Exploiting FeFETs via Cross-Layer Design from In-memory Computing Circuits to Meta-Learning Applications

Dayane Reis, Ann Franchesca Laguna, Michael Niemier, and Xiaobo Sharon Hu
Department of Computer Science and Engineering
University of Notre Dame, Notre Dame, IN, USA, 46556
E-mails: {dreis, alaguna, mniemier, shu}@nd.edu

***Abstract*—A ferroelectric FET (FeFET), made by integrating a ferroelectric material layer in the gate stack of a MOSFET, is a device that can behave as both a transistor and a non-volatile storage element. This unique property of FeFETs enables area efficient and low-power merged logic and memory functionality, desirable for many data analytic and machine learning applications. To best exploit this unique feature of FeFETs, cross-layer design practices spanning from circuits and architectures to algorithms and applications is needed. The paper presents FeFET-based circuits and architectures that offer, either independently or in a configurable fashion, content addressable memory (TCAM) and general-purpose compute-in-memory (GP-CiM) functionalities. These in-memory computing modules bring new opportunities to accelerating data-intensive applications. We discuss the use of these FeFET based in-memory computing fabrics in meta-learning applications, specifically as attentional memory. System-level task mapping and end-to-end evaluation will be discussed.**

***Index Terms*—In-memory computing, ternary content-addressable memory, CiM, TCAM, FeFET, meta-learning.**

I. Introduction

Transistor scaling in CMOS technology has immensely helped to reduce cost for both on-chip and off-chip memories. Still, for the former, low density and high leakage power associated with CMOS memories make it challenging for Static Random-Access Memories (SRAMs) to satisfy the growing demands from data-centric tasks. For the latter, memory density is significantly improved with Dynamic Random-Access Memories (DRAMs). Despite high density, the volatile nature of DRAM requires periodic data refreshing, which incurs significant energy overhead. Non-volatile memories (NVMs) based on emerging technologies offer high density and low power, and require no periodic data refreshes. As such, NVMs could be used as either on-chip and off-chip compute-enabled memories targeting data-centric tasks.

The ferroelectric field-effect transistor (FeFET) is a promising NVM because of its area efficiency and low power compared to other NVMs such as spin-transfer torque magnetic random-access memories (STT-MRAMs), resistive random-access memories (RRAMs) and phase-change memories (PCMs). FeFETs are non-volatile devices with a metal oxide semiconductor field-effect transistor (MOSFET)-like three-terminal structure that do not require high write currents for writing unlike other NVMs. An FeFET's unique characteristics introduce interesting possibilities to the design of low power and dense in-memory computing (IMC) circuits and architectures.

IMC – where bitwise logic and arithmetic operations are performed within the memory boundaries – is an architectural paradigm that could significantly reduce both the energy consumption and the computational overheads associated with data transfers in data-centric tasks. Ternary content-addressable memories (TCAMs) and general-purpose computing-in-memory (GP-CiM) arrays are among the kernels used to realize IMC. TCAMs enable parallel search for an input vector over all the contents stored in an array, and have been used in the design of hardware accelerators for data-centric tasks in the machine learning domain [1]. GP-CiM, in turn, can perform a wide subset of logic and arithmetic operations on words stored in the memory for reducing the energy and latency associated with data transfers in data-centric tasks [2]. Finally, configurable arrays that enable multiple functionalities with the same piece of hardware (e.g., TCAM and GP-CiM) are also used to accelerate inference in meta-learning tasks with IMC architectures [3].

FeFET-based IMC architectures are attractive to emerging machine learning paradigms such as meta-learning [4]–[6]. Meta-learning applications have attentional memories that require nearest neighbor search and memory updates, which incurs significant memory transfer overhand, hence are impeded by the memory wall. IMC can reduce the amount of memory transfer between the compute and the memory units thus improving energy and delay. TCAMs can also aid in performing fast parallel searches. Moreover, in meta-learning, neural networks (NNs) can adapt to learn new classes by using knowledge from previous tasks. The non-volatility of FeFETs makes it ideal in preserving previous attentional knowledge required in meta-learning. To this end, an occasional power failure in the IMC module could lead to the loss of parameters saved in the memory (i.e., weights, activation functions), which would in turn require expensive re-training to be performed.

The paper is structured as follows: Section II reviews the basics of FeFETs and meta-learning models. Section III introduces circuits and architectures based on FeFETs [3] that are used for IMC-friendly execution of meta-learning algorithms. Section IV presents the mapping of IMC-friendly operations of meta-learning to IMC computing kernels. Section V presents an evaluation of meta-learning tasks executed with FeFET IMC fabrics. Finally, Section VI concludes the paper.

II. Background

In this section, we review the basics of FeFETs and metric-based meta-learning models executed on CiM architectures.

978-3-9819263-5-4/DATE21/©2021 EDAA

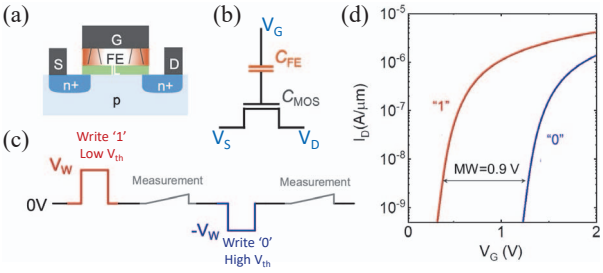


Fig. 1: (a) A physical representation and (b) the equivalent circuit of an FeFET transistor. (c) Write/measurement pulses applied to the gate of the FeFET. (d) The I_D–V_G characteristics of the FeFET.

A. Ferroelectric FETs

FeFETs are devices that structurally resemble MOSFETs. A physical representation of an FeFET transistor is depicted in Fig. 1(a). A thin layer of a ferroelectric (FE) material (commonly, hafnium zirconium oxide (HZO)) is deposited into the FeFET's gate stack. In Fig. 1(b), the equivalent circuit of an FeFET is illustrated. The coupling between the FE capacitance (C_{FE}) and the gate capacitance of the underlying MOSFET (C_{MOS}) creates hysteresis, which confers non-volatility to the device. Distinct logic states – e.g., low V_{th} (high V_{th}) – can be written into the FeFET by applying a positive (negative) write voltage pulse V_W ($-V_W$) to the FeFET's gate (i.e., by asserting $V_G = \pm V_W$). A typical value of $\pm V_W = \pm 4V$ is frequently used in CiM architectures [3], [7]. To read an FeFET, a voltage pulse of smaller amplitude is applied to the device's gate. In Fig. 1(c), measurement pulses are used to assess the length of the device's memory window (MW) and its I_{on}/I_{off} ratios. Fig 1(d) depicts a sufficiently wide MW of 0.9V separating the low and high V_{th} states of the FeFET, with an I_{on}/I_{off} ratio of four orders of magnitude (10^4).

The non-volatility, voltage-based writing, three-terminal structure, and the high I_{on}/I_{off} ratio of FeFET devices are an unique combination of attributes that is extremely desirable for the design of efficient circuits and CiM architectures. For instance, references [1]–[3] demonstrate area, power, and latency advantages of FeFET-based TCAMs, GP-CiMs, and configurable arrays with TCAM/GP-CiM functionality with respect to CMOS and other emerging technologies (e.g., RRAMs), when performing data-centric tasks.

Among data-centric tasks that can exploit the non-volatility, low-power and area efficiency of FeFET-based IMC are meta-learning applications. FeFET IMC are ideal candidates for solving the Von-Neumann bottleneck of attentional memories in meta-learning models. Moreover, the non-volatility of FeFETs can reduce the leakage power of IMC architectures and aid in keeping parameters for lifelong learning.

B. Meta-Learning

A large training dataset is required to achieve high accuracies in deep NNs, while humans only need a few examples to learn. Moreover, the gradient-based methods used in NNs cannot quickly adapt to new tasks because they rely on slow parameter changes, which can cause the network to forget previously learned classes when a new class is introduced. **Meta-learning, or learning how to learn, aims to improve the learning algorithm such that it can quickly adapt to new situations with little information and utilize information from previous knowledge.**

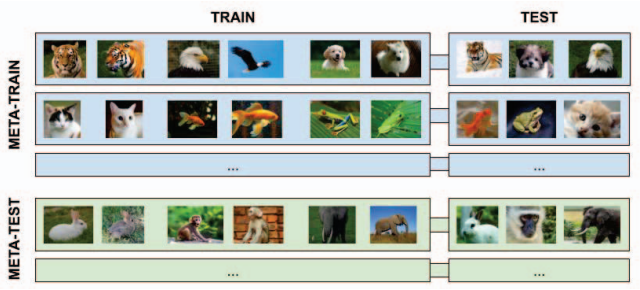


Fig. 2: Meta-learning Training and Testing Setup

The meta-learning setup [4] is composed of the *meta-train* and *meta-test* phase. Each example in the meta-training set is composed of training and a test set. Each meta-train and meta-test example can have a different set of classes, as shown in Fig. 2. Unlike a traditional NN test setup, the meta-test contains classes that are not in the meta-train. During deployment, the train set of a meta-test sample can be considered as a mini-training, or training to an unseen set of classes with little information. Meta-learning is often used to accomplish few-shot learning or learning using only a few examples. Fig. 2 is an example of an N-way K-shot learning (3-way 2-shot learning) where N represents the number of classes and K is the number of examples per class. The $N \times K$ test images are also called the support set.

Metric-based meta-learning is similar to nearest neighbor search, where a query (test example) is compared to the embeddings of the support set. An example of this is prototypical networks (PNs) [5] in which an embedding function (a neural network) encodes the support set and the query to an N-dimensional feature space. A prototype vector for each class is then defined by obtaining the embedded support set vectors' mean. The query's class is then obtained by performing a nearest neighbor search between the embedded query vector and the prototype vectors.

Another example of meta-learning is the memory augmented neural network (MANN) [6], which uses a memory or storage buffer to store previously learned parameters. In MANN, the neural network generates a feature vector $\mathbf{x}$ to retrieve the relevant entries in the memory. This newly generated feature vector $\mathbf{x}$ either updates the most relevant entries or replaces the least recently used memory entry. Different similarity metrics can be used to retrieve memory entries.

Compared to PNs, MANNs have memories which help in catastrophic forgetting. However, MANNs have less generalization capabilities for an out-of-distribution task and are harder to train. PNs and MANNs are very similar during meta-test but are meta-trained differently because of different loss functions.

III. FeFET IMC Circuits and Architectures

In this section, we describe 3 FeFET-based CiM kernels (i.e., circuits and architectures) used in the design of accelerators for attentional memories in meta-learning tasks.

A. Ternary Content-Addressable Memories

TCAMs are associative memories that have the ability to perform *parallel* searches for an input vector (i.e., a query)

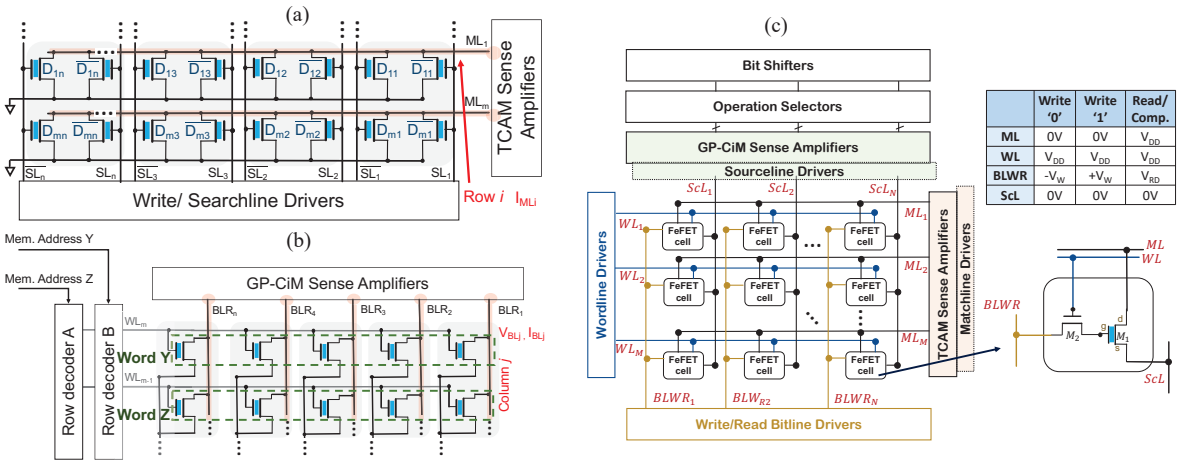


Fig. 3: (a) A 2FeFET TCAM array, (b) a 1FeFET compute-enabled RAM, and (c) a configurable FeFET array with dual functionality.

over all the contents stored in an array. Traditionally used in routers and caches, TCAMs have been gaining popularity in the design of hardware accelerators for meta-learning models, as they allow for efficient implementation of similarity metrics used for nearest neighbor search [1], [3]. TCAM designs based on either CMOS or emerging technologies have been proposed. For instance, a CMOS TCAM cell consists of 16 transistors and occupies an estimated area of 1.12 μm^2, which is 4× (8×) larger than a RRAM (FeFET)-based TCAM cell [1]. The large area of the CMOS TCAM and its high static power can be prohibitive. RRAM and FeFET-based TCAMs have notably lower power consumption and higher densities.

Fig. 3(a) depicts an FeFET TCAM array where 2 rows and 4 columns are represented. Each row i of the TCAM array stores a pattern (and its complement) D_{i1}, D_{i2}, D_{i3}, ..., D_{in} ($\overline{D_{i1}}$, $\overline{D_{i2}}$, $\overline{D_{i3}}$, ..., $\overline{D_{in}}$). The match lines ML_i of the array need to be pre-charged at the beginning of each search operation. Following pre-charge, an input pattern (and its complement) are applied at the search lines SL_{i1}, SL_{i2}, SL_{i3}, ..., SL_{in} ($\overline{SL_{i1}}$, $\overline{SL_{i2}}$, $\overline{SL_{i3}}$, ..., $\overline{SL_{in}}$) to initiate a parallel search over the m rows of the TCAM array. At this point, if all the bits of the query $SL_{1...n}$ are equal to the bits $D_{i1...in}$ stored in the row i of the TCAM, the ML_i remains fully charged, which is sensed as a 'match' by the TCAM sense amplifiers.

In case there is a mismatch between one (or more) bits of a query and the contents stored in row i of a FeFET TCAM array, the ML_i discharges, which is sensed as a 'mismatch' by the TCAM sense amplifiers. The speed of discharge is proportional to the number of mismatched bits and determines a 'degree of mismatch'. Specialized sense amplifier circuits [1] can be used to detect similarity between a query and the data entries with precision of up to 8 bits. Similarity search with the FeFET TCAM can be used for performing nearest neighbor searches in meta-learning classification tasks based on locality sensitivity hashing (LSH) encoding [1].

B. Compute-Enabled Random Access Memories

Compute-enabled RAMs enable a subset of operations (e.g., Boolean logic, basic arithmetic) to be carried out by customized peripheral circuits in the RAM. Compute-enabled RAMs belong to the family of IMC circuits and architectures that have been considered as candidates for easing the burden of data transfers in data-centric tasks. As the subset of operations supported by compute-enabled RAMs can be generalized to a vast application space, compute-enabled RAMs are also known as general-purpose CiM (GP-CiM) architectures.

FeFETs enable the design of ultra-dense, low-leakage, and fast RAMs and GP-CiM architectures. 32 Mbit AND-type FeFET-RAMs comprised of single FeFET memory cells (1-FeFET) have been fabricated [8]. Furthermore, a GP-CiM based on 2T+1FeFET memory cells have been proposed in [2]. The CiM design of [2] is based on a customized sense amplifier (SA) – a circuit that can also be used with denser (1-FeFET) memory cells.

Fig. 3(b) depicts a 1-FeFET GP-CiM array where 2 rows and 5 columns are represented. Two n-bit words Y and Z are stored in the memory in a column-aligned fashion [2]. The index j of a column refers to the individual bits of Y and Z (i.e., Y_j, Z_j). Computation at the bitline j (BLj) with 1-FeFET GP-CiM requires simultaneous activation of the two wordlines (WL) for words Y and Z, which is accomplished with row decoders A and B. The voltage on or current flow in BLj (i.e., V_{BLj} or I_{BLj}) can be sensed with GP-CiM SAs. Through voltage and current sensing combined with additional logic circuits, the outputs of the GP-CiM SA generate Boolean logic results, e.g., AND, OR and XOR, between Y_i and Z_i. Addition between the words Y and Z is also possible with the use of an in-memory adder circuit embedded in the SA [2]. In-memory bit shifts are also possible with the logarithmic shifter introduced in [9].

C. Configurable Arrays with Dual Functionality

Configurable arrays with multiple functionalities can be reconfigured to work as RAMs, GP-CiMs, TCAMs, crossbars, etc. The modality of a configurable array can be dynamically changed. Configurable arrays based on CMOS and FeFET have been proposed in [10] and [3], [11], respectively. As noted in [3], there is a direct relationship between the number of supported functionalities and the amount of peripheral circuits in a configurable array. Thus, if a complicated and large peripheral circuitry is needed in order for the array to support many (e.g., more than 2) modes, leakage power and area overheads may outweigh the benefits of dynamic modality switching in configurable arrays.

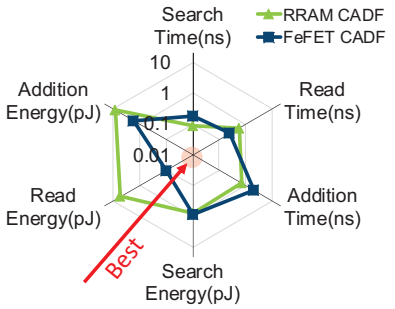


Fig. 4: Comparison between RRAM and FeFET CADF designs

We consider a specific configurable array that can perform a search in a TCAM mode, and then use a GP-CiM mode to perform additional computation on the result (e.g., bitwise logic, additions, and subtractions between memory words). We refer to this configurable array as CADFs. CADFs can be good candidates for accelerating meta-learning models such as MANNs, which often require similarity searches with TCAMs and the computation of distance metrics such as the L_1 with a GP-CiM array.

Fig. 3(c) depicts a FeFET CADF based on 1T+1FeFET memory cells. The access transistor (M_2) of a 1T+1FeFET memory cell is turned on during GP-CiM operations and searches (by asserting WL=V_{DD}). A 1T+1FeFET memory cell can be written by applying a voltage $\pm$ V_W to $BLWR$, while ML, ScL are kept at $0V$. The search operation with a CADF proceeds similarly to the search on regular FeFET TCAM arrays (i.e., [7]). The peripheral circuits used solely in the GP-CiM mode (i.e., GP-CiM SAs) can be power gated during search to lower power consumption. To perform GP-CiM operations, a read voltage (V_{RD}) is applied to $BLWR$, while (V_{DD}) is applied to the ML. ScL is kept at $0V$. The current flow between ML and ScL is sensed by the current-based GP-CiM SAs in the CADF design.

D. Evaluation of CADF

Here, we compare the time and energy figures-of-merit (FoM) of searches, reads and in-memory additions performed with RRAM and FeFET-based CADF designs in [11] and [3], respectively. The designs evaluated are based on a 45nm predictive technology model (PTM) [12], and simulated with HSPICE. IMC arrays of 1024×64 are employed. We pick these two designs for our evaluation as they are alternatives to CMOS architectures due to their lower leakage and area [3]. Furthermore, CADF designs enable all the basic operations needed by meta-learning tasks.

In the radar-chart of Fig. 4, the central point indicated by the arrow represents the best value for a FoM. We note that while RRAM and FeFET designs are comparable in terms of search/ read/ addition times, as well as search energy, the FeFET design is more energy efficient than the RRAM counterpart. The low energy of FeFETs is a direct consequence of the device's characteristics (e.g., the high I_{on}/I_{off} ratio and low area). Comparison between other IMC architectures (e.g., separate GP-CiM and TCAM designs) can be found in references [1], [7], [13], [13].

IV. IMC-Friendly Operations

Here, we describe the mapping of operations needed by meta-learning tasks to the IMC hardware.

A. Fixed Point Operations

A fixed-point, in-memory (iM)-Addition is straightforward to implement with an iM adder. Subtraction can be implemented by performing iM-additions with 2's complement method. On the other hand, fixed-point divisions can be expensive to perform in hardware and be constrained to 2^n, where $n \in \mathbb{Z}$, as the iM-division operations can be simplified as bit shifts.

B. Floating Point Operations

Floating point (FLP) operations achieve higher accuracy than fixed-point operations in various applications. To implement an addition or a subtraction operation of FLP numbers, the mantissa must be shifted such that the exponents are the same. To minimize rounding errors, given two numbers, the mantissa of $\mathbf{a}$, the number with the larger exponent, is shifted to the left. The number of bit shifts is obtained by performing an iM-subtraction of the exponents and a 2's complement operation (if necessary). The exponent of $\mathbf{a}$ is then replaced by the smaller value. After the exponents are aligned, an iM-addition or iM-subtraction operation on the mantissa is performed. The sum (or difference) is then normalized to represent the proper FLP format using in-memory bit shifts on the mantissa and iM-addition/iM-subtraction on the exponent based on the number of bit shifts. The FLP 2^b division can be implemented using an iM-subtraction and bit-shifts, with the minuend as the exponent of the sum of the samples and the subtrahend equal to b.

C. In-memory Nearest Neighbor Search

Nearest neighbor search (NNS) can be implemented using either the TCAM or the GP-CiM function. TCAMs directly perform NNS based on Hamming distance. By using range encoding [14], TCAMs can implement L_∞ distance based NNS. TCAMs can also implement locality sensitive hashing (LSH) search by performing a TCAM lookup with the LSH signatures. The signatures are obtained using a matrix multiplication. This operation can be integrated in the last layer of the neural network embedding function of meta-learning architectures.

GP-CiM can be used to compute a Manhattan (L_1) distance metric. L_1 distance requires three operations: (i) subtraction, (ii) absolute value, and (iii) summation. These are implemented using a FLP iM-subtraction, a NOT if the value is negative, and FLP iM-additions/subtractions. The minimum distance can be obtained by subtracting two elements at a time, choosing the minuend if the result is negative and choosing the subtrahend if the result is positive. The chosen element is then compared again to other chosen elements. This is repeated until a minimum is found. The CADF functionality can also be exploited as the minimum can be obtained in a TCAM quickly by searching for a query of all 0's.

A TCAM-based NNS using L_∞ distance metric is faster but less accurate than GP-CiM based NNS using L_1 distance metric. To obtain a good balance with accuracy and speed, CADF can be used to perform an initial L_∞ k-NNS using the TCAM functionality followed by L_1 k-NNS using the GP-CiM functionality.

V. IMC for Meta-Learning Tasks

Here, we present the mapping and evaluation for PNs (Sec. V-A) and MANNs (Sec. V-A). The GPU evaluations are obtained using the Nvidia Profiler (NVProf), Nvidia Visual Profiler (NVVP) and Nvidia System Management Interface (nvidia-smi).

A. Prototypical Networks

In prototypical learning, a neural network encodes the samples into a feature embedding space. To perform a meta-test, a nearest neighbor search is implemented in this feature embedding space between the prototypes of each class and a given query. The support feature embeddings for each class (way) are averaged to obtain the prototype vector. The nearest prototype to query x is chosen based on a distance metric (i.e. L_1, L_2, L_∞ or cosine distance) to determine x's class.

PNs require the following operations during meta-test: (1) prototype computation and (2) nearest prototype search. We use in-memory FLP representation to implement PNs. Computing for the prototype requires averaging the embedded support vectors. This averaging can be implemented using additions and a division operation; To avoid expensive division operations, we constrain the number of training examples in a meta-test sample to 2^b (or 2^b-shots, where $b \in \mathbb{Z}$, for few-shot learning). The original PNs [5] use L_2 squared distance, however L_2 distance is expensive because of the number of required multiplications. L_2 distance can be replaced by the more hardware friendly (faster and more energy efficient) L_1 distance with a small loss in accuracy. The GP-CiM nearest neighbor search is hence implemented based on the FLP L_1 distance; the FLP representation preserves the accuracy while the L_1 distance guarantees that it is still hardware friendly (compared to L_2 or Cosine distance).

The GPU and GP-CiM based PNs are summarized in Table I. The L_1 GP-CiM implementation obtained iso-accuracy with the L_2-squared GPU (using squared L_2 distance) using the Omniglot dataset [15], and a small (less than 1%) reduction in accuracy with the mini-ImageNet dataset. Since the GP-CiM-PN is constrained to 2^b shots, GP-CiM-PN can only perform 1-shot, 2-shots, and 4-shots. The 5-shot can be implemented as an 8-shot with some repeated support samples. For the 5-way 5-shot approach this achieves comparable accuracies.

For a single episode, the GP-CiM-PN can achieve 2808x speedup and 2372x energy improvement when compared with the GPU approach for the prototype calculation as shown in Fig. 5. A single episode does not fully utilize the GPU parallellism capabilities. Implementing in batch, the memory overhead can be reduced by latency hiding [16]. On a batch scenario of 1000 episodes, a 1650× speedup and 89× energy reduction is achieved by using parallel implementation of 1000 episodes of the prototype calculation. Consequently, GP-CiM achieves 111× speedup and 5170× energy improvement for a single episode, and a 154× speedup and 457× energy improvement for a parallel implementation of 1000 episodes. In the batch case, the GPU and GP-CiM achieves their maximum computational capabilities. The L_1 distance based implementation shows slightly smaller improvements when compared to using an L_2-squared distance based implementation.

TABLE I: GPU and GP-CiM accuracy for 2^b PNs using Omniglot and miniImageNet Dataset

Dataset	Hardware (Distance)	5-way 1-shot	5-way 2-shot	5-way 4-shot	5-way 5-shot
Omniglot	GPU (L_2^2)	99.33	99.71	99.56	99.70
	GP-CiM (L_1)	98.35	99.17	99.56	99.65
miniImageNet	GPU (L_2^2)	53.92	61.95	68.74	69.48
	GP-CiM (L_1)	52.05	61.04	67.83	68.85

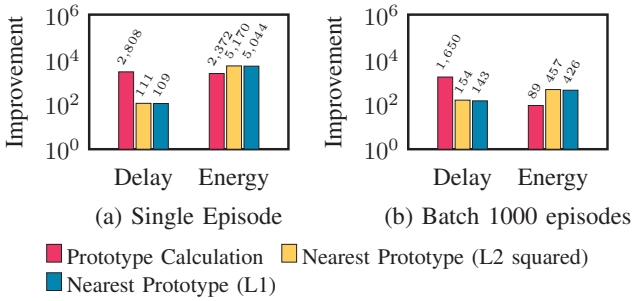


Fig. 5: Delay and Energy improvements of using CiM-PN compared to the GPU implementation using L_2-Squared and L_1 distance

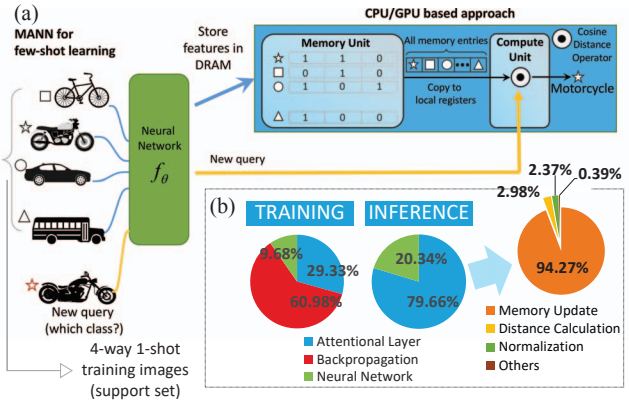
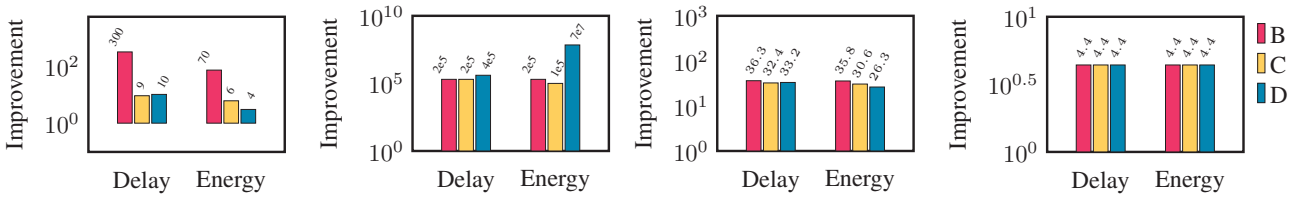


Fig. 6: **(a)** An example of few-shot learning classification that uses MANNs. **(b)** The GPGPU run time distribution of MANNs during training and inference, for the Omniglot dataset [17].

B. Memory-Augmented Neural Networks

MANNs improve a neural network architecture's ability to learn how to learn and prevent catastrophic forgetting by adding a dedicated associative memory, referred to as the memory module. As shown in Fig. 6(a), MANN also includes a neural network f_θ which embeds the input to a feature vector for either updating a recently accessed memory entry, replacing an older memory entry or performing classification.

MANN's attentional memory operations can be separated into two main operation groups: nearest neighbor search and memory update. The nearest neighbor search can be implemented in TCAM either by using an L_∞ distance metric or an LSH approach or GP-CiM using L_1 distance. When using the LSH approach, the LSH encoding can be implemented by multiplying the LSH function with the last fully connected layer's weights. The memory module should be able to do a nearest neighbor search, update the memories, and decide which entries to replace or remove. The memory updates average each train (or support) vector and the closest memory entry to generate new class representations.



(a) Distance Metric (b) Memory Update (c) Attention Layer (d) End-to-End MANN

Fig. 7: Delay and Energy improvements of evaluation setups B through E with respect to Setup A with the (a) distance metric, (b) memory updates, (c) the attentional layer operations as well as the (d) end-to-end evaluation for MANN

We compare five implementations of MANNs based on the different hardware and distance metrics as shown in Table II. Setup A implements a MANN in the GPU using cosine similarity. Setup B uses TCAMs to perform an LSH lookup using a Hamming distance-based distance metric. Setup C uses separate TCAM and GP-CiM arrays to implement a combined $L_\infty + L_1$ distance metric. Setup D uses CADF to perform an L_∞ distance metric using a TCAM lookup followed by an L_1 distance metric using the GP-CiM mode. The physical memory size for all the setups are the same; the memory module size for Setup C is hence half of the attentional memory size of Setup D. For Setups B and C, the averaging required in the memory update is implemented in the GPU while Setup D uses GP-CiM to perform the averaging.

Table II show the accuracies for the different setups. Setup D achieves higher accuracy (95.14%) because of the use of floating point operations as compared to using Setup C (91.98%). Setup B uses LSH, which implements an approximate nearest neighbor search and is less accurate (95.14% v.s. 91.98%) than an exact search of Setup D using L_∞+L_1 distance metric at iso-memory size. The cosine distance used in the GPU performs better in terms of accuracy than CADF by 4%.

Fig. 7 show the latency and energy improvements for setups B, C and D compared to setup A. Setup B have the greatest latency improvement among the different setups for distance metric calculations because it only requires one approximate TCAM search using the LSH approach and does not require any GP-CiM-based computation. Setups C and D compute their distance metrics based on multiple TCAM searches, followed by GP-CiM additions and subtractions. Consequently, Setup B also achieves the best energy improvement among the different setups. Setup C and D have higher energy consumption because of the GP-CiM operations and more TCAM searches when compared to Setup B. Using a CADF (Setup D) can achieve even greater FoM improvements compared to using the LSH approach (Setup B) and a separate GP-CiM and CAM (Setup C) because of the less memory required. This in turn improves the latency by 2× and the energy by up to 70× when compared with Setup C. The five orders of magnitude improvement in the memory update is because of the reduction of memory transfers between the GPU and the memory.

The latency and energy improvement for the attentional memory (Fig. 7(c)) and the end-to-end MANN (Fig. 7(d)) are calculated using the Ahmdal's law and the breakdown of operations shown in Fig. 6(b). Only the distance calculation and memory updates are accelerated by the hardware. Hence we can achieve up to 36.4× improvement for the attentional layer and 4.9× improvement for the end-to-end implementation.

TABLE II: Evaluation Setups for MANNs

Setup	Hardware Type	Distance Metric	Ref.	Accuracy
A	GPGPU+DRAM	Cosine	[1]	99.70%
B	TCAM	LSH	[1]	92.21%
C	TCAM+GP-CiM	L_∞+L_1	[17]	91.98%
D	FeFET CADF	L_∞+L_1	[3]	95.14%

VI. Conclusion

This paper presents FeFET based circuits/architectures that bring new opportunities to accelerating data-intensive applications. We specifically consider the use of FeFET-based IMC fabrics in meta-learning applications, specifically as attentional memory. System-level task mapping and end-to-end evaluation are discussed.

Acknowledgments

This work was supported in part by ASCENT, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] K. Ni, et al. Ferroelectric ternary content-addressable memory for one-shot learning. *Nature Electronics*, 2(11):521–529, 2019.
[2] D. Reis, et al. Computing in Memory with FeFETs. In *ISLPED*, pages 24:1–24:6, New York, NY, USA, 2018. ACM.
[3] D. Reis, et al. Attention-in-Memory for Few-Shot Learning with Configurable Ferroelectric FET Arrays. In *ASP-DAC*, 2021. (In press).
[4] S. Ravi et al. Optimization as a Model for Few-Shot Learning. In *ICLR*, 2017.
[5] J. Snell, et al. Prototypical Networks for Few-shot Learning. In *NIPS*, pages 4080–4090, 2017.
[6] A. Santoro, et al. Meta-Learning with Memory-Augmented Neural Networks. In *ICML*, pages 1842–1850, 2016.
[7] X. Yin, et al. An Ultra-Dense 2FeFET TCAM Design Based on a Multi-Domain FeFET Model. *IEEE TCAS-II*, 66(9):1577–1581, 2019.
[8] S. Dunkel, et al. A fefet based super-low-power ultra-fast embedded nvm technology for 22nm fdsoi and beyond. In *IEDM*, Dec 2017.
[9] D. Reis, et al. A Fast and Energy Efficient Computing-in-Memory Architecture for Few-Shot Learning Applications. In *DATE*, pages 127–132, 2020.
[10] S. Jeloka, et al. A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory. *JSSC*, 51(4):1009–1021, 2016.
[11] X. Zhang, et al. FeMAT: Exploring In-Memory Processing in Multifunctional FeFET-Based Memory Array. In *ICCD*, pages 541–549, 2019.
[12] Y. Cao. *Predictive technology model for robust nanoelectronic design*. Springer Science & Business Media, 2011.
[13] D. Reis, et al. Modeling and benchmarking computing-in-memory for design space exploration. In *GLSVLSI*, pages 39–44, 2020.
[14] A. Bremler-Barr, et al. Encoding Short Ranges in TCAM Without Expansion: Efficient Algorithm and Applications. *IEEE/ACM Transactions on Networking*, 26(2):835–850, April 2018.
[15] B. M. Lake, et al. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
[16] H. Sato, et al. I/O chunking and latency hiding approach for out-of-core sorting acceleration using GPU and flash NVM. In *BigData, Washington DC, USA, December 5-8, 2016*, pages 398–403, 2016.
[17] A. F. Laguna, et al. Ferroelectric FET Based In-Memory Computing for Few-Shot Learning. In *GLSVLSI*, page 373–378, New York, NY, USA, 2019. ACM.