Hyperdimensional Computing for Robust and Efficient Unsupervised Learning

Sanggeon Yun[1], Hamza Errahmouni Barkam[1], Paul R. Genssler[2], Hugo Latapie[3]
Hussam Amrouch[2,4,5] and Mohsen Imani[1,⋆]

[1]University of California Irvine, [2]University of Stuttgart, [3]CISCO Systems
[4]Munich Institute of Robotics and Machine Intelligence, [5]Technical University of Munich

⋆Corresponding Author: m.imani@uci.edu

2023 57th Asilomar Conference on Signals, Systems, and Computers | 979-8-3503-2574-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/IEEECONF59524.2023.10476861

***Abstract*—Clustering has emerged as a critical tool in diverse fields. Nevertheless, its high computational cost has been a persistent challenge, particularly for large-scale datasets. To address this, various compute-in-memory (CiM) approaches have been proposed, including the use of Ferroelectric FET (FeFET) technology due to its ultra-efficient and compact CiM architecture. However, non-idealities resulting from cell thickness and device temperature have impeded the scaling of FeFETs and thus hindered their potential to be used for clustering. In light of this, we propose a Hyper-Dimensional Computing (HDC) framework specifically for FeFET technology in the context of clustering. Our approach involves a cross-layer FeFET reliability model that captures the effects of scaling on multi-bit FeFETs, taking into account the impact of process variation and inherent stochasticity. We use two models in our HDC framework, a full-precision, ideal model for training, and a quantized error-impacted version for validation and inference. This iterative adaptation strategy helps to overcome the challenges associated with the non-idealities of FeFET technology.**

Our results demonstrate the proposed HDC framework performs better than traditional algorithms such as k-means and BIRCH. Moreover, our model can function as its ideal counterpart without noise, proving its potential to scale FeFET technology for clustering applications.

***Index Terms*—clustering, data science, computing in memory, FeFET, hyperdimensional computing**

I. Introduction

Clustering is a fundamental technique in unsupervised machine learning and data analysis, which aims to partition data points with similar features into distinct clusters. However, clustering in high-dimensional spaces presents several challenges, including the difficulty of determining the optimal number of clusters, sensitivity to noise and outliers, and the curse of dimensionality [1]–[3]. To address these issues, emerging research has proposed new clustering algorithms. Among these, hyperdimensional computing and vector symbolic architectures have shown great promise for clustering in high-dimensional spaces [4]. Hyperdimensional computing provides a potent framework for high-dimensional data representation and processing inspired by the way the brain processes information [5]–[11]. The approach employs high-dimensional vectors to represent patterns and concepts, capable of encoding complex relationships between different data points and handling noisy and incomplete data. Thus, it is an ideal method for clustering tasks where the relationships between data points are crucial and the data are imperfect.

Vector symbolic architectures offer a way to represent data as symbolic structures that can be manipulated and processed through algebraic operations. This approach enables the encoding of abstract and complex relationships between data points, rendering it useful for clustering tasks where higher-level concepts need identification. Additionally, vector symbolic architectures can manage missing data and noise, making them robust in the face of real-world data.

Hyperdimensional computing and vector symbolic architectures can further mitigate the curse of dimensionality, a significant problem in high-dimensional clustering. These methods can enhance clustering accuracy in high-dimensional spaces by providing a means to represent high-dimensional data as lower-dimensional symbolic structures that preserve the critical relationships between data points.

Recent research has proposed novel clustering algorithms that leverage innovative computing paradigms to overcome these challenges. In particular, computing in memory (CiM) has emerged as a promising approach, enabling efficient and scalable processing of large-scale clustering tasks. Ferroelectric field-effect transistor (FeFET) technology is a particularly attractive candidate for implementing CiM due to its non-volatile memory element, which can store and manipulate data using electrical signals.

Clustering algorithms process the same data repeatedly. With the rise of more and more data-heavy applications, CPU caches are overloaded and cannot fit the whole dataset at once. Thus, data has to be repeatedly transferred from lower memory levels. Such transfers cost orders of magnitude more energy than the computations themselves, highlighting a major bottleneck of the traditional Von Neumann architecture. To overcome this memory wall, the separation between computing and storage has to be resolved through novel computing in memory (CiM) architectures. However, traditional SRAMs consume a high chip area through their six or eight-transistor design. Their leakage increases power consumption and makes large SRAMs undesirable for low-power or embedded systems. FeFETs alleviate those concerns with their single-transistor design and non-volatility (i.e., minimal leakage). A ferroelectric (FE) layer is added to the transistor gate stack, which can be polarized to change the threshold voltage and, thus the current. Different logical states (i.e., bits) are encoded with different currents.

Combining FeFET technology with hyperdimensional computing can enhance clustering performance further, particularly in high-dimensional spaces. Hyperdimensional computing's ability to deal with most of the non-idealities introduced by CiM architectures and CiM can significantly reduce the memory and computation requirements for hyperdimensional computing clustering tasks, generate an ideal symbiotic relationship for several tasks [12], [13]. This paper aims to investigate the potential of hyperdimensional computing and vector symbolic architectures for clustering in high-dimensional spaces. We will present an exhaustive review of current clustering algorithms and their limitations given several conditions of noise, followed by an analysis of our framework's performance when introduced to FeFET-modeled non-idealities. The results are successful as traditional algorithms such as k-means and BIRCH suffer losses of 11.83% and 55.81% for the smallest noise probabilities, while our algorithm presents almost no quality loss. Our objective is to underscore the potential of hyperdimensional computing and computing in-memory architectures to advance clustering performance in high-dimensional spaces, opening avenues for more sophisticated and efficient data analysis and machine learning applications.

II. Related Work

Besides traditional clustering algorithms that are based on the distance or distribution of data points, a clustering algorithm HDCluster [4] that uses Hyperdimensional Computing (HDC) is also proposed. It encodes given data points into high-dimensional vectors called hypervectors. And uses a well-defined set of HD operations to perform clustering. The HD operations are known for their robustness in the presence of failures, making the HDCluster robust on any noise during clustering computation [14]. It also has shown higher robustness in terms of the dimensionality of given data points compared to the k-means algorithm which indicates that the HDCluster can efficiently handle complex datasets [4].

As for the traditional clustering algorithms in the literature, k-means [15] is a distance-based clustering algorithm that clusters given data points into k clusters by finding k cluster centers called centroids, which has a similar structure of HDCluster but for lower dimensionality. It assigns a data point to a cluster where the nearest centroid is located. Since the k-means algorithm was first proposed, various heuristic algorithms such as Lloyd's algorithm have been proposed that enable the k-means algorithm to be applied to large datasets [16]. It is well-used in various domains [17], [18] by today.

Another popular clustering algorithm is BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [19], which also consists of a distance-based clustering algorithm that was proposed to resolve issues of previous clustering algorithms on computation inefficiency and outliers. BIRCH makes clustering decisions without scanning all data points which allows it to process a very large dataset that even cannot load on main memory. Since it does not treat every data point equally important, it also has the advantage of dealing with outlier data points. Its strong advantage in processing a very large dataset made it widely used in commercial products [20], [21].

In [22], two implementations of the K-means clustering algorithm are described: one on a Memristive Logic Array-based CiM and the other on a Machine Learning-specific Transport-Triggered Architecture (TTA) processor, which is fabricated on 28nm FDSOI technology and operates at 0.35V. The Memristive CiM implementation outperforms the TTA processor, achieving a speedup of 5.7 times and using 2.3 times less energy. However, the average power consumption of the Memristive Logic Array is 2.5 times higher than that of the TTA processor. DUAL [23] is a digital-based processing-in-memory (PIM) architecture that performs parallel encoding and clustering computation over the encoded hypervectors stored in memory. DUAL uses two computation blocks, a data block, and a distance block, to support search-based and arithmetic operations. The architecture maps hierarchical clustering into PIM acceleration using row-parallel operations. DUAL eliminates the use of large ADC/DAC blocks, resulting in high throughput/area and scalability. The design also eliminates internal data movement overheads by using interconnects for bit-serial/row-parallel data transfer between the data and distance blocks.

III. Computing in memory with FeFET

FeFET technology is a promising candidate for on-chip memories. Its full CMOS-compatibility [24] makes it easy to integrate alongside regular transistors and tightly coupled logic and memory. Compared to traditional SRAM, FeFET is non-volatile and thus has very little leakage during idle and compared to other NVMs low read/write energy, and a short read latency [25], [26]. As depicted in Figure 1, a HfO_2 layer is added to the traditional gate stack. This layer is typically 10nm thick, ferroelectric, and can be polarized by a voltage pulse to the gate terminal. The voltage pulse flips the domains inside the FE layer which, in turn, impacts the drain-source current I_D. The difference in I_D is mapped to the low V_{th} and high V_{th} states and then to logical states of '0' and '1'.

The FE layer is inherently stochastic and not all domains flip at the same time. This creates a multi-bit storage device, the intermediate I_D are mapped to intermediate logical values [27]. However, the stochasticity introduces a high variation in addition to the process variation of the device itself. This variation is further increased if the thickness of the FE layer is scaled down to 3nm [27]. The highly-scaled FE layer has fewer domains and each becomes more impactful. To counteract the increase in variation, [28] proposed to extend the gate stack with an additional back gate for read operations. Such a design has two contradictory effects. On one hand, the impact of variation is increased because of the increased distance between the back gate and the FE layer. On the other hand, the memory window (MW), the difference between the most and least polarized states, is increased by up to 17V as shown in Figure 2. A larger MW provides higher margins

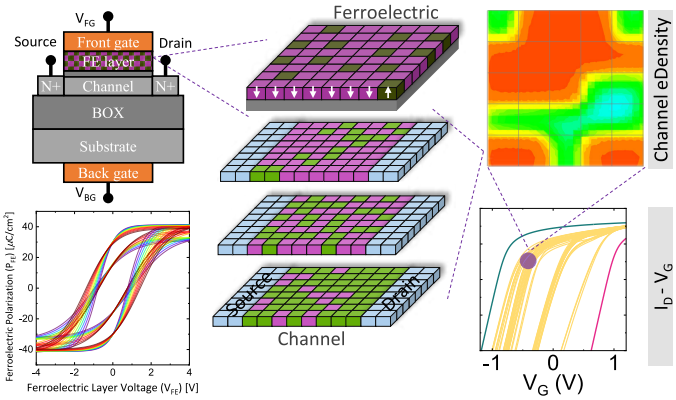


Fig. 1: A FeFET is based on a regular transistor but with an additional ferroelectric (FE) layer. This layer comprises domains, which can be polarized by a voltage pulse. The inherent stochasticity of the polarization process causes variation and a spread in the sensed current I_D during a read operation [29].

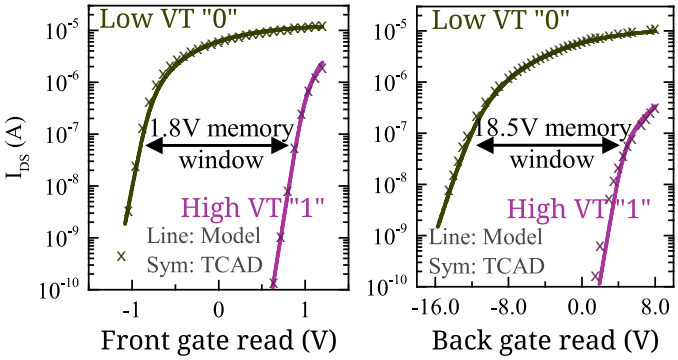


Fig. 2: Adding the back gate for read operations increases the difference between the low V_{th} and high V_{th} states significantly [29].

against design-time and run-time variation. In summary, this increase in the MW is more substantial than the other impact of variation from the back gate. Hence, the overall robustness of the read operation is increased. The improved robustness opens the door for highly-scaled FeFET devices, which otherwise would be too noisy [27].

A. Our FeFET modeling

To accurately model the reliability, we start with the multi-physics simulator technology CAD (TCAD) at the device level [29]. The device is a ferroelectric capacitor built from the same materials as a transistor. Using TCAD, a model is carefully calibrated [27] to reproduce experimental measurements of a commercial 22nm FDSOI technology [30]. The parameters of the TCAD models for the incorporated FE layer are remnant polarization, saturation polarization, and coercive field. Because these parameters describe the FE layer itself, they can be extracted from a capacitor model and applied to a transistor model. To model the reliability at different temperatures (27°C and 80°C), the same methodology is applied repeatedly. Only the parameters describing the FE layer have to be adapted since the existing transistor models already capture the effects of the underlying device. Hence, our additional modeling is indispensable to account for the degradation in the FE layer induced by an increased temperature. As described above, a FeFET can be used as a multi-bit storage. To extract the different I_D, the gate voltage range is swept. In addition, Monte Carlo simulations are performed to capture process variation and the inherent stochasticity. As a result, the I_D distributions for each logical state are known and can be used for further modeling at the circuit level.

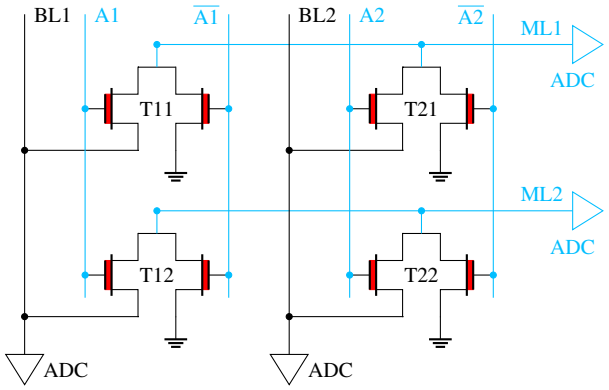
B. FeFET-based TCAM Cells

A FeFET-based TCAM cell comprises two FeFET devices [31]. The left-hand FeFET is programmed into the inverted state of the right-hand FeFET. The input value is applied on the 'A' line and inverted on the '$\overline{A}$' line. If the input value and the state match, both FeFET devices are in a non-conducting state. For example, the stored value is a '1', i.e., the left-hand FeFET is in the high V_{th} state whereas the right-hand is in low V_{th} state. If the input is a '1', the 'A' line (connected to the left-hand FeFET in high V_{th}) activates the FeFET but it does not conduct because of its high V_{th} state. On the right side, both state and input are opposite which leads to the same result. However, if the input and state mismatch, then the FeFETs conduct.

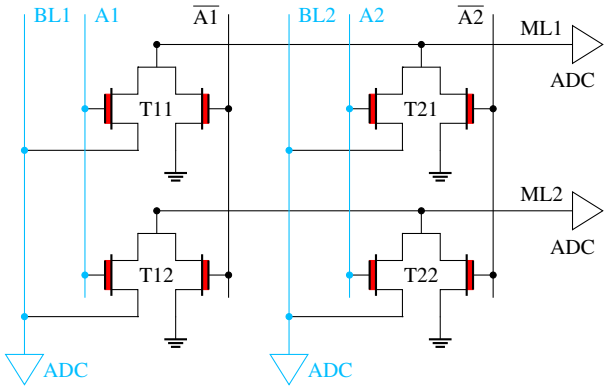
C. Circuit Design

Multiple TCAM cells are connected by a matching line (ML) as shown in Figure 3a. This ML is charged before the inputs are applied and the bundle line (BL) is tied to the ground. Depending on the number of conducting (mismatching) cells, the ML is discharged fast (many mismatches) or slow (few mismatches). The rate of this discharge can be measured, e.g., with a CSRSA [31] or a FeFET-based comparator [32]. The impact of process variation and the resolution of a comparator limits the number of TCAM cells attached to an ML. In [31], 15 TCAM cells were employed. To store and compare whole hypervectors, multiple such arrays are combined. The partial similarity scores are added through traditional CMOS logic. The full similarity scores are then used by the clustering algorithm to determine new centers.

The second major operation in the clustering algorithm is the creation of new center vectors. This bundling operation is very data intensive in a traditional von Neumann architecture because for each dimension a counter has to be kept. In addition, all vectors for the new center have to be processed, i.e., transferred from the memory to the compute unit where they compete with the counters for the limited caches. With the proposed in-memory architecture, the new center vectors can be derived without data transfers. For the bundling operation, the states of the left-hand FeFETs have to be read and averaged. To read, the read voltage is applied to the 'A' line, which activates the left-hand FeFETs as shown in Figure 3b. The ML is tied to the supply voltage and, depending on the



(a) For a search operation, BL is set to ground, the match line (ML) precharged, and the inputs are applied to the A lines. Each ML then presents the similarity search result.



(b) For a bundling operation, the match line (ML) is set to VDD, the A lines are set with the read voltage, and the summation is done on the bundle line (BL). If an ML is set to ground, the TCAM cells do not contribute enabling the summation of vectors only belonging to one cluster.

Fig. 3: Architecture for a FeFET-based TCAM array to facilitate search as well as bundling operations.

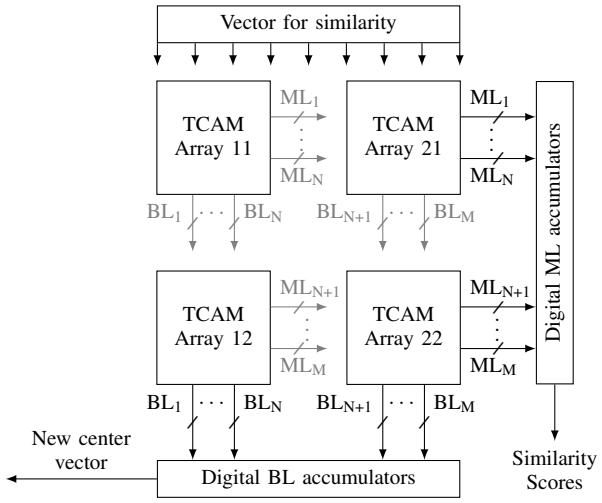


Fig. 4: Architecture for the whole vector storage, which comprises many TCAM arrays (only four shown here). The storage can compute the similarity to a provided vector and bundle a selectable number of stored vectors into a new vector.

state of the FeFET, a smaller or larger current flows to the BL. These currents are added through Kirchoff's law on the BL and sensed by the connected ADC. Since not all vectors are part of the new center vector, only for the selected ones the ML is tied to the supply voltage. The ML of the not selected vectors is tied to a high impedance and no current flows. Similar to the ML, the number of connected TCAM cells is limited and requires multiple arrays to store a large amount of vectors.

D. Storage Design

Each individual TCAM array can only store a limited number of vector components. Multiple such arrays are combined to form the whole storage as shown in Figure 4. The digital output of the MLs for one vector is combined with accumulators to get the total similarity of the stored vector to the query. One such accumulator exists per vector and holds the similarity score. A second set of accumulators combines the digital bundle results for each component. After normalizing the accumulated values, they return a new center vector.

IV. Clustering with Hyperdimensional Computing

We propose a new brain-inspired clustering algorithm that clusters input data into a high-dimensional space. There are many properties in the hyperspace that are suitable for clustering, most specifically the ability to maintain the distance between inputs thanks to the precise kernel approximation given by the clear HDC algebra. Also, having the information of each feature spread through each one of the dimensions of the vector and having the feature vectors be quasi-orthogonal generates holographic hypervectors that are robust against random noise, making it a suitable algorithm for Computing in Memory with unreliable variations on its values. The algorithm is broken down into different blocks, starting with encoding the input data, generating the centroids, projecting them, and adding some adaptability. Finally, we describe the updates to the algorithm applied in order for the modeling to be faithful to the technology being deployed, which consists of model quantization, the introduction of noise, and retraining to make the centroids learn to adapt to the non-idealities previously described.

As described in Figure 5, we encode by associating each feature of the input (❶) with a feature hypervector that is randomly generated from a Gaussian distribution (❷), resulting in several hypervectors each one containing the information of one feature. The resulting hypervectors are combined by bundling (sum element-wise). Next, a model is initialized with 5 given parameters: B, D, K, and M^c, where B indicates the number of bits, D indicates the number of dimensions, K indicates the number of clusters, and M^c indicates current matrix containing mapping values for computing similarities.

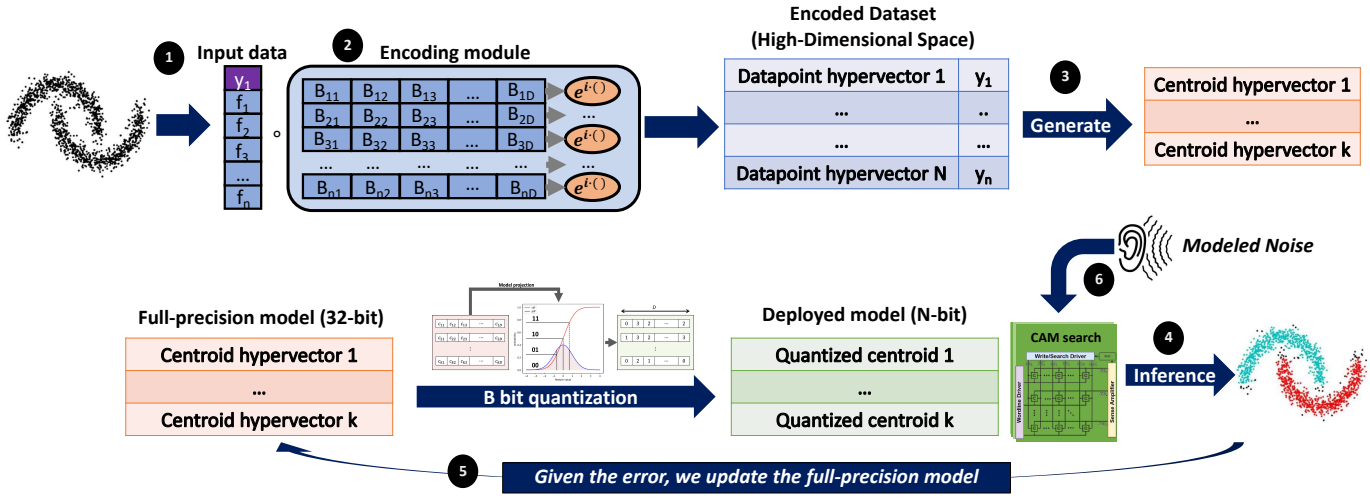


Fig. 5: Architecture of our proposed framework. We begin by having an encoding block that associates the input features (❶) to each of the basis (❷) and generates hypervectors for the whole dataset. Then we generate the centroid hypervectors (❸), adapt them to the CiM requirements (❹), and update them adaptively (❺).

The model generates K initial center hypervectors for each cluster using a given train set X to generate the initial centroids. In order to generate such hypervectors, the model first encodes the given train set X into D dimensional hypervectors $\phi(X)$. Then, the model applies the k-means++ algorithm on the encoded train set $\phi(X)$. The selected hypervectors are used as initial cluster centroids $\{\vec{H}_1, \vec{H}_2, \ldots, \vec{H}_K\}$, as seen in Figure 5 (❸).

Next, we need to adapt and update the centroids with new data (❺). Each adaptive centroid update step uses an encoded train set $\phi(X)$ that is also used in the initial centroid generation step. For each hypervector $\vec{h}_j \in \phi(X)$, it updates centroid hypervectors as follows:

$$\vec{H}_i \leftarrow \frac{\sum_{\vec{h}_j \in \phi(X)} \alpha_{ij} \vec{h}_j}{\sum_{\vec{h}_j \in \phi(X)} \alpha_{ij}}$$

$$\text{where } \alpha_{ij} = \begin{cases} 1 & \text{if } i = \text{argmax}(\{\delta(\vec{H}_1, \vec{h}_j), \ldots, \delta(\vec{H}_K, \vec{h}_j)\}) \\ 0 & \text{otherwise} \end{cases}$$

M^c is used in δ, computing the similarity between the given hypervectors, which will be described later. At the end of each adaptive centroid update, $\vec{H}_i^q$ is computed from updated $\vec{H}_i$ by the centroid projection step.

A. Hyperdimensional Computing - Adaption to FeFET Framework

We proceed to describe the adaptation of the hardware modeling to our framework, which consists of a modified similarity function based on our designed CAM, bit-precision quantization, to convert our full-precision model to an N-bit precision that fits the FeFET-based CiM used and the noise introduction to the model.

In order to have a B-bit(s) precision model, the full precision cluster centroids $\vec{H}_i$ needed to be quantized (❹). The quantization is conducted by mapping each continuous value of the component in the hypervectors to one of the B-bit(s) symbols. Since feature values do not generally follow uniform distribution, it calculates scores of the hypervector components and uses the cumulative normal distribution function (CDF) to quantize feature values. In formally, for the CDF of normal distribution Φ, each component h_i in a hypervector $\vec{H}$ with mean value of m and standard deviation value of σ, each component h_i is quantized as $\lfloor \Phi(\frac{h_i - m}{\sigma}) \times 2^B \rfloor$. Through this, quantized cluster centroid hypervectors $\vec{H}_i^q$ are computed from $\vec{H}_i$.

For the purpose of applying noise (❻), the modeled probability is applied and defined as **state shift probability**. It consists of making a value v increase or decrease to $v+1$ or $v-1$. Each value in the quantized hypervector has the same probability of random value changing. For instance, if the random changing probability is p, a value v in the hypervector will be changed to the value of $v+1$ in $p/2$ probability and $v-1$ in $p/2$ probability. But, if $v-1$ is less than 0 or $v+1$ is greater than $2^B - 1$, only an increase or decrease will be applied in p probability, respectively. The random value change is conducted between the model projection and the inference (for the test set) step. Changed values remain still until the next model projection step. When the "noise-only inference" option is set, value changes are not conducted on the model currently in the training process. The noise will be applied to the copied model during the iterative training.

Last but not least, we want to specify that traditional HDC algorithms deal with the similarity between two hypervectors with the inner product, i.e., the dot product. However, on most CiM devices, that is not available; instead, we have a function that discharges current depending on the two symbols being compared (❹). To apply this to our framework, we used the following structure:

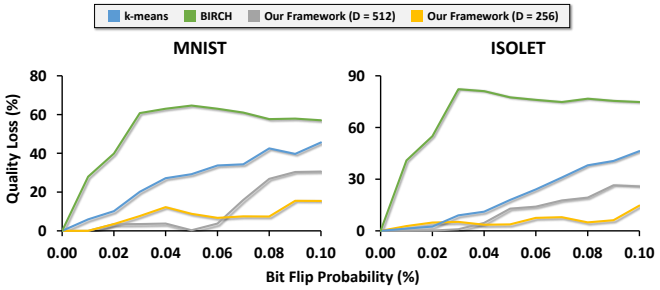


Fig. 6: Testing the accuracy of the different algorithms given different random bit flip probabilities.

- Given two B-bit(s) hypervectors H_1 and H_2, current similarity $\delta(H_1, H_2)$ is computed using $\vec{d} = |H_1 - H_2|$ and M^c indicating current matrix containing mapping values for computing similarities. Now, the similarity is computed as follows:

$$\delta(H_1, H_2) = \sum_{i=0}^{D-1} M^c_{\vec{d}_i}$$

Note that $0 \leq \vec{d}_i < |M^c| = 2^B$.

V. Evaluation

Computing in memory is a promising approach for the development of high-performance computing systems with low power consumption. However, its implementation is hindered by several challenges, such as the presence of noise and non-ideal conditions. Therefore, it is crucial to develop algorithms that are able to handle such challenges and provide reliable results. This is why we evaluated the performance of three clustering algorithms for computing in memory: k-means, birch, and our proposed solution, for different bit-precisions.

A. Robustness analysis of clustering algorithms against random noise

To make the evaluation fair, we proceed to use random noise applied equally to all the bits of the input features of the models. Our results show in Figure 6 that under ideal conditions, all three algorithms perform similarly. However, once the smallest noise is introduced, k-means and birch fail to keep up with performance, having the first one losing 11.83% clustering quality and the latter 55.81%, while our framework is able to handle noise and still providing accurate results with 0% quality loss. As we increase the noise, k-means begins to close up to 50% quality loss, while in our best case, for dimension 254, we only lose 15%.

B. Hollsitic FeFET non-ideal noise analysis

To further explore the performance of our algorithm under different FeFET cell conditions. Table I reports the noise values for different cases of the CiM FeFET cells, bit-precision (3 or 4 bits), the gate of the cell (front or back), the thickness of the cell (10 or 3 nm), and the temperature (27 or 80C). As shown in Figures 7 and 8, we observe that our algorithms

TABLE I: Noise levels for different thicknesses of the ferroelectric layer, bit precision, and temperatures.

HfO_2 Thickness (nm)	Precision (bit)	Front/Back Gate Read	Temepera-ture (°C)	Modeled Noise (%)
10	3	B	27	0.04
10	3	B	80	0.74
10	3	F	27	0.10
10	3	F	80	1.36
10	4	B	27	8.88
10	4	B	80	20.44
10	4	F	27	11.13
10	4	F	80	24.21
3	3	B	27	0.73
3	3	B	27	5.88
3	3	F	27	42.43
3	4	F	27	20.30
3	4	F	80	36.71

perform successfully and at a similar performance to its full precision counterpart regardless of the non-ideal noise that ranges from 0.1% to 42.43%. HDC algorithms tend to be robust and consistent through dimensionality, meaning that more than often, increasing the dimensionality does not increase the quality or accuracy, which is shown in the figures. Furthermore, we also introduce the noise during training, to see if the model is able to learn to deal not only with the loss of information by quantizing or reducing the bit-precision but also with the noise. The results show that there is no need to introduce the noise during training as there are no significant improvements for higher bit-precisions.

C. Complete noise analysis

Next, we studied the effect of different noise probabilities, bit precisions, and dimensionalities to test the effectiveness of our algorithm that would result from different technologies. Our findings in Figures 9 indicate that in order to tackle the noise present in computing in memory, a bit precision of at least 3 or 4 bits is necessary to perform as well as the full-precision counterpart. Additionally, increasing the dimensionality of the hypervectors improves performance, but higher bit precision can compensate for lower dimensionality. These results highlight the importance of developing algorithms that are robust to noise and non-ideal conditions in computing in memory. By doing so, we can unlock the full potential of this promising approach and pave the way for the development of highly efficient computing systems.

VI. Conclusion

In this paper, we introduce a hyperdimensional clustering algorithm into a framework able to perform FeFET-based Computing in Memory. The algorithm was evaluated with several FeFET non-idealities and conditions. Our outcomes demonstrate that our approach provides the highest robustness to random noise and is able to maintain performance on a highly-scaled FeFET-based CiM. Our results demonstrate that highly-scaled FeFET realizing 3-bit and even 4-bit, can withstand any noise given high dimensionality during inference.

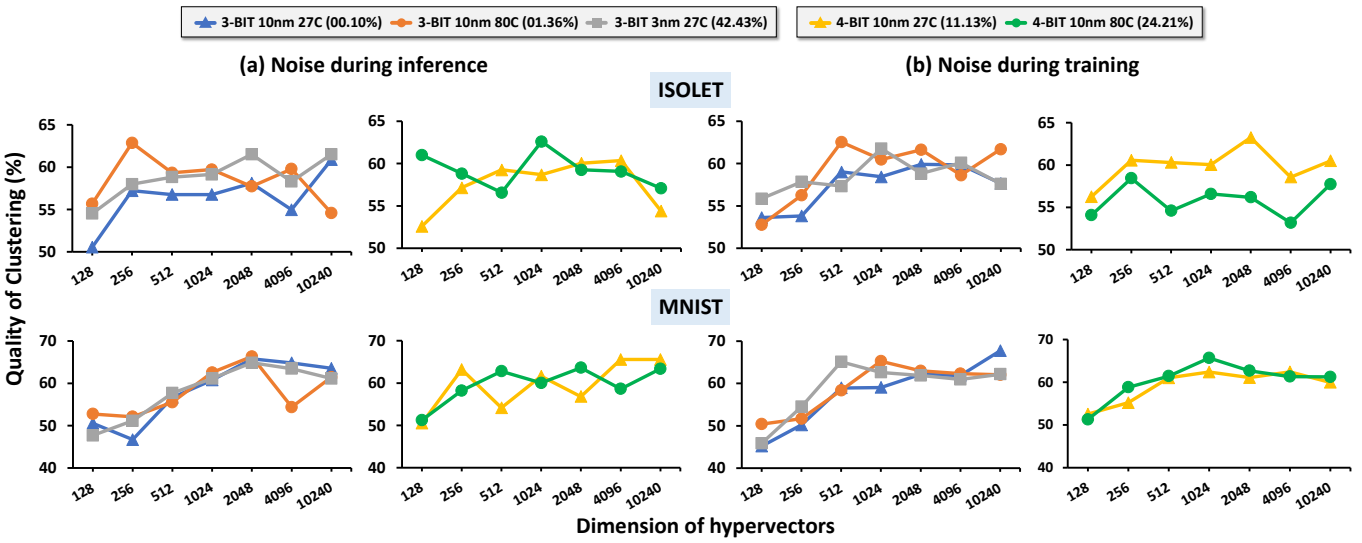


Fig. 7: Testing the quality of our framework for several thicknesses and bit-precision for FeFET with a front gate for ISOLET and MNIST datasets respectively.

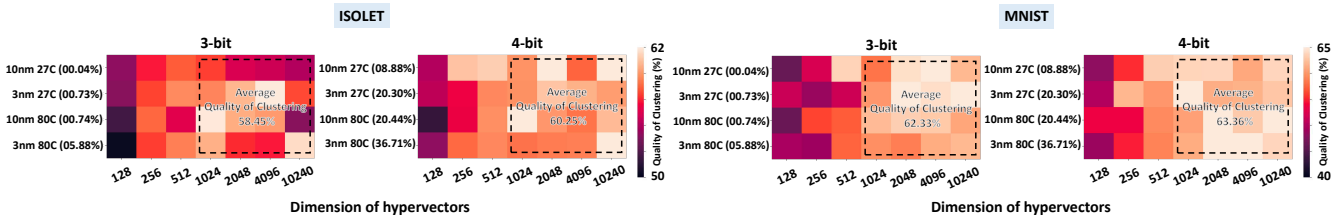


Fig. 8: Testing the quality of our framework on MNIST and ISOLET for several thicknesses and bit-precision for FeFET with a back gate. We introduce the noise during training and during inference to test the ability of the model to learn to deal with the noise.

ACKNOWLEDGMENT

The authors would like to thank Yogesh S. Chauhan, Shubham Kumar, and Swetaki Chatterjee, Kai Ni, Albi Mema, Simon Thomann, and Om Prakash for their support with the FeFET and FDSOI device modeling and calibration as well as for the insightful discussions. This work was supported in part by National Science Foundation #2127780, #2319198, #2321840 and #2312517, Semiconductor Research Corporation (SRC), Office of Naval Research, grants #N00014-21-1-2225 and #N00014-22-1-2067, the Air Force Office of Scientific Research under award #FA9550-22-1-0253, and a generous gift from Cisco and Xilinx. This research was partially supported by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart.

REFERENCES

[1] S. Im, M. M. Qaem, B. Moseley, X. Sun, and R. Zhou, "Fast noise removal for k-means clustering," 2020.

[2] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Information Sciences*, vol. 622, pp. 178–210, 2023.

[3] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *Engineering Applications of Artificial Intelligence*, vol. 110, p. 104743, 2022.

[4] M. Imani, Y. Kim, T. Worley, S. Gupta, and T. Rosing, "Hdcluster: An accurate clustering using brain-inspired high-dimensional computing," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1591–1594, IEEE, 2019.

[5] A. Hernandez-Cane, N. Matsumoto, E. Ping, and M. Imani, "Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 56–61, IEEE, 2021.

[6] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey, "Exploring hyperdimensional associative memory," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 445–456, IEEE, 2017.

[7] M. Imani *et al.*, "Control of gene regulatory networks using bayesian inverse reinforcement learning," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 16, no. 4, pp. 1250–1261, 2018.

[8] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "Voicehd: Hyperdimensional computing for efficient speech recognition," in *2017 IEEE international conference on rebooting computing (ICRC)*, pp. 1–8, IEEE, 2017.

[9] P. R. Genssler and H. Amrouch, "Brain-inspired computing for wafer map defect pattern classification," in *IEEE International Test Conference (ITC'21)*, 10 2021.

[10] P. R. Genssler and H. Amrouch, "Brain-inspired computing for circuit reliability characterization," *IEEE Transactions on Computers*, vol. 71,

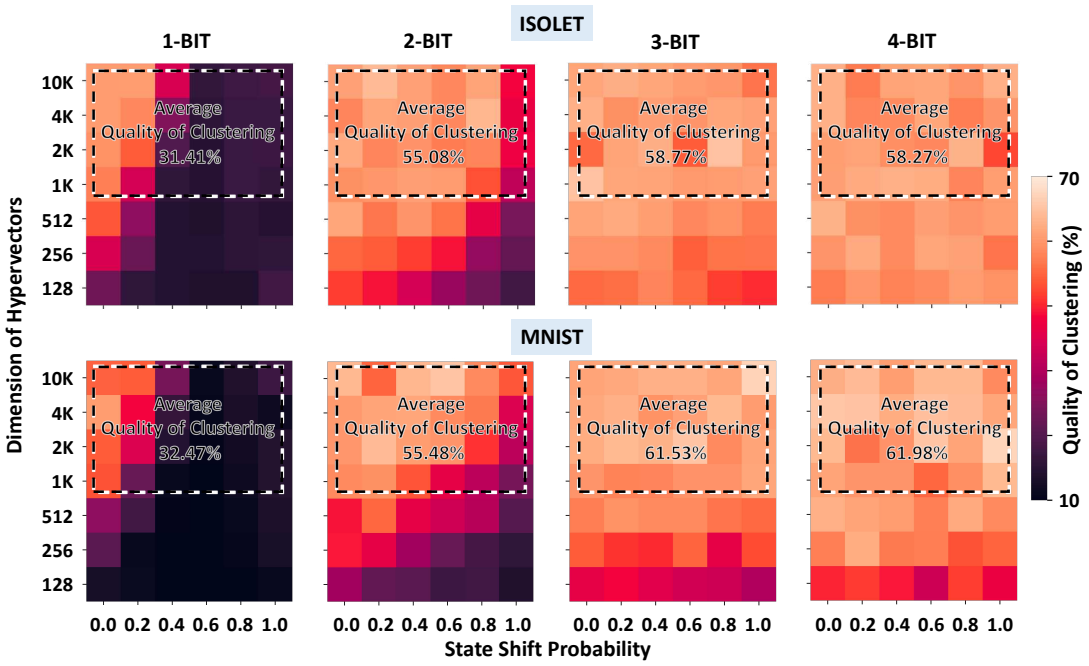


Fig. 9: Study of the clustering performance for different dimensionality and noise probabilities for the state shift error on the MNIST and ISOLET datasets.

pp. 3336–3348, 2 2022.

[11] P. R. Genssler, H. E. Barkam, K. Pandaram, M. Imani, and H. Amrouch, "Modeling and predicting transistor aging under workload dependency using machine learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2023.

[12] H. E. Barkam *et al.*, "Hdgim: Hyperdimensional genome sequence matching on unreliable highly scaled fefet," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2023.

[13] A. Kazemi, F. Müller, M. M. Sharifi, H. Errahmouni, G. Gerlach, T. Kämpfe, M. Imani, X. S. Hu, and M. Niemier, "Achieving software-equivalent accuracy for hyperdimensional computing with ferroelectric-based in-memory computing," *Scientific Reports*, vol. 12, p. 19201, Nov 2022.

[14] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory hyperdimensional computing," *Nature Electronics*, vol. 3, no. 6, pp. 327–337, 2020.

[15] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[16] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[17] O. J. Oyelade, O. O. Oladipupo, and I. C. Obagbuwa, "Application of k means clustering algorithm for prediction of students academic performance," *arXiv preprint arXiv:1002.2425*, 2010.

[18] P. Govender and V. Sivakumar, "Application of k-means and hierarchical clustering techniques for analysis of air pollution: A review (1980–2019)," *Atmospheric pollution research*, vol. 11, no. 1, pp. 40–56, 2020.

[19] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," *ACM sigmod record*, vol. 25, no. 2, pp. 103–114, 1996.

[20] G. Pitolli, L. Aniello, G. Laurenza, L. Querzoni, and R. Baldoni, "Malware family identification with birch clustering," in *2017 International Carnahan conference on security technology (ICCST)*, pp. 1–6, IEEE, 2017.

[21] J. Garcia and A. Brunstrom, "Clustering-based separation of media transfers in dpi-classified cellular video and voip traffic," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2018.

[22] L. Koskinen, J. Tissari, J. Teittinen, E. Lehtonen, M. Laiho, and J. H. Poikonen, "A performance case-study on memristive computing-in-memory versus von neumann architecture," in *2016 Data Compression Conference (DCC)*, pp. 613–613, 2016.

[23] M. Imani, S. Pampana, S. Gupta, M. Zhou, Y. Kim, and T. Rosing, "Dual: Acceleration of clustering algorithms using digital-based processing in-memory," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 356–371, IEEE, 2020.

[24] S. Beyer *et al.*, "Fefet: A versatile cmos compatible device with game-changing potential," in *IEEE IMW*, 2020.

[25] K. Ni *et al.*, "Ferroelectric ternary content-addressable memory for one-shot learning," *Nature Electronics*, 2019.

[26] P. R. Genssler, V. Van Santen, J. Henkel, and H. Amrouch, "On the reliability of fefet on-chip memory," *IEEE Transactions on Computers*, vol. 71, pp. 947–958, 3 2022.

[27] S. Chatterjee *et al.*, "Comprehensive variability analysis in dual-port fefet for reliable multi-level-cell storage," *IEEE TED*, 2022.

[28] H. Mulaosmanovic, D. Kleimaier, S. Dünkel, S. Beyer, T. Mikolajick, and S. Slesazeck, "Ferroelectric transistors with asymmetric double gate for memory window exceeding 12 v and disturb-free read," *Nanoscale*, vol. 13, no. 38, pp. 16258–16266, 2021.

[29] H. E. Barkam, S. Yun, P. R. Genssler, Z. Zou, C.-K. Liu, H. Amrouch, and M. Imani, "HDGIM: Hyperdimensional genome sequence matching on unreliable highly-scaled FeFET," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2023*, 3 2023.

[30] K. Ni *et al.*, "On the channel percolation in ferroelectric fet towards proper analog states engineering," in *IEEE IEDM*, 2021.

[31] S. Thomann, P. R. Genssler, and H. Amrouch, "HW/SW co-design for reliable tcam-based in-memory brain-inspired hyperdimensional computing," *IEEE Transactions on Computers*, 2023.

[32] S. Thomann, H. L. G. Nguyen, P. R. Genssler, and H. Amrouch, "All-in-memory brain-inspired computing using FeFET synapses," *Frontiers in Electronics*, vol. 3, 2 2022.