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

Piergiulio 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.

P. Mannocci, E. Melacarne and D. Ielmini are with Politecnico di Milano, Milano, Italy.

A. Baroni, C. Zambelli and P. Olivo are with Università degli Studi di Ferrara, Ferrara, Italy.

E. Pérez and C. Wenger ar with IHP-Leibniz Institut für innovative Microelektronik, Frankfurt (Oder), Germany.

C. Wenger is also with BTU Cottbus-Senftenberg, Cottbus, Germany.

Correspondence should be addressed to piergiulio.mannocci@polimi.it and daniele.ielmini@polimi.it.

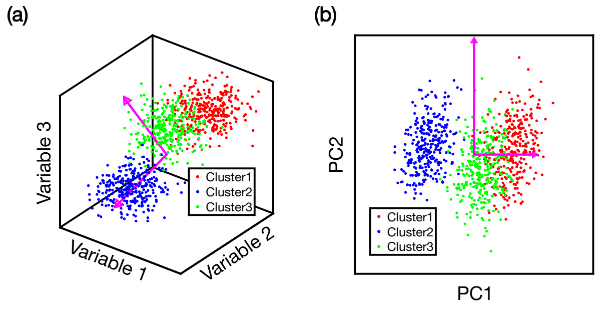

Fig. 1. Principal Component Analysis. From a multi-dimensional dataset with observation of many variables (a), PCA extracts the directions of greater variance, a projection upon which (b) allows to highlight clusters and untangle correlations.

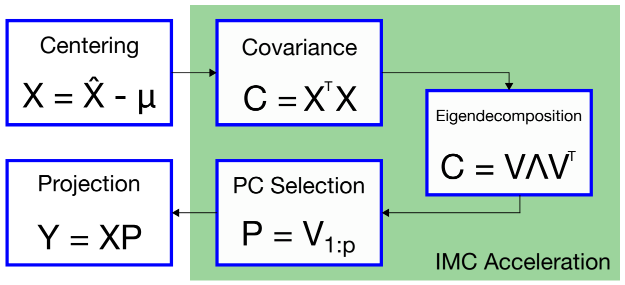

Fig. 2. Eigendecomposition approach to PCA. The dataset is centered to remove bias, then its covariance matrix is computed and decomposed to find eigenvectors (principal components) and eigenvalues (variance contribution). A subset of the PCs is used as reference frame for projection.

II. IMC-based PCA Algorithm

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

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

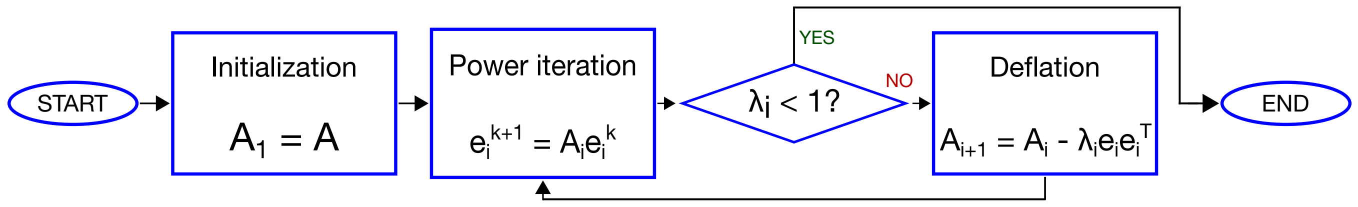

Fig. 3. Deflation algorithm for ED of a generic matrix $\mathbf{A}$. Every time an eigenpair is computed, the associated subspace is removed from the iteration matrix and a new power iteration is run. The stopping criterion is adapted to suit IMC-PCA.

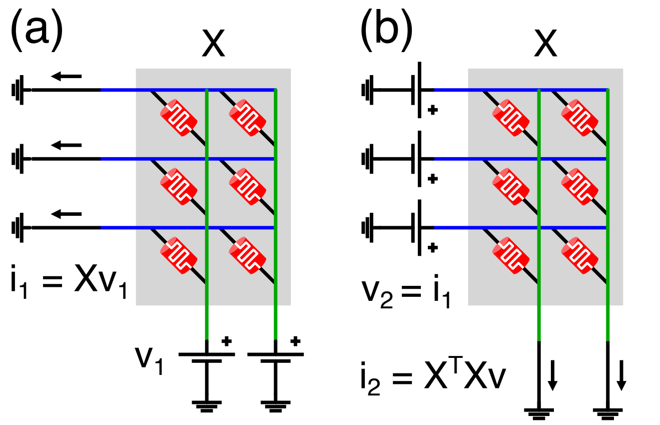

Fig. 4. Schematic representation of 2-step MVM scheme with crossbar arrays. (a) Voltage vector $\mathbf{v_1}$ is applied to the memory array $\mathbf{X}$, generating current vector $\mathbf{i_1}$. (b) $\mathbf{i_1}$ is converted to voltage $\mathbf{v_2}$ in DSP, and reapplied to the array rows. Current vector $\mathbf{i_2}$ is then probed on columns.

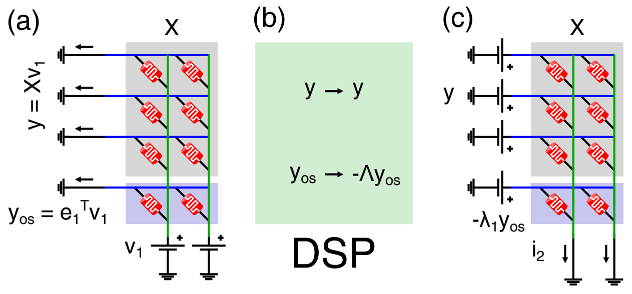

Fig. 5. In-memory deflation algorithm. (a) Voltage vector $\mathbf{v_1}$ is applied to the $[\mathbf{X}^T\mathbf{e_1}]^T$ array, generating current vectors $\mathbf{y}$ and $\mathbf{y_{os}}$. (b) In DSP, $\mathbf{y}$ is converted to voltage, whereas $\mathbf{y_{os}}$ is inverted, each entry scaled by its λ and converted to voltage. (c) Converted vectors are reapplied to the array rows. Current vector $\mathbf{i_2}$ is probed on columns.

where $\mathbf{V}$ is the matrix of eigenvectors and $\mathbf{\Lambda}$ 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 $\mathbf{P} = \mathbf{V}_{1:p}$. Finally, the dataset is recast in the new basis as $\mathbf{Y} = \mathbf{XP}$.

The first eigenpair $\{\lambda_1, \mathbf{e_1}\}$ can be readily computed by means of a power iteration on the empirical covariance matrix $\mathbf{X}^T\mathbf{X}$. 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 $\mathbf{A_1}$ with eigenvalues $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ and eigenvectors $\{\mathbf{e_1}, \mathbf{e_2}, \dots, \mathbf{e_n}\}$, the deflated matrix $\mathbf{A_2} = \mathbf{A_1} - \lambda_1\mathbf{e_1}\mathbf{e_1}^T$ has eigenvalues $\{0, \lambda_2, \dots, \lambda_n\}$ and eigenvectors $\{\mathbf{e_1}, \mathbf{e_2}, \dots, \mathbf{e_n}\}$. Performing a power iteration on $\mathbf{A_2}$ then returns the eigenpair $\{\lambda_2, \mathbf{e_2}\}$. By deflating again with respect to $\mathbf{e_2}$, we obtain matrix $\mathbf{A_3} = \mathbf{A_2} - \lambda_2\mathbf{e_2}\mathbf{e_2}^T$ with eigenvalues $\{0, 0, \lambda_3, \dots, \lambda_n\}$ and eigenvectors $\{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}, \dots, \mathbf{e_n}\}$, where the power iteration yields the eigenpair $\{\lambda_3, \mathbf{e_3}\}$. 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 $\mathbf{X}$ into a physical memory array. Then we use the two-step approach of Fig. 4 to execute the power iteration on $\mathbf{X}^T\mathbf{X}$. In the first MVM step (Fig. 4a), we apply a vector $\mathbf{v_1}$ of voltages to the columns of the memory array thus obtaining a current vector $\mathbf{i_1} = \mathbf{X}\mathbf{v_1}$ at the rows of the same array. The current vector $\mathbf{i_1}$ is then converted back to a voltage vector $\mathbf{v_2}$ and applied to the rows of the memory array for the second MVM step (Fig. 4b), resulting in a current vector $\mathbf{i_2} = \mathbf{X}^T\mathbf{v_2} = \mathbf{X}^T\mathbf{X}\mathbf{v_1}$. Repeating the process allows to complete the first power iteration to extract the leading eigenpair $\{\lambda_1, \mathbf{e_1}\}$ of the empirical covariance matrix $\mathbf{X}^T\mathbf{X}$. Note that the 2-step MVM approach allows to execute the power iteration without explicitly computing and mapping the empirical covariance matrix $\mathbf{X}^T\mathbf{X}$.

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 $\mathbf{X}$, as shown in Fig. 5. After the first power iteration, eigenvector $\mathbf{e_1}$ is stored in an additional row of the memory array. Applying a voltage vector $\mathbf{v_1}$ to the columns of the memory array now yields the current vector (Fig. 5a):

$$\mathbf{i_1} = \begin{bmatrix} \mathbf{y} \\ \mathbf{y_{os}} \end{bmatrix} = \begin{bmatrix} \mathbf{X} \\ \mathbf{e_1}^T \end{bmatrix} \mathbf{v_1} \tag{2}$$

The sub-vector $\mathbf{y}$ is directly converted to voltage, whereas sub-vector $\mathbf{y_{os}}$ 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 $\mathbf{i_2}$ on

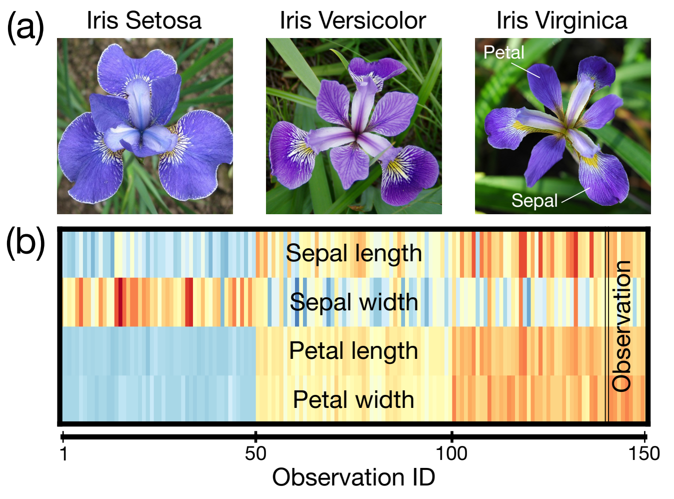

Fig. 6. (a) The Iris dataset records 150 observations of sepal and petal widths and lengths, for three different species of the *Iris* genus. (b) Colorplot of the 600 dataset entries. Each column represents a single observation of the four variables, namely sepal length and width, and petal length and width.

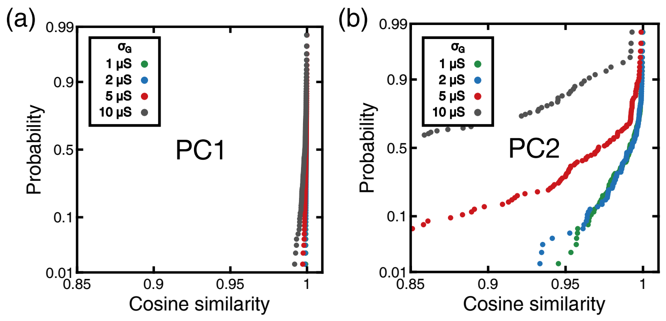

Fig. 7. Cosine similarity of Iris dataset PCs computed using IMC-PCA and simulated 3-bit conductances with variable programming error σ_G, with respect to 64-bit floating-point PCs. (a) CoSim of the first principal component, showing minimal variations. (b) CoSim of the second principal component, higlighting a greater impact of conductance variability.

array columns (Fig. 5c), given by:

$$\mathbf{i_2} = \mathbf{X}^T\mathbf{y} + \mathbf{e_1}(-\lambda_1\mathbf{y_{os}}) \quad (3)$$

$$= \mathbf{X}^T\mathbf{X}\mathbf{v_1} - \lambda_1\mathbf{e_1}\mathbf{e_1}^T\mathbf{v_1} \quad (4)$$

$$= (\mathbf{X}^T\mathbf{X} - \lambda_1\mathbf{e_1}\mathbf{e_1}^T)\mathbf{v_1} \quad (5)$$

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 µS and a variable programming error $\sigma_G = 1, 2, 5$ and 10 µS. As a figure of merit, we considered the

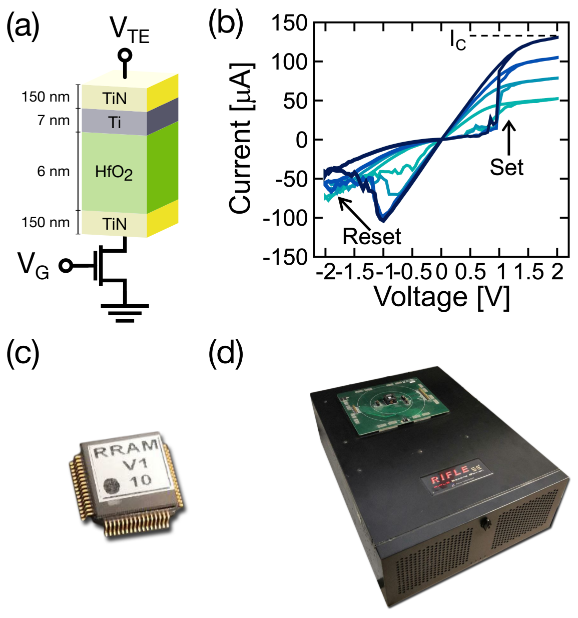

Fig. 8. Experimental setup. (a) Schematic of a 1T1R RRAM device used in this work. The RRAM has a stack consiting of a Ti-based oxygen reservoir and an amorphous HfO_2 switching layer sandwiched between TiN TE and BE. (b) Multilevel I-V characteristics of 1T1R RRAM device measured for increasing V_G. (c) Packaged 4-kbit array of 1T1R RRAM devices manufactured by IHP Microelectronics. (d) Active Technologies non-volatile memory tester Rifle SE.

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 σ_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 RRAM 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 µm 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 µs 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 $G = 50$ µS and 225 µS, and one level (L_0) at low conductance $G = 25$ µS. Fig. 9b shows the cumulative distribution of G for the 9 levels: the standard deviation σ_G

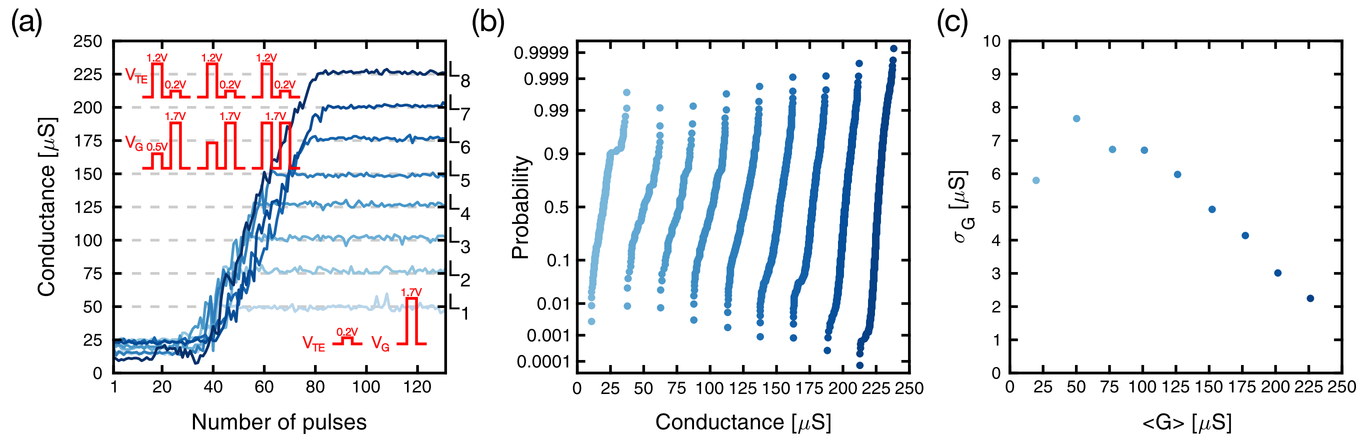

Fig. 9. IGVVA algorithm. (a) Measured conductance of a single RRAM device as a function of the number of applied programming pulses. (b) Cumulative distributions of the 9 test levels, comprising 8 LRS states linearly spaced between 50 µS and 225 µS and 1 HRS level at 25 µS. (c) Conductance variability as a function of the median conductance, for the nine target levels L_0-L_8. L_1 is the less controllable level of this technology, whereas L_8 achieves the minimum variability.

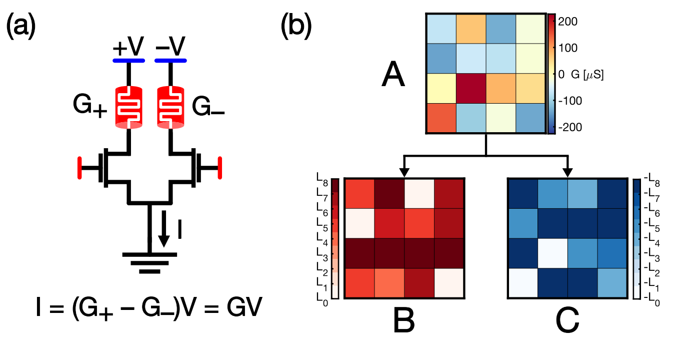

Fig. 10. Differential mapping scheme for dataset entries. (a) Differential cell, with two conductances G_+, G_- driven by opposite voltages. (b) Systematic mapping procedure for a generic matrix $\mathbf{A}$ by generation of matrices $\mathbf{B}, \mathbf{C}$: $\mathbf{A} = \mathbf{B} - \mathbf{C}$.

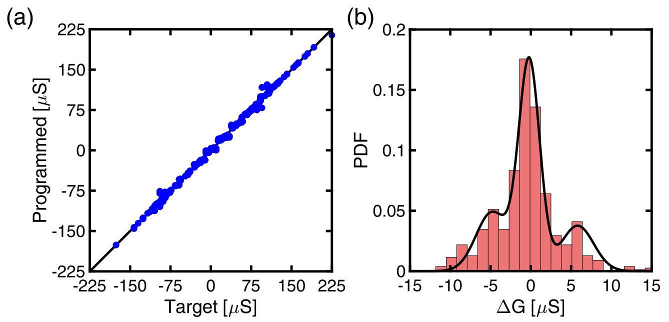

Fig. 11. Mapping accuracy of Iris dataset. (a) Correlation plot of target and programmed conductances. (b) Histogram of conductance errors, fitted with a 3-component Gaussian mixture with ensemble $\mu_G = -0.2\,\mu$S, $\sigma_G = 4.53\,\mu$S.

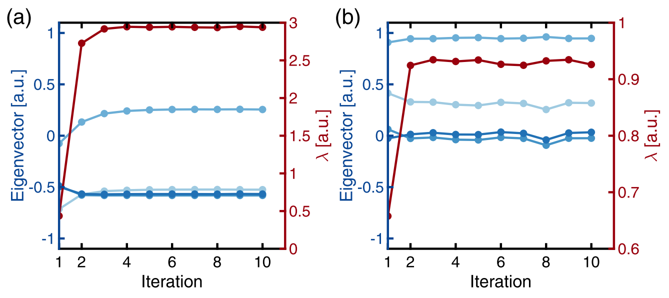

Fig. 12. IMC-PCA experiment on Iris dataset. (a) Power iteration transient for first PC. (b) Power iteration transient for second PC. Both PCs show a fast convergence due to the high ratio between the dominant and second eigenvalue.

ranges from a minimum $\sigma_G = 2.25\,\mu$S for L_8 to a maximum $\sigma_G = 7.66\,\mu$S for L_1. Level L_0, which was obtained by controlled reset, shows instead a moderately-low variability of $\sigma_G = 5.8\,\mu$S, as shown in Fig. 9c.

The dataset coefficients X_{ij} were mapped with the differential scheme shown in Fig. 10, where X_{ij} is mapped as the conductance difference $G_+ - G_-$. In this differential scheme, either G_+ or G_- (or both) were programmed in L_8, to benefit from the narrow distribution of high conductance levels and minimize the impact of large σ_G at low conductance. The differential mapping allows to encode equivalent conductance values between $-200\,\mu$S and $200\,\mu$S with $25\,\mu$S step, corresponding to an equivalent resolution of 4.1 bits.

The readout noise of our setup was characterized based on the programming characteristics of Fig. 9a and modeled as a Gaussian current noise with $\sigma_N = 0.8\,\mu$A.

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

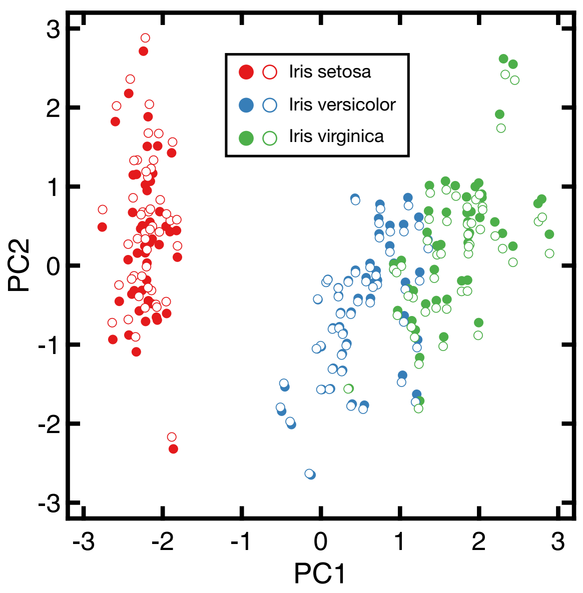

Fig. 13. Biplot representation of the Iris dataset in the space of the first two PCs (filled: FP-64 PCA, open: IMC-PCA). The *Iris setosa* cluster is revealed to be linearly separable from the *I. versicolor* and *I. virginica* clusters.

second PC was computed in 10 MVM iterations with 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 $\mu_G = 0.68\,\mu\text{S}$ with standard deviation $\sigma_G = 15.1\,\mu\text{S}$. IMC-PCA was carried out, with the first PC extracted in 10 iterations with CoSim = 0.97 (Fig. 15a) and the second PC extracted in 10 iterations with 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

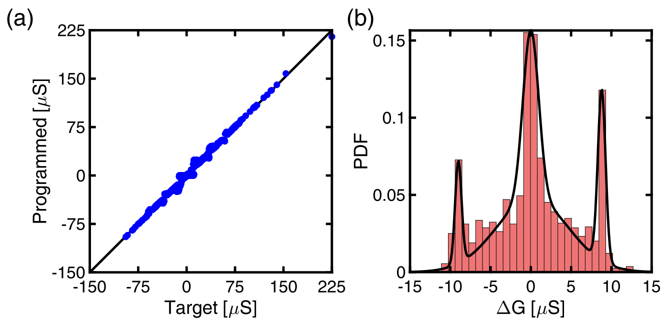

Fig. 14. Mapping accuracy of Glass dataset. (a) Correlation plot of target and programmed conductances. (b) Histogram of conductance errors, fitted with a 4-component Gaussian mixture with ensemble $\mu_G = 0.68\,\mu\text{S}$, $\sigma_G = 15.1\,\mu\text{S}$.

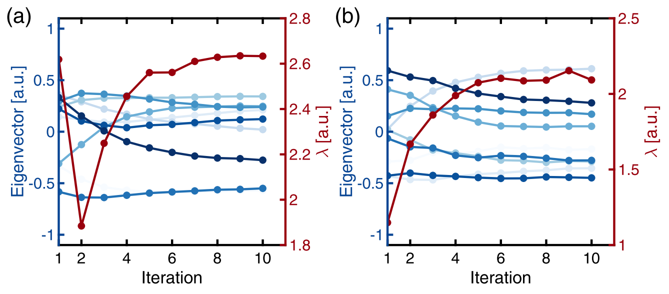

Fig. 15. IMC-PCA experiment on Glass dataset. (a) Power iteration transient for first PC. (b) Power iteration transient for second PC. While convergence appears to be slower than the Iris PCs, a limited number of iterations is required to achieve high CoSim of 0.97 and 0.91.

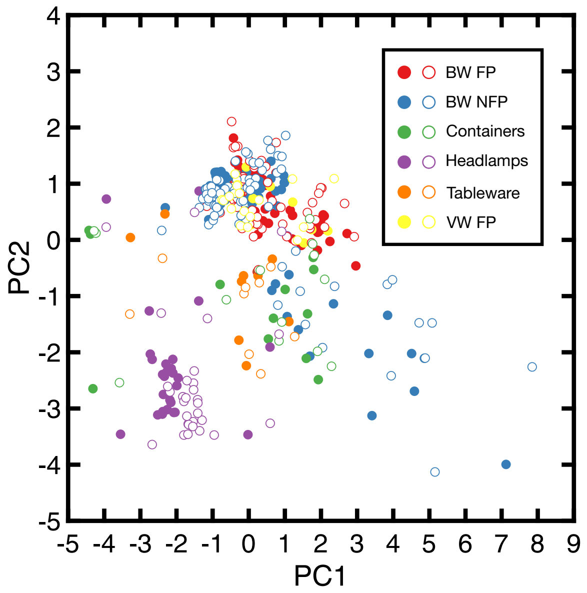

Fig. 16. Biplot representation of the Glass dataset in the space of the first two PCs (filled: FP-64 PCA, open: IMC-PCA). Differently from the Iris dataset, PCA does not allow to higlight any separable cluster among those recorded.

the mapping accuracy with $\mu_G = 0.29\,\mu\text{S}$, $\sigma_G = 8.40\,\mu\text{S}$ 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

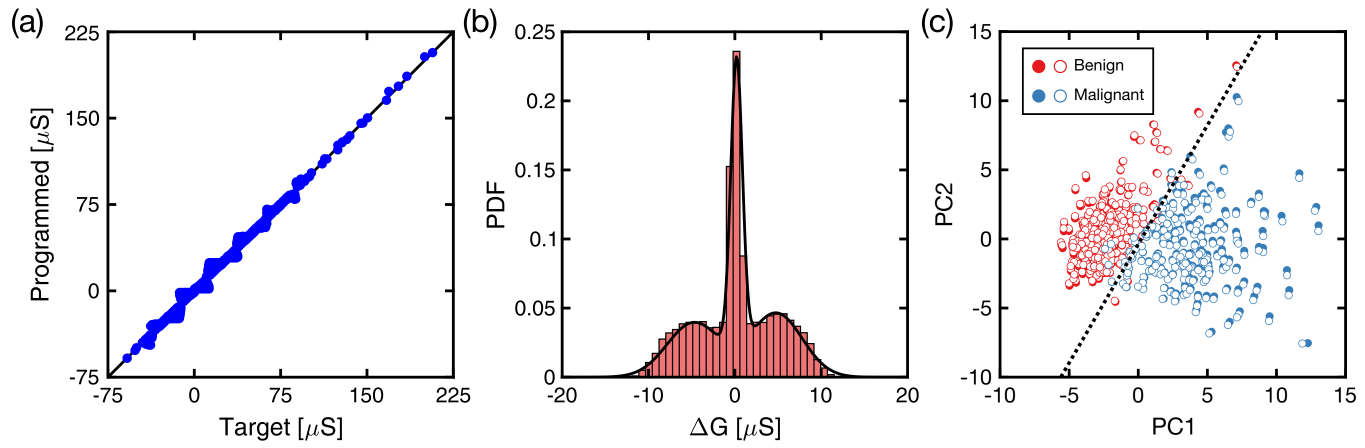

Fig. 17. IMC-PCA simulation on Wisconsin Breast Cancer dataset. (a) Correlation plot of target and programmed weights. (b) Histogram of conductance errors, fitted with a 3-component Gaussian mixture with ensemble $\mu_G = 0.29\,\mu\text{S}$, $\sigma_G = 8.40\,\mu\text{S}$. (c) Biplot representation of the Breast Cancer dataset in the PC space (filled: FP-64 PCA, open: IMC-PCA). The two classes of Benign and Malignant tumors are separated with 95.43% classification accuracy in IMC-PCA, close to FP64-PCA 95.61% accuracy.

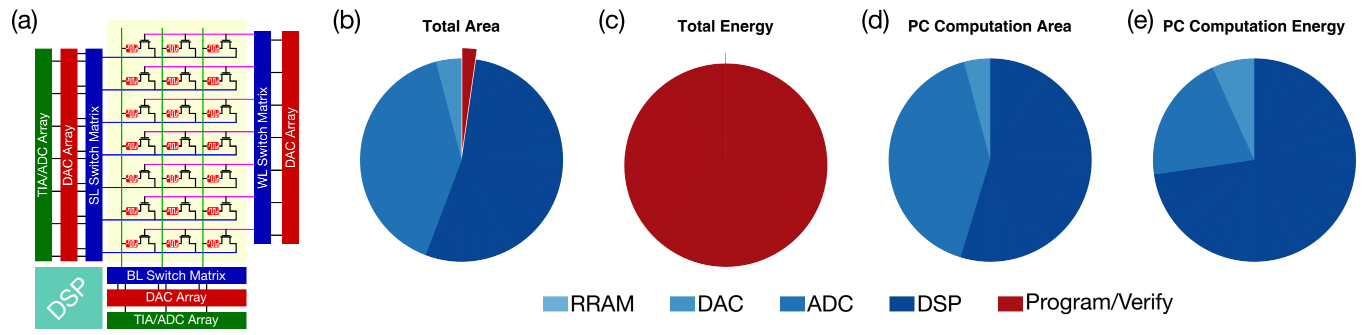

Fig. 18. (a) Proposed in-memory computing architecture, comprising a 1T1R array, reroutable DAC/ADC arrays and DSP. (b-e) Breakdowns of area and energy for the architecture components, namely RRAM memory array, DACs, ADCs, DSP and program/verify peripherals, considering both programming and PC computation (b,c) and PC computation only (d,e).

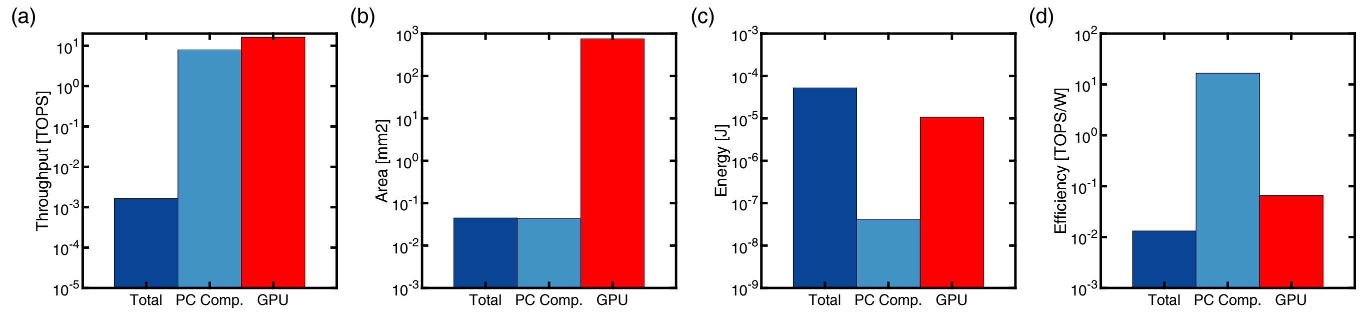

Fig. 19. Benchmark of proposed IMC architecture and algorithm on (a) throughput, (b) area, (c) energy and (d) energy efficiency, evaluated on the Wisconsin Breast Cancer dataset and compared with a commercial GPU [17].

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 $14\,\text{nm}$ 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-

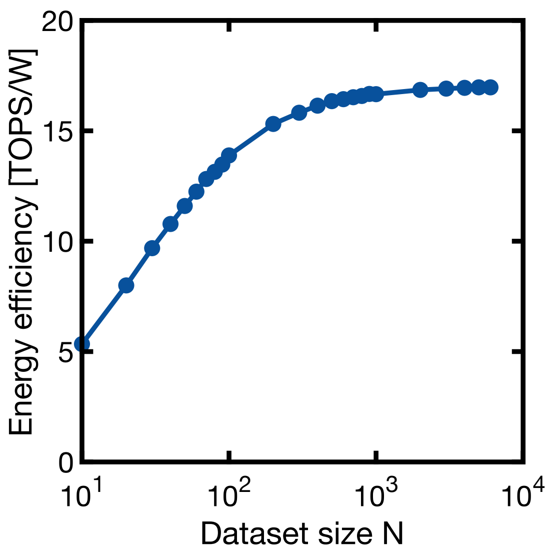

Fig. 20. Energy performance as a function of dataset size for the proposed IMC-PCA algorithm, for synthetically-generated Gaussian-distributed datasets.

ble latency (Fig. 19a) and better area and energy efficiency (Figs. 19b-c), with a $250\times$ increase in energy performance (Fig. 19d). Despite both metrics drop when considering also the programming procedure, moving to lower conductances and reducing the number of programming steps may help to alleviate the energy and latency burden. Moreover, additional latency and energy are to be expected also for GPUs when considering the continuous shuttling of data between memory and computing units [22]. Finally, system robustness to stuck-on/off devices, as well as overall programming precision, may be enhanced by employing redundancy and analog slicing techniques [23] at cost of increased energy and area overhead.

The energy efficiency of IMC largely benefits from scaling, due to the high parallelism of MVM (Fig. 20). The computational complexity of the in-memory PCA algorithm is $\mathcal{O}(p \cdot k \cdot mn)$ floating-point operations (FLOPs), where mn is 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:

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

where the first term accounts for the in-memory MVMs, and the second term accounts for the DSP steps. α is the typical energy dissipated on a single device during MVM, and β 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 $\mathcal{O}(n)$ 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 $\sim 17\,\mathrm{TOPS/W}$, which may be further improved by moving to higher resistive states. These results support IMC for energy-efficient accelerators of data-intensive ML tasks.

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 $0.25\,\mu\mathrm{m}$, and complemented with extensive simulations in a realistic benchmark framework. The IMC-based PCA shows an accuracy comparable to FP64, while benefitting 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

[1] E. Strubell, A. Ganesh *et al.*, *arXiv:1906.02243 [cs]*, Jun. 2019, arXiv: 1906.02243.
[2] M. A. Zidan, J. P. Strachan *et al.*, *Nature Electronics*, vol. 1, no. 1, pp. 22–29, 2018. doi: 10.1038/s41928-017-0006-8
[3] M. Prezioso, F. Merrikh-Bayat *et al.*, *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015. doi: 10.1038/nature14441
[4] I. Jolliffe, "Principal Component Analysis," in *Encyclopedia of Statistics in Behavioral Science*, B. S. Everitt and D. C. Howell, Eds. Chichester, UK: John Wiley & Sons, Ltd, Oct. 2005, p. bsa501. https://onlinelibrary.wiley.com/doi/10.1002/0470013192.bsa501
[5] G. H. Golub and C. F. Van Loan, *Matrix computations*. Baltimore: The Johns Hopkins University Press, 2013, oCLC: ocn824733531.
[6] H. Martens and M. Martens, *Multivariate analysis of quality: an introduction*. Chichester ; New York: Wiley, 2001.
[7] M. Andrecut, *Journal of Computational Biology*, vol. 16, no. 11, pp. 1593–1599, Nov. 2009. doi: 10.1089/cmb.2008.0221
[8] R. A. Horn and C. R. Johnson, *Matrix analysis*, 2nd ed. Cambridge ; New York: Cambridge University Press, 2012.
[9] K. A. Yeomans and P. A. Golder, *The Statistician*, vol. 31, no. 3, p. 221, Sep. 1982. doi: 10.2307/2987988
[10] H. Hotelling, *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933. doi: 10.1037/h0071325
[11] R. A. Fisher, *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936. doi: 10.1111/j.1469-1809.1936.tb02137.x
[12] E. Pérez, D. Maldonado *et al.*, *Microelectronic Engineering*, vol. 214, pp. 104–109, Jun. 2019. doi: 10.1016/j.mee.2019.05.004
[13] "Active Technologies RIFLE-SE." www.activetechnologies.it/products/non-volatile-memory-testers/rifle-se/
[14] V. Milo, A. Glukhov *et al.*, *IEEE Transactions on Electron Devices*, vol. 68, no. 8, pp. 3832–3837, Aug. 2021. doi: 10.1109/TED.2021.3089995
[15] D. J. Hand, G. J. McLachlan *et al.*, *Applied Statistics*, vol. 38, no. 2, p. 384, 1989. doi: 10.2307/2348072
[16] I. W. Evett and E. J. Spiehler, "Glass Identification Dataset," 1987.
[17] "NVIDIA Quadro RTX 8000." www.nvidia.com/en-us/design-visualization/quadro/rtx-8000/
[18] K. Bache and M. Lichman, "Breast Cancer Wisconsin diagnostic dataset," 2013.
[19] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information science and statistics. New York: Springer, 2006.
[20] P.-Y. Chen, X. Peng *et al.*, in *2017 IEEE International Electron Devices Meeting (IEDM)*. San Francisco, CA, USA: IEEE, Dec. 2017, pp. 6.1.1–6.1.4. doi: 10.1109/IEDM.2017.8268337
[21] A. Wang and C.-J. R. Shi, *Integration*, vol. 62, pp. 246–257, Jun. 2018. doi: 10.1016/j.vlsi.2018.03.010
[22] M. Horowitz, in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2014, pp. 10–14. doi: 10.1109/ISSCC.2014.6757323
[23] G. Pedretti, P. Mannocci *et al.*, *IEEE Transactions on Electron Devices*, vol. 68, no. 9, pp. 4373–4378, Sep. 2021. doi: 10.1109/TED.2021.3095433