Associative clustering with analog content-addressable memory

Sara Ameli
Forschungszentrum Jülich GmbH, PGI-14,
Jülich,, Germany

Jiaao Yu
Forschungszentrum Jülich GmbH, PGI-14,
Jülich,, Germany

John Paul Strachan
Forschungszentrum Jülich GmbH, PGI-14, Jülich, Germany
RWTH Aachen University, Aachen, Germany

Abstract

The backbone of clustering algorithms is similarity search, which requires accessing all data in explicit memory. The physical separation between memory and computing units in the von Neumann architecture leads to costly data transfer, making it inefficient for processing large amounts of data. By facilitating parallel search within memory, Content Addressable Memories (CAMs) offer fast in-memory search. Additionally, analog CAMs provide search capabilities in the analog domain, thereby alleviating the need for analog-to-digital conversion (ADC).

Developing a Python-based circuit model that bridges between circuit design and algorithm development, we demonstrate how to obtain a single-point exact-match response from a 6T2M differentiable-CAM (diff-CAM) circuit. A single-point exact match provides the possibility of a full analog similarity search. Using this analog response and adaptive programming of diff-CAMs, we introduce the novel concept of using analog CAM for in-memory similarity search in clustering tasks. We demonstrate the performance of our approach for K-means clustering and fuzzy C-means clustering. The results show comparable clustering metrics to other similarity functions, with the advantage of in-memory computations. By demonstrating a mechanism for adaptive programming of diff-CAMs, we demonstrate how they can support a trainable search architecture, positioning them as a foundational component for end-to-end trainable in-memory computing pipelines.

Keywords

Content-addressable memory, in-memory computing, and clustering.

1 Introduction

The rapid expansion of data in domains such as natural language processing, computer vision, and bioinformatics highlights the growing demand for scalable, power-efficient hardware solutions. Developing such hardware has become increasingly critical to ensure sustainable, high-performance computation. Among the core operations in these applications, clustering depends on similarity search across large datasets—a process that is computationally expensive on traditional architectures such as general-purpose graphics processing units (GPGPUs). This inefficiency arises from the physical separation between memory and compute units, which leads to high energy and latency costs associated with data movement, as well as scalability concerns related to memory capacity.

Digital computing architectures offer high accuracy but incur significant overhead in area and power consumption. Analog approaches, by contrast, leverage simpler circuit architectures to achieve comparable precision at lower energy cost, positioning them as attractive alternatives. For instance, mixed-signal designs have been proposed for computing distances between multidimensional vectors while interfacing with CMOS logic for high-speed clustering applications [1]. Other examples include mixed-signal associative memories for efficient nearest-distance searche [2] and programmable current-mode circuits that support both Euclidean and Manhattan metrics in winner-take-all networks [3].

Conventional search methods, such as brute-force algorithms and hash tables, face significant computational bottlenecks when applied to large-scale and high-dimensional data [4]. To address these challenges, in-memory computing (IMC) accelerators have emerged as a promising paradigm, enabling computations directly within memory arrays and thereby mitigating the latency and energy costs associated with data transfer. Content-addressable memory (CAM)-based accelerators, in particular, have been proposed as specialized hardware solutions for accelerating similarity search in machine learning workloads [5–7]. Recent work has further demonstrated the potential of crossbar-based ternary CAMs (TCAMs) to support few-shot in-memory learning for memory-augmented neural networks [8].

Unlike Random-Access Memory (RAM), which retrieves data from a specified address, CAMs perform parallel searches across all memory words, directly comparing them with the input query. This parallelism enables high-throughput data retrieval and has made CAMs indispensable in applications such as network traffic monitoring, access control lists, and associative memory recall. Their ability to support in-memory similarity search also makes them well-suited for clustering and other unsupervised learning tasks. However, conventional binary CAMs, which perform Hamming distance-based lookups, are constrained by high power consumption and large area requirements, as well as inefficiency in handling analog data due to repeated analog-to-digital conversions (ADCs).

Analog Content-Addressable Memories [10] have been introduced to overcome these limitations by operating directly in the analog domain, eliminating the need for costly ADC/DAC operations and reducing overall energy consumption. Furthermore, they provide a platform for analog in-memory computing accelerators with search-based operations. As summarized in Table1, binary CAMs are effective for exact-match lookups, whereas analog CAMs extend this capability to similarity scoring, paving the way for hardware-efficient analog computing.

Integrating analog CAM into hardware-software co-design architectures is a promising approach to improve the efficiency and effectiveness of in-memory implementations of machine learning and data analysis algorithms. Previous works have used diff-CAM with matching intervals for various applications, including tree-based search [12], and logic implementation [13]. We demonstrate how to achieve a single-point exact-match (V-shaped response)

Match type	Circuit type	Distance function	Application
Exact match	Binary-Ternary CAMs [9]	Hamming distance	
Best match	Multi-bit CAMs [10, 11]	Sigmoid-like/ Squared Euclidean	NN search
Threshold match	Multi-bit CAMs [10, 11]	Sigmoid-like/ Squared Euclidean	
Range match	Analog CAMs [10]		Decision tree

Table 1: Different match categories of CAMs.

using our circuit-based Python model for a 6T2M diff-CAM circuit. As a demonstration of analog in-memory similarity search, we showcase trainable associative clustering with diff-CAM.

Methods

Similarity search using differentiable Content Addressable Memory with single-point exact match

To enable rapid evaluation of complex, large-scale problems, it is essential to develop concise and reliable model of analog CAM based on existing taped-out design [10]. Since full SPICE simulations are computationally prohibitive for large systems, we developed a Python-based model that faithfully reproduces the circuit's behavior. This framework leverages Python's flexibility to efficiently explore circuit parameters and systematically investigate different response functions, bridging the gap between circuit-level simulation and algorithm development.

Fig. 1a shows the 6T2M diff-CAM circuit, which produces analog-valued outputs [14, 15]. It has primarily been utilized for interval matching [12], where the stored range is compared with an analog input to determine match or mismatch. The circuit consists of two branches: A pull-down and a pull-up branch, whose responses depend on the conductances of memristors M_1 and M_2. Using the same parameters as the taped-out design[10], we simulate different combinations of G_{M_1} and G_{M_2} in memristors M_1 and M_2, as shown in Fig. 1. b, c, producing outputs consistent with experimental reports [10]. Fig.1 b shows the output from transistor T1, and Fig.1 c shows the output from transistor T2. As shown in Fig.1 d, a combination of these responses will lead to different types of response profiles. A U-shaped response occurs when the two branches are well separated, as shown in Fig.1e. By tuning stored conductance values to achieve a single-point exact match, where the match-line current equals zero, we derive a measure of distance from the center. In this configuration, the stored conductance values for $M1$ and $M2$ are adjusted such that the match interval is a single point, effectively defining distance from the stored center.

We use our Python model to program diff-CAM to sixteen single-point exact matches as shown in Fig.1. panel g. We adjust the conductances of G_1 and G_2 so that the output of the match line is an exact match at a single point. This gives a relation between G and V, such that for a given voltage we get the corresponding conductance values, as shown in Fig.1 h. This is essential for training diff-CAMs, since the conductances in memristors M_1 and M_2 define the stored values/ ranges.

Recent work has explored iterative programming of diff-CAMs through lookup tables [16, 17], and [18]. Our Python-based framework complements these approaches by providing a fast, flexible environment for analyzing and optimizing single-point exact-match behavior in analog CAMs.

Distance function of the diff-CAM circuit with a single-point exact match. A single-point exact match provides the possibility of a full analog similarity search. In this case, the output current I_{ML} is an analog value in the range between the maximum and minimum of the match-line current, that is, $I_{ML_{min}}$ and $I_{ML_{max}}$. The match-line current gives a measure of the distance of the input data from a single stored value at the center, providing analog similarity search.

To obtain the distance characteristic of the single-point exact match, we consider a two-dimensional feature space. Mapping the x and y coordinates of a point in this feature space into the conductance values in memristors of two diff-CAMs with a single-point exact match provides a reference point for measuring distance. Fig.1i shows the resulting distance measure with respect to a reference point, which in this case is chosen to be $V_{\max} = 0.725$ in both x and y coordinates. In other words, the distance of each point is measured with respect to $V_{\max}$. As input, we consider all the possible combinations of x and y in this space. The sum of currents from each diff-CAM determines the distance in that direction: $I_{out}(x)$ gives the distance in the x-direction, and $I_{out}(y)$ provides the distance in the y-direction. The sum of the currents in the match-line,

$$I_{ML} = I_{\text{out}}(x) + I_{\text{out}}(y), \tag{1}$$

results in a Manhattan type distance characteristic. Manhattan distance, which measures the L1 norm, is particularly effective when different dimensions are not directly comparable. Additionally, Manhattan distance is specifically advantageous when input variables have different types.

A diff-CAM operates by storing a value or range and evaluating the similarity of an analog input against this stored reference to produce a matching score. The circuit can be configured to implement various similarity functions by adjusting internal parameters.

As illustrated in Fig. 2(a), the 6T2M diff-CAM circuit consists of two main branches: a pull-down branch and a pull-up branch. The similarity response is governed by the conductances of two memristive elements, M_1 and M_2, which can be programmed to define the match interval. By tuning the conductances of these elements, the matching behavior- and consequently the underlying distance function- can be modified. Previous work has primarily used diff-CAMs for interval matching [12]. The programmed memristor values determine the interval and slope of the response. In particular, a V-shaped response arises when the conductances of M_1 and M_2 are tuned to define a single-point match, where the

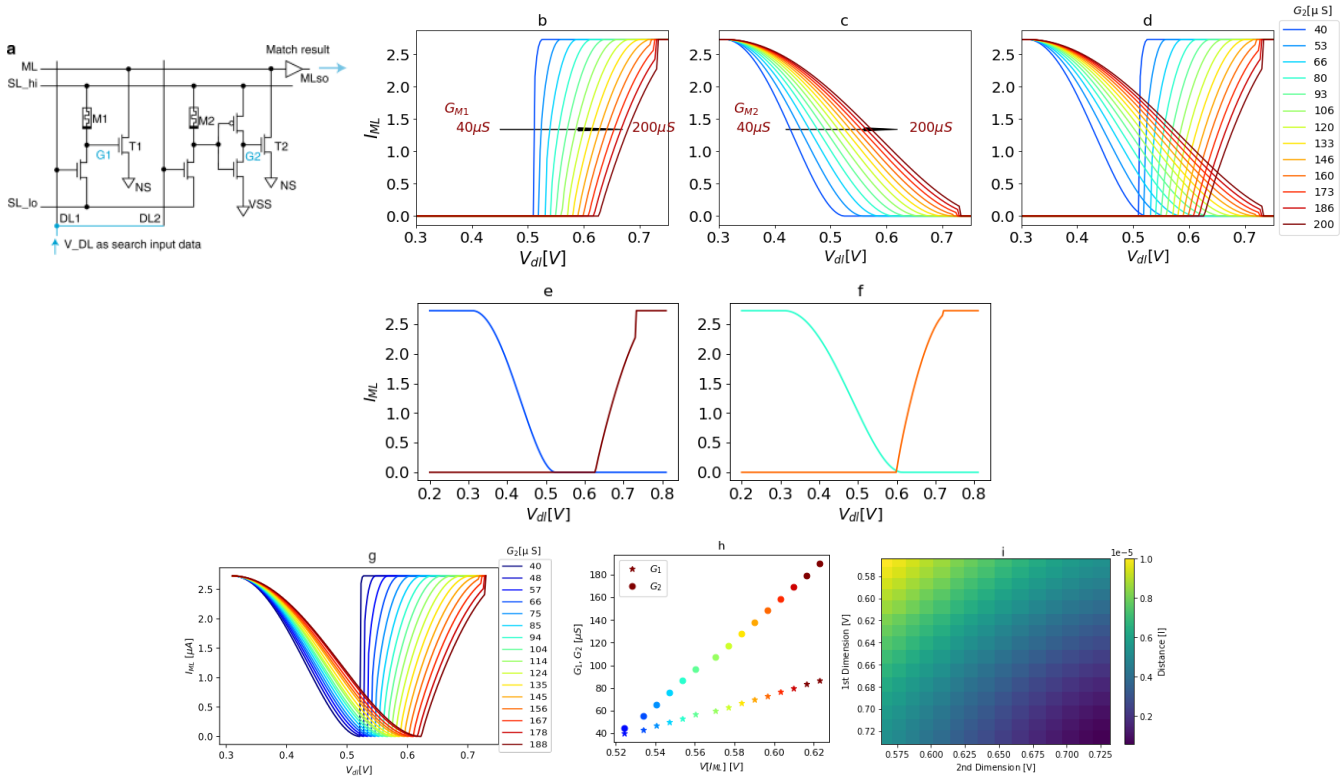

Figure 1: a) analog CAM circuit with six transistors and two memristors (6T2M) [14]. b) The I-V characteristic of the branch with the $M1$, which is the current at $T1$. c) The I-V characterisitc of the branch containing $M2$, which is the current from $T2$ transistor. d) A combination of b and c. e) An example of a U-shape response of the diff-CAM cell. f) An example of the V-shape response of the diff-CAM cell. g) V-shaped response for 16 conductance values. h) G-V relation. i) Distance measure in two-dimensional feature space. The color represents the match-line current, i.e., $I_{ML} = I_{\text{out}}(x) + I_{\text{out}}(y)$. In two dimensions, the measured distance shows a characteristic similar to Manhattan distance.

match-line current (I_{ML}) reaches a minimum, ideally zero, at the stored value.

This single-point exact match behavior forms the basis for analog similarity search, where the output current I_{ML} provides a graded similarity score relative to the input. The output current I_{ML} is an analog value in the range between the minimum and maximum values of the current output, that is, $I_{\text{ML}_{\text{min}}}$ and $I_{\text{ML}_{\text{max}}}$. The current of the match line gives a measure of the analog distance of the input data from a single stored value at the center.

Associative clustering using the diff-CAM circuit model

We propose a hardware-based model for K-means and fuzzy clustering, in which the core operation—similarity search—is performed directly within memory by leveraging the intrinsic similarity search capabilities of analog content-addressable memory (CAM).

1.1 K-means clustering

K-means clustering is an unsupervised learning algorithm that groups unlabeled data into clusters. This centroid-based approach aims to minimize the inertia, i.e., the sum of squared distances to the nearest centroid. The advantage of this method over other approaches is that it is scalable, fast converging, and simple. The objective of K-means is to find

$$\arg\min_{x} \sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2 = \arg\min_{x} \sum_{i=1}^{k} | S_i | \, \text{Var} S_i \tag{2}$$

where μ is the mean of the points in S_i:

$$\mu_i = \frac{1}{S_i} \sum_{x \in S_i} \mathbf{x} \tag{3}$$

S_i is the size of S_i, and $\| \, . \, \|$ is the L^2 norm.

1.2 Fuzzy c-means clustering

Fuzzy clustering is an unsupervised machine learning technique that allows data to belong to multiple clusters with varying degrees of membership, rather than assigning them strictly to one cluster. The fuzzy C-means (FCM) algorithm is one of the most widely used methods, iteratively minimizing an objective function to optimize cluster centers while assigning membership values based on the inverse distance between a data point and each cluster center. Unlike hard clustering, which assigns each data point to a single cluster,

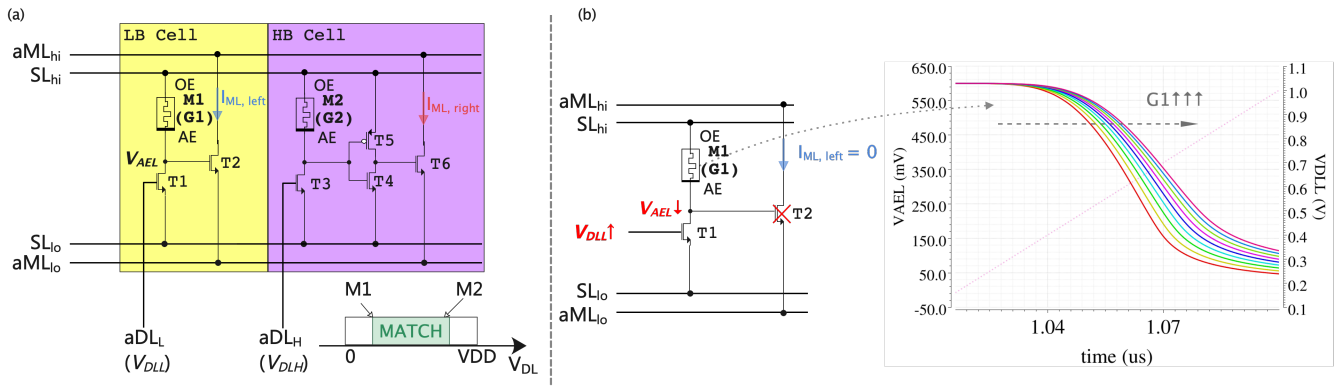

Figure 2: a) This diagram shows the basic structure of a 6T2M diff-CAM cell, including a low-bound (LB) cell and a high-bound (HB) cell. The voltage boundaries are controlled by the conductance values of two memristors (M1 and M2). (b) This diagram uses the LB cell to explain the circuit behaviors of diff-CAM, which is a basic voltage-sharing process. When the gate voltage of T1 is rising to a switching point, V_{AEL} will be pulled down, which closes T2, and thus there's no current flowing through T2. By controlling the conductance of M1, it's possible to control the boundary of the switching point. The HB cell has the same circuit behaviors but with an additional inverter.

fuzzy clustering provides a more flexible and nuanced representation, making it effective for handling uncertainty and overlapping cluster structures. The algorithm proceeds by initializing cluster centroids, updating membership values, and iteratively refining cluster centers until convergence is reached based on a predefined threshold. The key steps in FCM are as follows:

(1) Choose the number of clusters (k).
(2) Randomly select initial cluster centers and membership degrees.
(3) Calculate the membership degree for each data point relative to each cluster.
(4) Update each cluster's centroid by computing a weighted average of data points, where the weights correspond to the membership degrees.
(5) Repeat steps 3 and 4 until the algorithm converges.

The centroid of a cluster c_k for a data point x is given by a weighted average:

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m}. \tag{4}$$

FCM minimizes the following objective function:

$$J(W, C) = \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^m \, \| , x_i - c_j \|^2 \tag{5}$$

were n is the number of data points, c is the number of clusters, $x = i$ data points. c_j are the centroids of clusters, w_{ij} is the membership value of data point of x_i to cluster c_j, and m is the fuzziness parameter (where $m > 1$).

The membership value w_{ij} is calculated as:

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}} \tag{6}$$

Update cluster centroid values using a weighted average of the data points using Eq.6.

The fuzzyness parameter $m \in (1, \infty)$ controls the degree of membership ambiguity. A larger m results in fuzzier clusters. In the limit $m \to 1$, the memberships, w_{ij}, converge to 0 or 1, and the Fuzzy C-means objective coincides with that of K-means. In the absence of experimentation or domain knowledge, m is commonly set to 2.

2 Results

2.1 Uniform distribution of random data

In Fig. 3, we present our approach using diff-CAMs with a V-shaped response function in two dimensions for K-means-type clustering for uniformly distributed random data. The first row shows a schematic of the algorithm. First, the centers of clusters are randomly initialized, and using the G-V relation obtained from the approach presented in Fig.1, the Voltage values in Volts are mapped to the corresponding conductances to be stored as keys, k, in the CAM cells. The distance of each of the data points, as queries q, are then measured from the center of the clusters using the current from diff-CAM for all the features. Note that the stored keys in the center of the V-shape (cluster centers) and the input query data (to be assigned to clusters) are Voltages, and the distance from the center of the V-shaped response is measured as current. The current at the match-line of each CAM cell in the nth iteration, for each feature f, is the sum of currents from branches including T1 and T2 in Fig.1a.

$$I_{\mathrm{ML}}^{fi}(k, q) = I_{T1}^{fi}(k, q) + I_{T2}^{fi}(k, q) \tag{7}$$

Considering a row of CAM cells with F cells, each responsible for the similarity search of a feature (f_i), the sum of currents of a row represents the overall distances from the centers.

$$I_{\mathrm{ML}}^{F}(k, q) = \sum_{i=1}^{F} I_{\mathrm{ML}}^{fi}(k, q) \tag{8}$$

Data points are then assigned to the closest cluster with the smallest distance. In the next step of training, the new cluster centers are

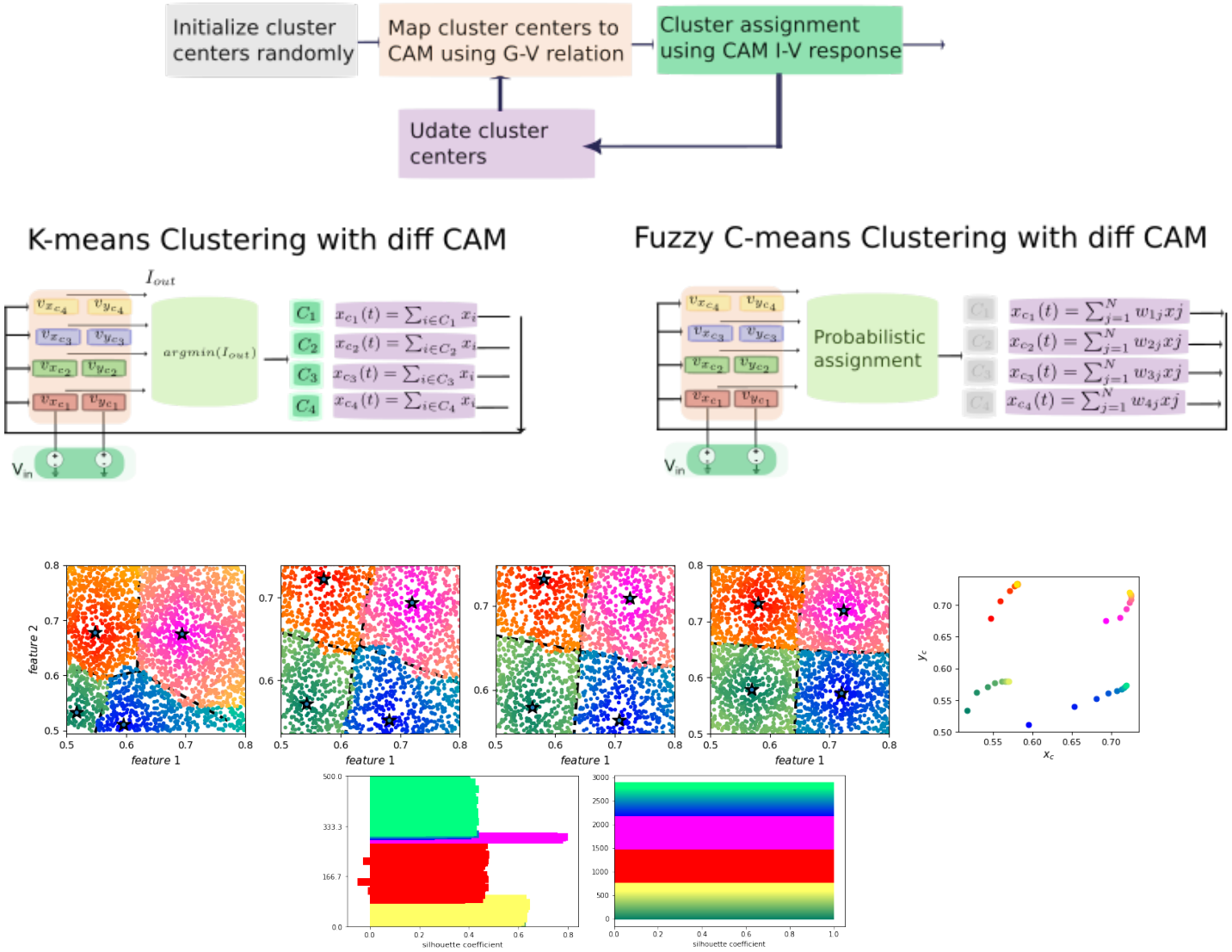

Figure 3: K-means clustering using Lloyd's algorithm with V-shaped CAM. Top: flow chart of the in-memory clustering. Second row: schematic of the K-means and fuzzy C-means algorithms. Third row: Cluster assignment in every second step of 10 steps, and convergence of cluster centers. x_c and y_c represent the coordinates of the cluster centers. Last row: Silhouette score in the first and last step of training, which reaches the maximum value, indicating a perfect clustering.

obtained by taking the average of the coordinates of members of each cluster, according to Eqs 3, 4 in K-means and fuzzy C-means clustering. The new centers are programmed in the CAMs using the G-V relation, and the previous steps are repeated until the centers converge to a fixed point, as shown in the second row. The evolution of cluster centers in the (x_c, y_c) plane shows convergence to a steady value. To evaluate the effectiveness and reliability of clustering algorithms and to assess the quality of the training, we use the silhouette score. This metric ranges from zero to one, with a score closer to one indicating better clustering. The Silhouette score is a measure of how well each data point is assigned to its own cluster compared to other clusters. The Silhouette value of data point i is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{if } \mid C_i \mid > 1, \tag{9}$$

where $a(i)$ is the average distance between data point i and all other points within the same cluster C_i:

$$a(i) = \frac{1}{\mid C_i \mid -1} \sum_{j \in C_i, i \neq j} d(i, j), \tag{10}$$

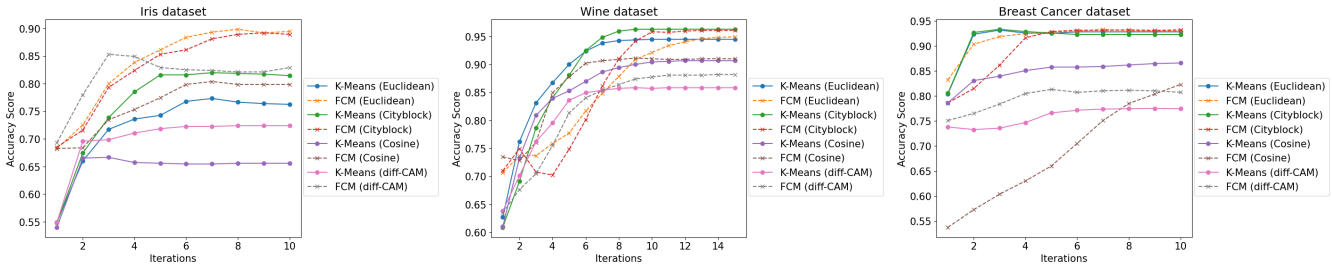

Figure 4: Accuracy of clustering averaged over five realizations for the Iris, Wine, and Breast Cancer datasets.

and $b(i)$ is the smallest average distance between data point i and all points in any other cluster C_J:

$$b(i) = \min_{J \neq I} \frac{1}{C_J} \sum_{j \in C_J} d(i,j). \tag{11}$$

Here, $d(i,j)$ represents the distance between points i and j, and C_I and C_J represent the clusters containing points i and j, respectively. A silhouette score close to one indicates that the data point is well-matched to its own cluster and poorly matched to neighboring clusters. In Fig.3, we observe that after ten iterations, the silhouette score reaches one, as shown in the last row.

2.2 Benchmarking

To test our approach on real-world data, we perform clustering on the Iris, Wine, and Breast Cancer datasets. In Fig.4, we compare the result of clustering using Manhattan, Euclidean, and Cosine distance functions to the distance function measured by the diff-CAM circuit. For clustering with diff-CAM, we use the same approach that we used for the uniform data. We report the results for the K-means clustering algorithm and fuzzy C-means algorithms. The accuracy score is defined as:

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i) \tag{12}$$

where $1(.)$ is the indicator function. As shown in Fig.4, starting from random cluster center assignments, CAM obtains an accuracy comparable to other distance functions in predicting the right clusters. According to Fig.4, while the accuracy obtained from diff-CAM performs better than the Cosine distance function for K-means, it is slightly lower in Wine and Breast Cancer datasets due to the non-idealities of the CAM as a distance measure and asymmetries. On the other hand, as explained in the Appendix, computations performed directly in memory offer time and energy advantages.

In all datasets, the FCM with diff-CAM outperforms the K-means. The analog and statistical nature of the FCM algorithm matches the analog similarity measured by diff-CAM. As noted, the slightly lower performance of diff-CAM compared to the other measures reported in Figs.4 and 5 highlights some remaining non-idealities of the circuit response. To mitigate these effects, we operate the CAM with memristor conductances offering the highest symmetric response. This comes at the cost of a reduced response range, so we find an optimal trade-off point.

In Fig.5, we report the Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) for the Iris, Wine, and Breast Cancer datasets. For all datasets, the FCM obtains a higher ARI and NMI score than K-means. The similarity measure from dfiff-CAM provides the required analog score, which is the main difference between the soft and hard clustering. While in the hard clustering algorithms, such as K-means, the members either belong to a cluster or do not, the membership assignment in fuzzy clustering is a scored membership. By measuring analog distance, diff-CAM provides the required score for the membership to each cluster. During the iteration process, the cluster centers are updated using the average value of the cluster members. Note that during training, the cluster centers are written in the diff-CAM using the G-V results, the same approach that we used for uniformly distributed data. According to Fig.5, the ARI and NMI scores of diff-CAM demonstrate a performance comparable to the other distance measures.

Conclusion

In this study, we proposed the use of diff-CAM with a single-point exact match for centroid-based similarity search, specifically in the task of clustering. Our approach explored the opportunities for CAM-based computing in these tasks. Leveraging a Python-based model of a differentiable Content Addressable Memory (diff-CAM), we could investigate the accuracy of parallel search operations in data processing pipelines in tasks like active learning, smart sampling, and item discovery. The ability to perform centroid-based similarity searches can enhance the performance of clustering, particularly in scenarios requiring iterative refinement, such as active learning cycles or when deploying clustering as a preprocessing step in machine learning pipelines.

By combining the strengths of CAM-based parallel search with our proposed programming approach, this work offers a promising avenue for future research and practical applications in data science and machine learning.

References

[1] Yong Shim, Seong-Wook Choi, Myeong-Gyu Yang, Keun-Yong Chung, and Kwang-Hyun Baek. Energy efficient distance computing: Application to k-means clustering. *Electronics*, 11(3):298, 2022.

[2] Md Anwarul Abedin, Yuki Tanaka, Ali Ahmadi, Tetsushi Koide, and Hans Juergen Mattausch. Mixed digital–analog associative memory enabling fully-parallel nearest euclidean distance search. *Japanese journal of applied physics*, 46(4S):2231, 2007.

[3] Tomasz Talaśka, Marta Kolasa, Rafał Długosz, and Witold Pedrycz. Analog programmable distance calculation circuit for winner takes all neural network realized in the cmos technology. *Ieee transactions on neural networks and learning systems*, 27(3):661–673, 2015.

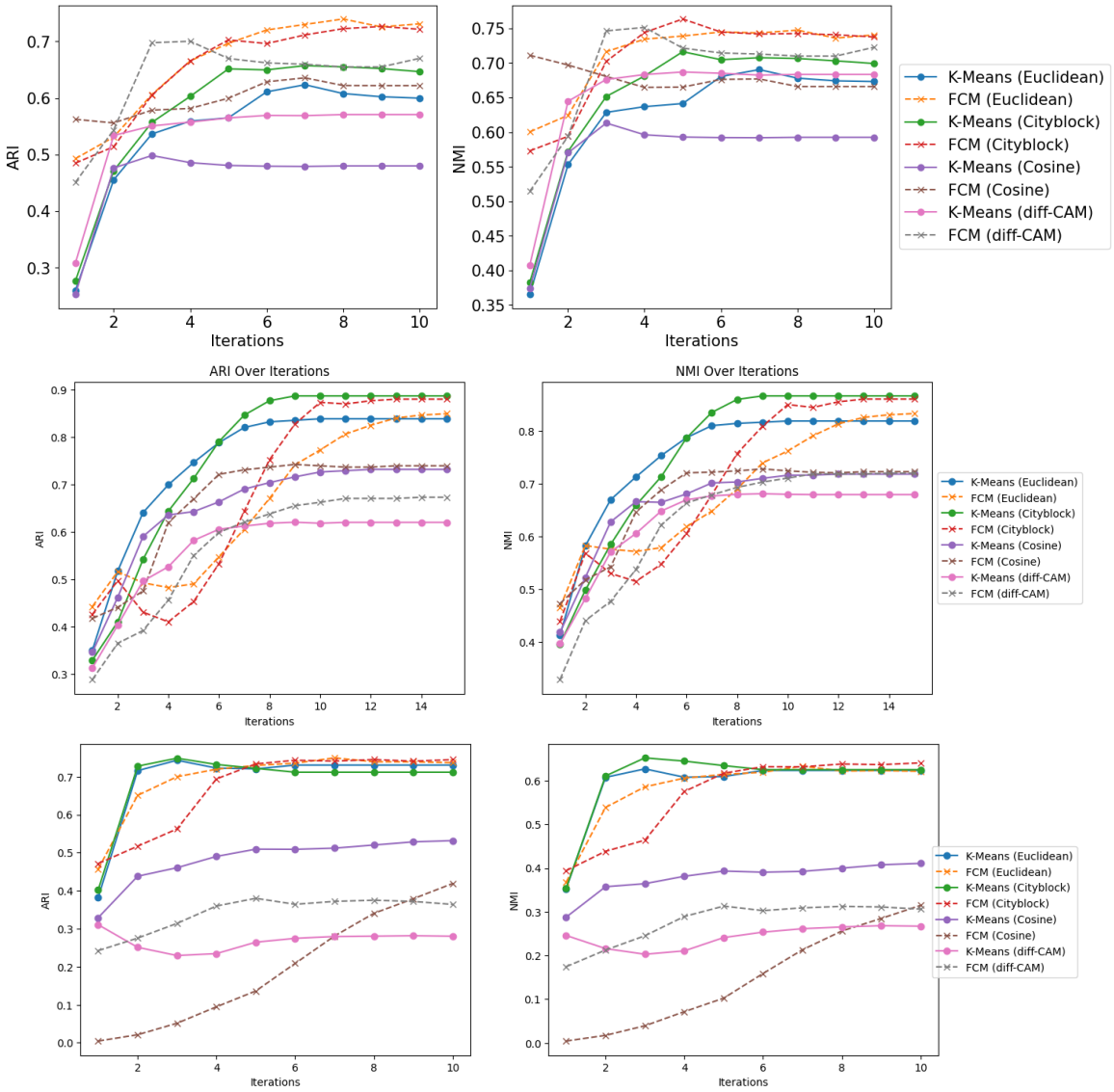

Figure 5: Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) for top row: Iris dataset, middle row: Wine dataset, and bottom row: Breast cancer dataset. The results are the average over five realizations.

Feature	CAM-Based Accelerator	SRAM-Based Accelerator
Energy Efficiency	✓ Excellent (match)	× Lower
Latency	✓ Low (parallel)	× Higher
Precision	× Limited	✓ Flexible
Area	× Larger (per bit)	✓ Compact
Scalability	✓ Good for small k	✓ Better for large k
Robustness	× Sensitive to variation	✓ Very stable

Table 2: Comparison of CAM-based vs SRAM-based accelerators for K-Means clustering

[4] Steven S Skiena. *The algorithm design manual*, volume 2. Springer, 1998.

[5] Giacomo Pedretti, Sergey Serebryakov, John Paul Strachan, and Catherine E Graves. A general tree-based machine learning accelerator with memristive analog cam. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 220–224. IEEE, 2022.

[6] Mohsen Imani, Daniel Peroni, Abbas Rahimi, and Tajana Simunic Rosing. Resistive cam acceleration for tunable approximate computing. *IEEE Transactions on Emerging Topics in Computing*, 7(2):271–280, 2016.

[7] Yue Zhang, Woyu Zhang, Shaocong Wang, Ning Lin, Yifei Yu, Yangu He, Bo Wang, Hao Jiang, Peng Lin, Xiaoxin Xu, et al. Dynamic neural network with memristive cim and cam for 2d and 3d vision. *arXiv preprint arXiv:2407.08990*, 2024.

[8] Ruibin Mao, Bo Wen, Arman Kazemi, Yahui Zhao, Ann Franchesca Laguna, Rui Lin, Ngai Wong, Michael Niemier, X Sharon Hu, Xia Sheng, et al. Experimentally validated memristive memory augmented neural network with efficient hashing and similarity search. *Nature communications*, 13(1):6284, 2022.

[9] DRB Ly, JP Noel, B Giraud, P Royer, E Esmanhotto, N Castellani, T Dalgaty, J-F Nodin, C Fenouillet-Beranger, E Nowak, et al. Novel 1t2r1t rram-based ternary content addressable memory for large scale pattern recognition. In *2019 IEEE International Electron Devices Meeting (IEDM)*, pages 35–5. IEEE, 2019.

[10] Can Li, Catherine E Graves, Xia Sheng, Darrin Miller, Martin Foltin, Giacomo Pedretti, and John Paul Strachan. Analog content-addressable memories with memristors. *Nature communications*, 11(1):1638, 2020.

[11] Arman Kazemi, Mohammad Mehdi Sharifi, Ann Franchesca Laguna, Franz Müller, Xunzhao Yin, Thomas Kämpfe, Michael Niemier, and X Sharon Hu. Fefet multi-bit content-addressable memories for in-memory nearest neighbor search. *IEEE Transactions on Computers*, 71(10):2565–2576, 2021.

[12] Giacomo Pedretti, Catherine E Graves, Sergey Serebryakov, Ruibin Mao, Xia Sheng, Martin Foltin, Can Li, and John Paul Strachan. Tree-based machine learning performed in-memory with memristive analog cam. *Nature communications*, 12(1):5806, 2021.

[13] G Pedretti, F Böhm, M Hizzani, T Bhattacharya, P Bruel, J Moon, S Serebryakov, D Strukov, JP Strachan, J Ignowski, et al. Zeroth and higher-order logic with content addressable memories. In *2023 International Electron Devices Meeting (IEDM)*, pages 1–4. IEEE, 2023.

[14] Giacomo Pedretti, Catherine Graves, Sergey Serebryakov, and John Paul Strachan. Differentiable content addressable memory, February 8 2024. US Patent App. 17/876,471.

[15] Catherine Graves, Can Li, and John Paul Strachan. Analog content addressable memory with analog input and analog output, August 22 2023. US Patent 11,735,281.

[16] Giacomo Pedretti, John Paul Strachan, and Catherine Graves. Iterative programming of analog content addressable memory, October 10 2023. US Patent 11,783,907.

[17] Jiaao Yu, Paul-Philipp Manea, Sara Ameli, Mohammad Hizzani, Amro Eldebiky, and John Paul Strachan. Analog feedback-controlled memristor programming circuit for analog content addressable memory. In *2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE)*, pages 983–988. IEEE, 2023.

[18] Jiaao Yu, Paul-Philipp Manea, Mohammad Hizzani, Sara Ameli, and John Paul Strachan. A memristor variation-aware analog memristor programming circuit for associative memories. In *2024 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 1–4. IEEE, 2024.

[19] Manuel Le Gallo, Riduan Khaddam-Aljameh, Milos Stanisavljevic, Athanasios Vasilopoulos, Benedikt Kersting, Martino Dazzi, Geethan Karunaratne, Matthias Brändli, Abhairaj Singh, Silvia M Mueller, et al. A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nature Electronics*, 6(9):680–693, 2023.

[20] Xiwen Liu, Keshava Katti, Yunfei He, Paul Jacob, Claudia Richter, Uwe Schroeder, Santosh Kurinec, Pratik Chaudhari, and Deep Jariwala. Analog content-addressable memory from complementary fefets. *Device*, 2(2), 2024.

[21] Yihan Pan, Adrian Wheeldon, Mohammed Mughal, Shady Agwa, Themis Prodromakis, and Alexantrou Serb. An energy-efficient capacitive-memristive content addressable memory. *arXiv e-prints*, pages arXiv–2401, 2024.

Appendix

3 Energy and Power Consumption Analysis

3.1 Energy Model for CAM Arrays

Each CAM cell exhibits a parasitic capacitance C_{DL}. Assuming a voltage swing from 0 to V, the energy required to charge a single capacitor is:

$$E_{\text{cell}} = \frac{1}{2} C_{\text{DL}} V^2 \tag{13}$$

For a row with H CAM cells in a column, the energy consumption is:

$$E_{\text{row}} H . \frac{1}{2} C_{\text{DL}} V^2 . \tag{14}$$

Extending this to M rows, each with H vertically stacked CAM cells, the total energy becomes:

$$M.H. \frac{1}{2} C_{\text{DL}} V^2 \tag{15}$$

3.1.1 Dynamic Power with Wire Resistance. For accurate dynamic power estimation, particularly at high switching speeds where wire resistance R_w is non-negligible, both the RC time constant and Elmore delay must be considered. According to Elmore's theorem, the propagation delay is given by:

$$\tau = \sum_{i=0}^{H} (R_{\text{out}} + iR_w) C_{DL} = C_{DL} \left[R_{\text{out}} H + R_w \frac{H(H-1)}{2} \right], \tag{16}$$

where H denotes the height of the analog CAM tile, i.e., the number of CAM cells per column[12]. The throughput is expressed as:

$$\tau = N_0 . f = 1.33 \text{ TOPS}, \tag{17}$$

with $N_0 = 4000$ nodes. The energy function is then defined as:

$$\eta = \frac{\tau}{\text{Power}} = 3.11 \text{ TOPS/W}, \tag{18}$$

corresponding to 461× improvement over the state of the art.

Components of Power Consumption. The overall power consumption can be decomposed into three components:

(1) Static power flowing into the voltage divider

$$P_{\text{static}} V_{sl_{hi}} I_{D0} \tag{19}$$

(2) Dynamic power to charge the data line (DL):

$$P_{DL} = \frac{V_{DD}^2 WN}{R} \tag{20}$$

(3) Dynamic power to charge/discharge the match line (ML):

$$\begin{aligned} P_{ML} = & \frac{1}{2t_{CLK}} (C_{ML} V_{ML0})^2 HN + \\ & \sum_{j=0}^{i=N} \sum_{i=0}^{i=H} \frac{1}{2t_{CLK}} \left(C_{ML} (V_{ML0} - V_{ML,i,j}) \right)^2 \end{aligned} \tag{21}$$

3.2 Area and delay comparison with SRAM

The propagation delay in SRAM accelerators can similarly be estimated via Elmore's theorem:

$$\tau = \sum_{i=0} H(R_{\text{out}} + iR_w) C_{DL} = C_{DL} \left[R_{\text{out}} H + R_w \frac{H(H-1)}{2} \right] \tag{22}$$

3.3 Case Study: K-Means Clustering on the Iris Dataset

3.3.1 CAM-Based Accelerator. For the Iris dataset ($N = 150$ data points, $d = 4$ features, $k = 3$ clusters) and assuming an 8-bit input resolution:

Metric	CAM-bsed	SRAM-based
Energy per iteration	$\sim 9\mu J$	$\sim 25 - 30\mu J$
Area	$\sim 0.0002mm^2$	$\sim 0.1 - 0.2mm^2$
Precision	8-bit	8–16 bit or float
Latency	Very low (1 cycle match)	Higher (MAC loop)

Table 3: Energy and Area Comparison of CAM vs. SRAM Accelerators for K-Means on the Iris Dataset.

Operation	*K-means*	*Fuzzy C-means*
Distance computation	Required	Required
Assignment	Head (one-hot)	Soft (real-valued membership)
Update	Simple mean	Weighted average using membership
Iteration	Fast convergence	Slower convergence
Complexity (per iteration)	O(nkd)	O(nkd)+exponentiation

Table 4: Comparison of operations and per-iteration complexity for K-means and Fuzzy C-means clustering.

- **Energy per match:** Approximately $\sim$ $20pJ$ per cluster center [19–21]. Thus, the energy per data point is:

$$k.E_{\text{match}} = 3 \times 20 = 60pJ. \tag{23}$$

  For 150 points:

$$150 \times 60pJ = 9\mu J. \tag{24}$$

- **Area (CAM size):**
  For three clusters and four features, we need 12 CAM cells, and assuming $20\mu m^2$ per cell, the total area is:

$$A = 12 \times 20\mu m^2 = 0.00024mm^2. \tag{25}$$

3.3.2 SRAM-Based Accelerator Estimate. For the same K-means workload (N=150, d=4, k=3):

- **Energy per distance:** Estimated at 150˜200PJ per data point, based on 12 MAC operations per point and an energy cost of 10˜20pJ.
  This results in:

$$E_{\text{total}} \approx 25 - 30\ \mu J \text{ per iteration.} \tag{26}$$

- **Area:** Dominated by the MAC array, with a negligible memory footprint for cluster centers.
  Estimated area requirement:

$$A \approx 0.1 - 0.2mm^2. \tag{27}$$

The comparative analysis, as shown in the Table.3, demonstrates that CAM-based accelerators provide significant advantages in both energy efficiency and area utilization compared to SRAM-based designs. Specifically, for the K-means clustering task on the Iris dataset, the CAM-based approach achieves over an order-of-magnitude reduction in energy consumption (from tens of microjoules to single-digit microjoules) while occupying a substantially smaller hardware footprint. These results highlight the potential of analog CAM architectures for enabling low-power, high-density machine learning accelerators.

Table 4 highlights the main differences between K-means and Fuzzy C-means clustering. Both algorithms require distance computation at each iteration; however, the assignment step differs significantly: K-means uses a hard one-hot assignment, whereas Fuzzy C-means assigns soft, real-valued memberships to clusters. Consequently, the update step in K-means involves computing simple means, while Fuzzy C-means relies on weighted averages. In terms of convergence, K-means typically reaches stability faster, whereas Fuzzy C-means converges more slowly due to its iterative refinement of memberships. This difference is also reflected in computational complexity: while both methods have a per-iteration cost of $O(nkd)$, Fuzzy C-means additionally requires exponentiation, increasing its computational burden.