n \} \) and eigenvectors \( \{ e_1, e_2, \ldots, e_n \} \). Performing a power iteration on \( A_2 \) then returns the eigenpair \( \{ \lambda_2, e_2 \} \). By deflating again with respect to \( e_2 \), we obtain matrix \( A_3 = A_2 - \lambda_2 e_2 e_2^T \) with eigenvalues \( \{ 0, 0, \lambda_3, \ldots, \lambda_n \} \) and eigenvectors \( \{ e_1, e_2, e_3, \ldots, e_n \} \), where the power iteration yields the eigenpair \( \{ \lambda_3, e_3 \} \). The deflation/iteration scheme is repeated until all eigenvalues/eigenvectors of interest, e.g., eigenvalues larger than 1 and their eigenvectors, are obtained.

To accelerate the computation of principal components using IMC, we first map the zero-mean dataset matrix \( X \) into a physical memory array. Then we use the two-step approach of Fig. 4 to execute the power iteration on \( X^TX \). In the first MVM step (Fig. 4a), we apply a vector \( v_1 \) of voltages to the columns of the memory array for the second MVM step (Fig. 4b), resulting in a current vector \( i_2 = X^T v_2 = X^T X v_1 \). Repeating the process allows to complete the first power iteration to extract the leading eigenpair \( \{ \lambda_1, e_1 \} \) of the empirical covariance matrix \( X^TX \).

Deflation techniques can be directly applied to the 2-step IMC-accelerated power iteration approach of Fig. 4, as they can be implemented by simply adding extra rows to the memory array containing the dataset \( X \), as shown in Fig. 5. After the first power iteration, eigenvector \( e_1 \) is stored in an additional row of the memory array. Applying a voltage vector \( v_1 \) to the columns of the memory array now yields the current vector (Fig. 5a):

\[
    i_1 = \begin{bmatrix} y \\ y_{os} \end{bmatrix} = \begin{bmatrix} X \\ e_1^T \end{bmatrix} v_1
\]

The sub-vector \( y \) is directly converted to voltage, whereas sub-vector \( y_{os} \) is multiplied by \( -\lambda_1 \) and converted to voltage (Fig. 5b). The converted sub-vectors are reapplied to the rows of the memory array, thus yielding the current vector \( i_2 \) on...
array columns (Fig. 5c), given by:

\[
\mathbf{i}_2 = \mathbf{X}^T \mathbf{y} + \mathbf{e}_1 (-\lambda_1 \mathbf{y}_{\text{os}}) \\
= \mathbf{X}^T \mathbf{X} \mathbf{v}_1 - \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T \mathbf{v}_1 \\
= (\mathbf{X}^T \mathbf{X} - \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T) \mathbf{v}_1
\]

which is equivalent to the \(\{\lambda_1, \mathbf{e}_1\}\)-deflated version of \(\mathbf{X}^T \mathbf{X}\).

Every time an eigenpair is computed, the eigenvector is stored in an additional row of the memory array and a new power iteration is executed to calculate the next eigenpair. This deflation scheme is repeated until an eigenvalue \(\lambda_{p+1} < 1\) is found. The total memory overhead to store the \(p\) eigenvectors is only \(p \times n\) cells. The algorithm only consists of \(\mathcal{O}(1)\) in-memory MVM iterations, except for \(\mathcal{O}(m+p)\) conversion and multiplication steps in the DSP.

To test our proposed algorithm, we ran simulations on the Iris dataset [11], a collection of 150 observations for three species of flowers of the Iris genus (Fig. 6a). For each species, 50 observations of petal and sepal length and width are recorded (Fig. 6b). Fig. 7 shows the result of 100 IMC-PCA simulations where we assumed ideal 3-bit precision of conductances with a maximum value of 100 \(\mu\)S and a variable programming error \(\sigma_G = 1, 2, 5\) and 10 \(\mu\)S. As a figure of merit, we considered the cosine similarity (CoSim) of the computed PCs with respect to software simulation results obtained by MatLab’s PCA routines with floating-point 64-bit precision. The results show a significantly larger error for PC2 compared to PC1, where the error increases with \(\sigma_G\). The results also highlight that error progressively accumulates as the number of computed PCs increases, thus imposing a constraint on precise programming to avoid excessive degradation.

### III. IMC HARDWARE AND PROGRAMMING

The PCA algorithm was validated by IMC experiments on a 4 kbit array of TiN/Ti/HfO_2/TiN RRAM manufactured by IHP Microelectronics [12]. Fig. 8 shows the structure (a) and the I-V characteristics (b) of the RAM device. Experiments were performed on packaged chips (Fig. 8c) by RIFLE SE, a non-volatile memory tester manufactured by Active Technologies [13] (Fig. 8d). RRAM devices have a one-transistor/one-resistor (1T1R) structure in 0.25 \(\mu\)m CMOS technology.

Accurate multilevel programming of the RRAM was achieved by the incremental gate voltage verify algorithm (IGVVA), where the gate voltage was gradually incremented from 0.5 V to 1.7 V with 10 mV step during the set programming pulse, consisting of a 1 \(\mu\)s pulse of amplitude \(V_{TE} = 1.2\) V applied at the top electrode [14]. The IGVVA algorithm allows for accurate control of the RRAM filament size, hence tight distribution of conductance \(G\) as shown in Fig. 9a. To test the PCA algorithm, the RRAM devices were programmed into eight conductance levels (\(L_1\)-\(L_8\)) linearly spaced between \(G = 50\) \(\mu\)S and 225 \(\mu\)S, and one level (\(L_0\)) at low conductance \(G = 25\) \(\mu\)S. Fig. 9b shows the cumulative distribution of \(G\) for the 9 levels: the standard deviation \(\sigma_G\)
IV. IMC EXPERIMENTS

To test the IMC experimental concept, we used two datasets with different size and complexity. We first considered the Iris dataset, whose data were mapped differentially onto 1200 devices, as shown by the accuracy plot of Fig. 11a. Statistical fitting by means of the 3-component Gaussian mixture [15] in Fig. 11b yields an overall mean error $\mu_G = -0.2 \mu S$ with standard deviation $\sigma_G = 4.53 \mu S$. The IMC-PCA algorithm was run on the RRAM array by executing the MVM operations in hybrid mode, i.e., multiplication was done in situ according to Ohm’s law $I_{ij} = G_{ij}V_{ij}$, while the summation was executed ex situ by summing cell currents externally in a digital processor. Fig. 12a shows the power iteration to extract the first PC, converging within about 10 cycles of 2-step MVM and achieving a final cosine similarity (CoSim) of 0.99997. The first eigenvector was then mapped in an extra row and the...
second PC was computed in 10 MVM iterations with $\text{CoSim} = 0.995$ (Fig. 12b). Fig. 13 shows the resulting biplot, namely the dataset projected along the extracted PCs: the cluster of *Iris setosa* is linearly separated from *Iris virginica* and *Iris versicolor* thus confirming the dimensionality reduction and classification. Results tightly agree with those obtained from software floating-point 64bit precision (FP64).

To provide further experimental support to our IMC-based concept of PCA, we considered the Glass dataset [16], a collection of 214 observations of glass materials from various sources. The 1926 entries of the dataset were programmed in 3852 RRAM devices in differential mode. Fig. 14a shows the correlation plot of programmed conductance as a function of target conductance. The error distribution in Fig. 14b was fitted by means of a 4-component Gaussian mixture, yielding a mean programming error $\mu_G = 0.68\,\mu$S with standard deviation $\sigma_G = 15.1\,\mu$S. IMC-PCA was carried out, with the first PC extracted in 10 iterations with $\text{CoSim} = 0.97$ (Fig. 15a) and the second PC extracted in 10 iterations with $\text{CoSim} = 0.91$ (Fig. 15b). Fig. 16 shows the resulting biplot, evidencing the good agreement with FP64 results, which supports the accuracy of the IMC-based PCA approach.

V. BENCHMARK AND SCALING

To assess the scaling potential of our in-memory PCA algorithm, we carried out large-scale simulations on the Wisconsin Breast Cancer dataset [18], which records 30 different variables of diagnostic interest for 569 breast cancer patients, totaling 17070 entries. Using the differential scheme of Fig. 10, 34260 devices would be required to perform IMC-PCA. The programming procedure was simulated in software, where stochastic variations were assumed according to the experimental distributions of Fig. 9b. Figs. 17a-b show the mapping accuracy with $\mu_G = 0.29\,\mu$S, $\sigma_G = 8.40\,\mu$S upon fitting with a 3-component Gaussian mixture. Fig. 17c shows the resulting biplot of the dataset projected along the first two PCs: the obtained clusters allow to classify benign and malignant tumors with 95.43% accuracy after logistic
regression [19].

To comprehensively account for all energy, area and latency contributions, we considered the architecture in Fig. 18a, where the RRAM array is complemented by switch matrices allowing to reroute alternatively rows and columns to digital-analog converter (DAC) or current-to-digital converter, realized by means of a transimpedance amplifier cascaded with an analog-to-digital converter (TIA/ADC), depending on the MVM step. The performance of all peripherals was extracted from NeuroSim [20] and available literature [21] considering the 14 nm lithographic node. Figs. 18b-c report area and energy breakdown for the whole system, including both dataset programming and PC computation contributions, highlighting that the majority of energy is consumed in programming due to both the large number of dataset entries and high target conductances. However, the programming procedure may be optimized independently of the PC computation algorithm, motivating the analysis of area and energy breakdown for the PC computation only (Figs. 18d-e), where now DSP represents the major source of energy dissipation and area occupation. When compared with a commercial GPU [17], PC computation using IMC-based PCA achieves compara-
The computational complexity of the in-memory PCA algorithm is due to the high parallelism of MVM (Fig. 20). The computational complexity of MVM on an $m \times n$ matrix, $k$ is the number of iterations of the power iteration section, and $p$ is the number of computed principal components. In contrast, the energy complexity of the IMC-PCA algorithm is:

$$O(p \cdot k \cdot (2\alpha \cdot mn + \beta \cdot (m + p - 1)))$$ (6)

where the first term accounts for the in-memory MVMs, and the second term accounts for the DSP steps. $\alpha$ is the typical energy dissipated on a single device during MVM, and $\beta$ is the energy of one FLOP in DSP, with $\alpha \ll \beta$ typically. At low dataset sizes, the energy consumption is dominated by DSP, corresponding to an $O(n)$ energy efficiency. As the dataset size increases, energy dissipation on the resistive array becomes the dominant contribution, thus leading to a saturation of energy efficiency to a constant value in the order of $O(1/\alpha)$, corresponding to $\sim 17$ TOPS/W, which may be further improved by moving to higher resistive states. These results support IMC for energy-efficient accelerators of data-intensive ML tasks.

Fig. 20. Energy performance as a function of dataset size for the proposed IMC-PCA algorithm, for synthetically-generated Gaussian-distributed datasets.

VI. Conclusion

We present a novel IMC-based PCA hardware architecture based on power iteration and deflation by in situ, parallel, 2-step MVM in RRAM arrays. The new concept is validated experimentally on a 4-kbit RRAM array in CMOS 0.25 $\mu$m, and complemented with extensive simulations in a realistic benchmark framework. The IMC-based PCA shows an accuracy comparable to FP64, while benefiting from smaller area and higher energy efficiency compared to commercial GPUs. These results support IMC as a strong candidate for data-intensive ML accelerators.

ACKNOWLEDGMENT

This work was supported in part by the European Research Council (grant ERC-2014-CoG-648635-RESCUE).

REFERENCES