Emerging Neural Workloads and Their Impact on Hardware

David Brooks*, Martin M. Frank†, Tayfun Gokmen†, Udit Gupta*, X. Sharon Hu‡, Shubham Jain§,
Ann Franchesca Laguna‡, Michael Niemier‡, Ian O'Connor¶, Anand Raghunathan§,
Ashish Ranjan†, Dayane Reis‡, Jacob R. Stevens§, Carole-Jean Wu‖, Xunzhao Yin**
E-mails: dbrooks@eecs.harvard.edu, mmfrank@us.ibm.com, tgokmen@us.ibm.com, ugupta@g.harvard.edu,
shu@nd.edu, jain130@purdue.edu, alaguna@nd.edu, mniemier@nd.edu, Ian.Oconnor@ec-lyon.fr, raghunathan@purdue.edu,
ashish.ranjan@ibm.com, dreis@nd.edu, steven69@purdue.edu, carolejeanwu@fb.com, xzyin1@zju.edu.cn
* Harvard University † IBM T.J. Watson Research Center ‡ University of Notre Dame
§ Purdue University ¶ École Centrale de Lyon ‖ Facebook ** Zhejiang University

***Abstract*—We consider existing and emerging neural workloads, and what hardware accelerators might be best suited for said workloads. We begin with a discussion of analog crossbar arrays, which are known to be well-suited for matrix-vector multiplication operations that are commonplace in existing neural network models such as convolutional neural networks (CNNs). We highlight candidate crosspoint devices, what device and materials challenges must be overcome for a given device to be employed in a crossbar array for a computationally interesting neural workload, and how circuit and algorithmic optimizations may be employed to mitigate undesirable characteristics from devices/materials. We then discuss two emerging neural workloads. We first consider machine learning models for one- and few-shot learning tasks (i.e., where a network can be trained with just one or a few, representative examples of a given class). Notably crossbar-based architectures can be used to accelerate said models. Hardware solutions based on content addressable memory arrays will also be discussed. We then consider machine learning models for recommendation systems. Recommendation models, an emerging class of machine learning models, employ distinct neural network architectures that operate of continuous and categorical input features which make hardware acceleration challenging. We will discuss the open research challenges and opportunities within this space.**

I. Introduction

The use of neural network (NN) and machine learning (ML) models related to language processing tasks (e.g., recurrent neural networks (RNNs)), image recognition (e.g., convolutional neural networks (CNNs)), etc. has seen widespread success over the last decade. Given algorithmic successes, within the last five years, there has been a push to support said models in hardware in order to (i) perform computations associated with a given ML model in a more energy efficient manner and/or (ii) to use more sophisticated models in resource constrained environments. Still, workloads/models exist and/or are emerging that are *not* well-suited for existing hardware platforms and/or existing accelerators. Moreover, emerging workloads could have substantial benefits for an end-user if said models can be implemented efficiently. In this paper, we review/discuss examples of said workloads, as well as ongoing/nascent efforts to support said models in hardware.

We begin in Sec. II by reviewing efforts that target analog crossbar architectures that (i) can be employed for NN inference, (ii) could be employed for NN training, and (iii) can meet the algorithmic requirements for inference and training *given the constraints of candidate crosspoint devices*. As matrix multiplication can dominate execution time in many existing ML models, the application-level impact of crossbars should be quite obvious. As we will discuss herein, crossbars may also be used in support of emerging ML models as well.

An algorithmic challenge for traditional NNs is that they cannot quickly adapt to new tasks without extensive retraining with a large amount of information [1]. However, humans can leverage acquired knowledge to quickly adapt to new situations. The task of learning how to learn – or *meta-learning* [2] – leverages past experiences to learn new tasks.

A promising NN-based approach for implementing meta-learning is memory augmented neural networks (MANNs), where features extracted from a NN can be stored and retrieved from an attentional memory (typically DRAM [3]–[6]). Recent demonstrations of MANNs include differentiable neural computers (DNC) [3], [4] which (i) can learn to construct complex data structures such as graphs and decision trees (e.g., navigating the London underground [4]), (ii) answer questions related to data structures, and (iii) perform one/few shot learning tasks [5], [6]. Relevant features for a classification task are extracted from a few training examples, stored in the network's memory, and retrieved to make accurate predictions.

A key function of the attentional memory is *content-based addressing*, where the distance between a search vector and all stored vectors is calculated to find the closest match. In a conventional approach, the stored memory vectors (in DRAM) need to be transferred to a compute unit (CPU or GPU) to compare distances with a given query. As such, energy dissipation and latency limitations can represent significant challenges to scaling up MANNs. Alternative memory architectures that support massively parallel searches are highly desirable.

In this regard, both crossbar architectures and content addressable memory (CAM) architectures have recently been proposed as a way to accelerate MANNs. As such, work in [7] (referred to as X-MANN) proposes to use crossbar arrays to perform distance calculation searches for MANNs [2] and neural Turing machines (NTM) [8] (networks that are also used for few-shot learning). (See Sec. III.)

Alternatively, CAMs, a special type of memory that performs parallel search, can be exploited to carry out parallel associative searches. This approach is suitable for a locality sensitive hashing (LSH)-based search approach [9]. CAM arrays are employed to find the Hamming distance between the stored binary signatures (based on LSH) and a query via a single, parallel search that obviates the need for data transfer (e.g., from DRAM-to-GPU for cosine-based distance calculations). Furthermore, compact CAM cells based on

978-3-9819263-4-7/DATE20/©2020 EDAA

emerging technologies (e.g., 2 ferroelectric field effect transistor (FeFET) cells per [9]) could enable larger NN memories to support MANN algorithms. (See Sec. IV.)

Finally, in Sec. V, we consider the challenges associated with accelerating NNs that are employed in recommender systems. Recommendation systems aim to provide personalized content to a user based on his or her interests, past searches, etc. While CNNs and RNNs operate over dense features (e.g., matrices), recommender systems can operate over both dense and *sparse* features (e.g., a small subset of the numerous items for sale via Amazon). A user's interactions are captured with sparsely indexed embedding tables that inject large irregular memory accesses into the inference stage and complicate the training process. Recommender models may employ different NN architectures, and can be either compute dominated or constrained by memory capacity and bandwidth. Given the above, this paper concludes with a discussion of how one might meet the computational needs of recommendation models – which (i) can dwarf CNN/RNN workloads in data centers and (ii) may be retrained as often as daily.

II. Analog Crossbar Arrays

When considering hardware for deep learning acceleration, matrix multiplications consume most of the execution time for many typical NNs representing speech, language, and vision processing [10]. Therefore, this operation is an attractive target for hardware acceleration, which accounts for the success of graphical processing units (GPUs) in deep learning.

Recently, **reductions in arithmetic precision** have emerged as a pathway to improved NN inference and training efficiency. With proper algorithmic advances, numerical precision in deep NN processing can be reduced from a conventional 32-bit floating point format without adversely affecting accuracy. For example, new techniques to maintain the fidelity of gradient computations during backpropagation (in addition to other modifications), can facilitate training with 8-bit floating point numbers without accuracy degradation [11] – even for challenging tasks [12]. Circuits optimized to execute reduced-precision arithmetic are being developed for the realization of custom accelerators for training, which can dramatically improve both throughput and power efficiency [10], [12]. For deep learning inference, innovations include a clipping parameter that is optimized during training to find the optimum activation quantization scale, and a statistical method to determine a scaling factor that minimizes the weight quantization error. As such, state-of-the-art classification accuracy across a range of popular models and datasets is achievable with just 2-bit integer weights and activations [13]. Still, the use of reduced precision computations in deep learning is not a *carte blanche*; rather, which algorithmic components may benefit must be carefully considered.

A. Crossbars for Inference and Training

The robustness of deep learning algorithms to reduced precision motivates the reconsideration of analog computation. For example, in a crossbar array of programmable resistive devices, the weight matrix can be stored via conductance values at each crosspoint. A fully parallel vector-matrix multiplication (VMM) – the main building block of generalized matrix multiplication and convolution computations during a forward pass – can then be performed by applying a voltage signal to the rows and reading out the current vector from the columns given the current-voltage relationship: $I_{ij} = \sum G_{ij} V_i$ (Fig. 1 (left)). Essentially, Ohm's law and Kirchhoff's law are used to perform multiplication and summation in the analog domain [15], [16]. Due to the simplicity of the operation, many resistive device technologies originally developed for digital non-volatile memory (NVM) and other applications have been proposed for VMM acceleration.

While the forward pass is sufficient for inference workloads, when analog hardware is employed for training, it must also support both backward pass and weight update operations. In view of their local weight storage and data processing capabilities, the resistive cross-point devices targeting training workloads are often referred to as 'Resistive Processing Units' (or RPUs) [14]. In the backward pass, the VMM is performed using the transpose of the weight matrix, which can be achieved by simply swapping the functionality performed by the peripheral circuits at the rows and the columns used in the forward pass. However, in contrast to forward and backward cycles, implementing the weight update requires incremental weight changes determined by the result of a vector-vector outer product (rank-1 update). Therefore, with resistive crossbar arrays, a multiplication operation and an incremental weight update must be performed locally at each cross-point element. As such, instead of an application of Ohm's and Kirchoff's laws, device switching characteristics must be carefully considered, and different (non-trivial) pulsing schemes must be supported by peripheral circuitry. As illustrated in Fig. 1 (right), the rank-1 update of crossbar arrays can performed by using stochastic pulses [14], and each coincidence can event increment or decrement the conductance of the crossbar element. Alternatively, deterministic pulsing schemes have also been proposed [17], [18].

A crossbar array of RPU devices can perform weight matrix operations locally and in parallel, and hence achieves O(1) time complexity in all three cycles (independent of array size). Moreover, conceptually, such an architecture has the potential to provide communication as well as computational benefits: (i) calculations are performed in place, with stationary weights, reducing the need for weight data transfer between processor and dynamic RAM (DRAM); (ii) a digital multiply-and-add involves many logic device operations, while the analog equivalent is performed in a single operation by a single weight element. Thus, lower energy per operation should be achievable, given optimized device characteristics.

As intriguing as the concept of analog computing is, its practical realization requires custom circuits as well as analog resistive devices that satisfy specifications for speed, power consumption, stability, update behavior, etc. Furthermore, these requirements depend on the intended use of the hardware. More specifically, inference applications only rely on the forward pass and require excellent long-term weight retention and stability to minimize refresh operations. In contrast, training applications rely on all three cycles – forward pass, backward pass, and weight update – resulting in more stringent requirements for incremental and symmetric weight updates and endurance.

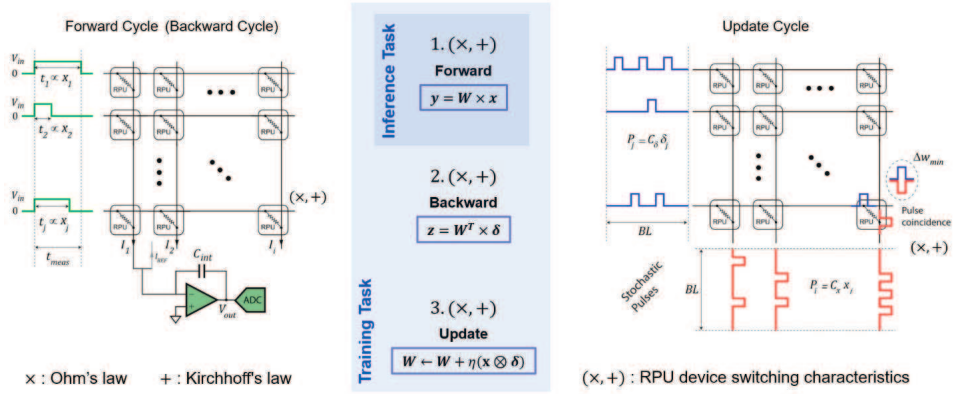

Fig. 1: Matrix-vector multiplication and rank-1 update in an analog array (after [14]). (Left) Multiply- and-accumulate operation performed by applying a voltage signal to conductances of the crossbar elements. (Right) Parallel rank-one update of all matrix elements performed by applying random voltage pulse trains to the rows and columns.

Notably, the combination of pulsing and incremental conductance switching makes the parallel update operation very different than the conventional write operations performed on NVM devices for memory applications. For each pulse coincidence, the device is expected to change its conductance by a small amount, in a process that is dictated by the underlying device and materials physics. This introduces a set of challenging device requirements.

In order to guide analog device and materials development for NN training, it is essential to establish a set of target device properties that support the successful execution of the back-propagation algorithm. Initial requirements were determined by executing a simple, fully connected network on simulated analog crossbar arrays with parameterized resistance, update, noise, and variability (cycle-to-cycle and device-to-device) characteristics [14]. Results suggest that the key technical challenge is to accumulate the gradient information in an unbiased way. Therefore, the device switching characteristics must be symmetric such that the conductance change for positive and negative voltage pulse stimuli are matched to within a few percent on average. Moreover, the change in conductance during a single coincidence event needs to be (on average) 0.1% of the whole conductance range, (although each conductance increment is not resolved by the peripheral circuits). We note that these initial device specifications were derived empirically for a small network using the MNIST dataset. They are being tested for progressively larger, more complex networks and datasets [19], [20]. In addition to these algorithmically defined specifications, due to voltage drop issues on large arrays, the RPU device resistance must be in the range of ∼10-100 MΩ.

B. Candidate Crosspoint Devices

Given these RPU specifications, we can assess **analog device technologies** proposed for analog crossbar acceleration. Device options include floating-gate or charge-trap ('Flash') memory, phase change memory (PCM), resistive random-access memory (RRAM), ferroelectric field-effect transistors (FeFET) and tunnel junctions (FTJ), and electrochemical RAM (ECRAM) among others. While digital Flash memory has been commercially available for more than three decades, a slow, high-voltage write and limited endurance may restrict analog Flash-based arrays [21] to NN inference. Herein, we focus on device options that have shown promise for NN training.

1) Phase Change Memory: We first consider **phase-change memory**, arguably the most mature among the new memory technologies. In analog PCM, conductance is increased by resistively heating an amorphous chalcogenide material such as $Ge_2Sb_2Te_5$ located between two metallic electrodes, thereby gradually crystallizing it, with resistance ranging in the tens or hundreds of kΩ [22], [23]. The metastable amorphous state is regenerated in a fast melt-quench process using a high current pulse. Therefore, PCM is a unidirectional switch. Weights are signed and need to be implemented by a differential pair of conductances: w = $G^+ - G^-$. Again as PCM is a unidirectional switch, both conductances will move towards saturation which eventually will prevent further weight updates. To avoid this situation, during training both devices periodically undergo a simultaneous reset, while maintaining the difference [18].

Experimental weight cells often show asymmetric weight updates caused by the underlying crystallization kinetics. This can be addressed by separating a weight into higher-significance and lower-significance portions. The latter will be updated more frequently and can be implemented on a capacitor with symmetric switching behavior, while the former remain in PCM, and a transfer of information from the capacitor to the PCM element(s) is performed once a threshold is reached. With this approach, test accuracy for MNIST, and for a transfer learning experiment with the CIFAR-10 dataset was found to match a digital floating-point calculation, while accuracy was slightly reduced for more challenging tasks [24].

Alternatively, one might choose to only perform matrix multiplications in the analog crossbar, while keeping the weight update in the digital domain [25]. This approach also minimizes the impact of weight update noise caused by the stochastic nature of the crystallization process [22].

Another non-ideality associated with PCM is caused by continued structural relaxation and electronic redistribution

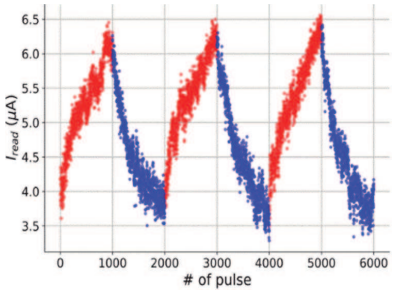

Fig. 2: Read current response during three cycles of 1000 potentiation and 1000 depression pulses applied to exemplary analog RRAM device (after [30]).

of trapped charges in the amorphous chalcogenide material, which results in a drop in conductance over time, and is known as resistance drift. To mitigate this, the PCM material can be surrounded with a metallic liner [26], or such a 'projection layer' can be placed underneath it in a lateral device geometry [27]. This liner diverts the read current around the amorphous high-resistance region, making the current insensitive to relaxation effects inside the amorphous phase, which dramatically reduces drift. We note that drift can also be compensated for by algorithmic correction in the activation function [28].

Further improvements in power efficiency are expected when increasing PCM device resistance deep into the MΩ range. For accelerators fabricated in 14 nm technology that combine such device and circuit improvements, energy efficiencies during training of up to 172- 250 TOP/s/W have been projected for base resistance of up to 100 MΩ [29]. For a recent overview of PCM-based in-memory DNN acceleration, we refer the reader to [23].

2) RRAM: We now turn to filamentary oxide-based **resistive random-access memory**. Being a bidirectional device, RRAM has the potential to reduce the circuit overhead required to accommodate the unidirectionality of phase-change memory. In an RRAM device, a metal oxide such as HfO_2 [22] located between two metallic electrodes is subjected to a one-time forming pulse in which an oxygen-deficient conductive path is formed between the electrodes. During analog operation, suitable voltage pulses (in the nanosecond range) gradually shrink or grow the filament (i.e., conductance reduction/reset and conductance increase/set respectively). Typically, a transistor is used in a 1T1R configuration in order to precisely control the forming and switching currents.

Analog RRAM typically exhibits a device resistance in the range of a few kΩ to tens of kΩ [22], [31], [32], resulting in voltage drops along the interconnect lines which limits array size. Other common challenges include imperfect yield, asymmetric weight updates, and a stochastic update behavior caused by the underlying atomistic switching mechanism [22], [30] (Fig. 2). Non-yielding, stuck-open devices can be accommodated by in-situ training [31] or hardware-aware training [33], substantially improving inference accuracy when compared to a conventional model trained with floating-point arithmetic. Regarding update asymmetry, with carefully optimized pulse conditions in the 1T1R configuration, oxygen vacancies are moved in a more incremental fashion. Promising update symmetricity and linearity have been achieved in this way [34], although there typically is a trade-off with signal-to-noise ratio. Fortunately, recent algorithmic advances relax the symmetricity requirements compared to the original RPU specifications [30], [35], giving the materials scientist more freedom to optimize other device properties.

3) Ferroelectrics: Hafnium-based oxides can instead be deployed in **ferroelectric field-effect transistors** (FeFETs) or **tunnel junctions** (FTJs) [36], making use of a ferroelectric phase of HfO_2. Originally proposed in the early 1990s [37], the threshold voltage of a synaptic ferroelectric field-effect transistor (FeFET), and thus its drain current, is controlled by the polarization state of the gate dielectric, which can be adjusted by suitable positive and negative gate voltage pulses. Crucially for NN training, FeFETs permit faster write operations at lower voltage than the conceptually similar Flash memory. Asymmetric weight update behavior (similar to what is observed for RRAM), can be accommodated in a 2T-1FeFET weight cell [38], by employing a mixed-precision concept similar to what has been demonstrated for PCM. Endurance is currently limited to approximately 10^6 to 10^9 cycles, and improvements require device, materials, and process innovation [39]. Alternatively, metal-ferroelectric-metal (MFM) ferroelectric tunnel junctions (FTJ) can be employed, as MFM devices feature better cycling endurance than FeFETs. Structurally similar to RRAM, the current through such a two-terminal device is controlled by polarization-induced changes to the tunneling barrier instead of a filament. Recently, analog, bidirectional resistance tuning in a simple, CMOS-compatible $TiN/HfO_2/TiN$ FTJ was demonstrated [40]. Challenges include asymmetric updates, stochasticity, and controlling an analog, mixed FE domain state in highly scaled devices.

4) Electrochemical RAM: The device types discussed thus far were originally devised for digital memory. **Electrochemical RAM (ECRAM)** is instead based on intrinsically analog materials inspired by solid-state battery technology. In three-terminal ECRAM, a gate electrode drives mobile ions into or out of a channel material such as $Li_{1-x}CoO_2$ [41] or WO_3 [42] to reversibly change the resistance between a source and a drain electrode. This separation of read and write operations leads to good control over the channel modulation. Highly symmetric potentiation and depression characteristics with ∼1000 up- and down-steps, and excellent SNR have been demonstrated with gate current pulses in the nanosecond range [42]. Voltage pulsing would obviate the compliance transistors required for current control, leading to a more compact design. However, devices demonstrated to date exhibit a non-zero open circuit potential, resulting in poor retention and asymmetric update characteristics when gate current control is not employed [43]. Also, most ECRAM devices demonstrated to date use lithium, which is not semiconductor-fab-friendly. Materials innovation is therefore needed. Recently, parallel array operation was demonstrated on a CMOS-compatible metal oxide ECRAM [44].

5) Paths Forward: As we have seen, challenges remain for each device option, in particular regarding weight update symmetricity under constant voltage pulsing. Optimum NN

processing will therefore benefit from adjustments to the computational algorithms to accommodate such non-idealities. Recently, a 'zero-shifting' technique was introduced that can compensate the imbalance between potentiation and depression for bidirectional but asymmetric devices. This technique shifts the weight range such that the zero weight corresponds to the conductance value (symmetry point) where the positive and negative pulse stimuli result in equal amounts of conductance change [30]. Building on this technique, one can substantially relax the symmetricity requirement. Device asymmetry introduces an unintentional implicit cost term into the conventional stochastic gradient descent (SGD) algorithm, causing it to find non-optimal solutions to the training task. Building on the zero-shifting technique, a modified training algorithm has been developed that introduces a coupled dynamical system to simultaneously minimize the original objective function of the network and the unintentional cost term in a self-consistent fashion [35].

Training simulations performed on various network architectures show that even aggressive bidirectional device asymmetry (as in the case of RRAM devices) can be compensated for by this algorithm, giving training results that are indistinguishable from those achieved with perfectly symmetric, ideal devices. Moreover, all operations performed on the crossbar arrays are still parallel and therefore the implementation cost of this new algorithm is minimal. It also maintains expected power and speed benefits. Assuming other device specifications are within tolerable margins, this algorithm allows for non-symmetric device technologies to be used for deep learning applications. In addition, even non-yielding, corrupt devices can be accommodated by hardware-aware training via randomly dropping connections [33]. Such algorithmic improvements are crucial to relaxing the material specifications and to realizing technologically viable resistive crossbar arrays that outperform digital accelerators for similar training tasks.

III. Crossbar-based Hardware for MANNs

A critical requirement for many machine learning applications is *one-shot or few-shot learning*, where the learning algorithm needs to rapidly assimilate new concepts from one or very few training examples. MANNs are emerging as a new direction to address these challenges [45], [46]. MANNs are composed of a controller, which is typically a feedforward or recurrent deep NN, enhanced with an external *differentiable memory module*, per the Neural Turing Machine (NTM) in Fig. 3. The differential memory is used as a working memory that is addressed through read and write heads with keys produced by the controller. Recent efforts from Google's DeepMind, Facebook AI, etc. have demonstrated the ability of MANNs to solve new classes of problems well beyond the capabilities of classical DNNs [3], [4], [46], [47].

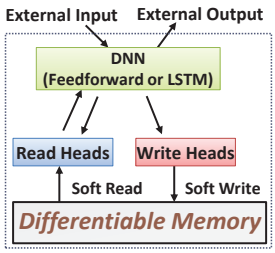

Fig. 3: Neural turing machine

To make the memory differentiable, NTMs use *soft read* and *soft write* operations, which are extremely memory intensive since each soft read or write requires access to *all the locations* in the memory. The extent to which each memory element is accessed is controlled by an attentional focus mechanism that leverages various similarity measures, *e.g.*, cosine similarity, Euclidean distance, etc. Consequently, the differentiable memory is a major performance and energy bottleneck in current (CPU and GPU) implementations of MANNs. This bottleneck will only grow when dealing with real-world data requiring thousands to millions of memory locations [4]. Specialized DNN accelerators such as Google's TPU, Microsoft's Brainwave, and several others, are tuned to the computational kernels of classical DNNs. While they can efficiently execute the controller network of MANNs, they do not address the memory-intensive kernels arising from soft reads and writes to the differentiable memory.

Per the above, resistive crossbars are considered promising as building blocks of future neural hardware fabrics due to their ability to exploit massive degrees of parallelism when performing in-memory dot product operations. X-MANN introduces a specialized crossbar-based architecture that is designed to efficiently accelerate the key differentiable memory operations in MANNs. Fig. 4 provides an overview of the X-MANN architecture, which is hierarchically organized into multiple banks, with each bank comprising multiple subarrays. Each subarray is further composed of *transposable crossbar-based processing tiles* (TCPTs) that are connected via a shared bus. The differentiable memory state is partitioned and stored across the different tiles in a distributed manner. To reduce the partial outputs from these tiles, X-MANN also includes a *global reduce unit*. The following sections describe the key components of the X-MANN architecture.

A. Transposable Crossbar-based Processing Tile

The transposable crossbar-based processing tile can perform in-memory computations along both its rows and columns. Two sets of register banks (key registers and weight registers) feed inputs to the columns and rows of the crossbar array within the TCPT, and an output buffer holds temporary outputs from the array. A Special Function Unit (SFU) interfaces with the output buffer to execute simple kernels with lower memory intensity. A decode and control unit takes the operation type ($OpType$) as input and generates the control signals, *viz.*, column/row wordlines (CWLs/RWLs), column/row source lines (CSLs/RSLs). Below, we describe the transposable crossbar array, how it realizes similarity and soft read operations, and finally outline the design of the SFU.

1) Transposable Crossbar Array: In X-MANN, the standard resistive crossbar array is enhanced with the additional capability of providing input voltages along the columns and reading out the output currents along the rows, making it a transposable array. Dedicated DACs are placed on rows and columns, while ADCs are shared across columns and rows due to their higher overheads. Dot product operations along both the rows and the columns form the basis for the soft read, soft write and the similarity measure operations in MANNs. The following paragraphs describe how two key operations on the differentiable memory (*i.e.*, similarity measure and soft read) are realized in the transposable array.

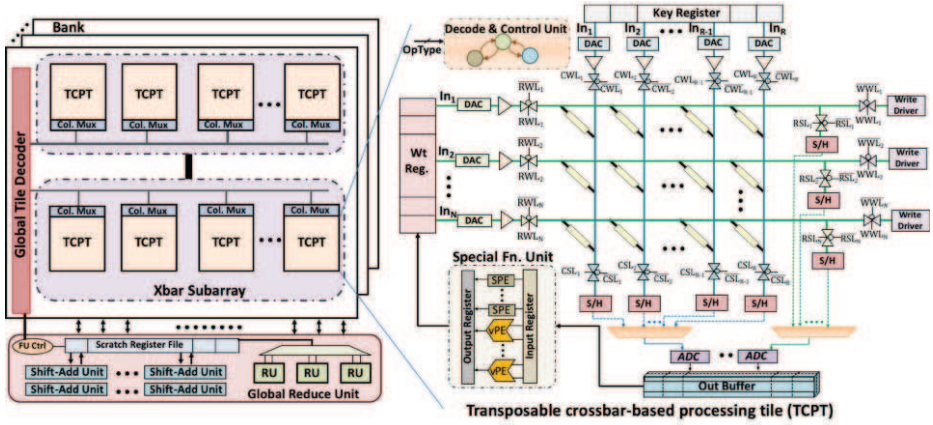

Fig. 4: X-MANN architecture: Overview

2) Similarity Measure: A similarity measure operation is realized in two steps. First, elements of the key vector are fed as inputs along crossbar columns by enabling the column wordlines (CWLs). Each memory vector, stored as conductances along different crossbar rows, is multiplied with the input key vector. The resulting output current across each row represents the dot product of the corresponding memory and key vectors. Next, voltages equivalent to a vector of all 1s are applied along the columns to produce L1-Norms of all memory vectors, across each row, in parallel. Thus, a transposable array can compute both the dot products and L1-Norms for the entire set of memory vectors in *two crossbar operations*. Using the computed dot products and L1-Norms, the SFU performs the remaining operations required to compute similarity measures for each memory vector.

3) Soft Read: A soft read is identical to the vector-matrix multiplication realized in standard crossbars and can be performed in a *single crossbar operation*. The voltages corresponding to the weights for each memory vector are applied to the rows. The resultant current flowing through each column is received by enabling the appropriate S/H circuit via the column source line (CSL). The output read across each column represents an element of the resulting soft read vector.

4) Special Function Unit: To minimize the data transfer cost for the operations that need to be performed outside the transposable array, X-MANN introduces a near-memory SFU that is tightly integrated with the output buffer in the TCPT. The SFU contains two different types of compute elements, *viz.* a vector Processing Element (vPE) and a Special Processing Element (SPE). The vPE contains a vector of adder and multiplier units, while the SPE contains logic needed to support functions such as exponentiation and division. The SFU is primarily designed to perform operations such as softmax, division, multiplication and addition/subtraction, required for similarity measure and soft writes.

B. XMANN Performance

For a suite of MANN benchmarks with diverse memory capacities, X-MANN achieves 23.7×-45.7× speedup and 75.1×-267.1× reduction in energy over a state-of-the-art GPU.

IV. CAM-based Hardware for MANNs

CAMs have also been proposed to accelerate and simplify attentional operations by performing distance metric calculations for similarity-based searches in memory; this also eliminates the need for data updates at random memory addresses, which can be prohibitively expensive on a GPU [9], [48]. Follow-on work has combined CAM-based approaches with Compute-in-Memory (CiM) hardware to support combinations of distance metrics to further improve classification accuracy in few-shot learning applications [49].

A. CAM-based MANNs

As noted earlier, augmenting a NN with an external memory can improve its abilities to "learn how to learn" and prevent catastrophic forgetting [50]. Training frequently occurs by applying a series of N-way, K-shot examples. A NN may be shown a set of supporting images from N classes, with K examples of each class. It is then shown new query images (from the N classes) and must identify which supporting images the query images are most similar to. Since a traditional NN updates its weights with a bias towards the most recent example it has seen, it can overfit on newer examples. Adding an external memory prevents this bias by caching information that it has previously seen [51], which can be used by the NN for better generalization. Memory entries are then retrieved based on their similarity with a query vector.

While recent work suggests that this approach can be effective from the perspective of accuracy [2], [4], [52], most research to date has considered standard datasets such as Omniglot, and acknowledges that substantially larger memories are needed as problem complexity increases – e.g., language processing at scale, etc. [4], [52]. Furthermore, networks with an attentional memory (i.e., MANN-like networks) often require a large number of floating-point operations and requisite data transfer – e.g., to perform cosine similarity metric calculations for necessary attentional operations, which can greatly increase energy consumption and hardware cost.

An attractive hardware alternative for performing parallel comparisons is to use CAMs. Unlike a conventional computer memory where each data word is retrieved from a provided

address and then compared, with a CAM or ternary CAM (TCAM), data is provided as input to memory, the closest possible match can be returned via a single reference, and no data transfer is required. With a TCAM cell, it is possible to store a "don't care" state, which is useful for encoding ranges [53], [54], and is employed in initial work.

Still, using CAMs for similarity metric calculations introduces challenges – e.g., cosine similarity calculations necessitate many multiplication and division operations, which are not supported per the above. Thus, there have been systematic studies of alternative distance metrics including L_1, L_2 and L_∞ (that may be more CAM friendly) to determine the resulting impact on accuracy, physical memory size required, and the number/type of operations to compare a query with a network's learned memory.

B. Evaluation of Different Encoding Schemes

As CAMs/TCAMs only support Hamming distance computation, data encodings can significantly impact the the accuracy of the resulting MANNs. Below we discuss representative encodings and their associated performance.

1) Range Encoding: Range encoding with no expansion (RENE) [53], [54], a TCAM similarity-based search, leverages binary reflected gray code (BGRC) range encoding. The query point is translated to a cube of increasing sizes until the k-nearest neighbor is found. As range size increases, the number of "don't care" states increases as well. Thus, range encoding can be used to compute nearest neighbor metrics such as L_1, L_∞, etc. RENE's range-encoding scheme has been used and applied to MANNs [48]. A lifelong learning memory module consisting of key-value pairs are added to a NN consisting of a 4-layer convolutional NN and 2-layer fully connected network. The lifelong learning memory module acts as an associative working memory while the NN learns, comparing the current input query to the past entries stored in the memory based on their similarity. To enable the use of TCAMs, floating point-based feature vectors are converted to a fixed-point representation.

While FOM such as latency and energy must be studied, for TCAM-based MANNs to be viable, they must also offer task-level accuracies that are comparable to a GPU-baseline – e.g., that uses a cosine distance metric. As a representative example, by employing combined L_∞ and L_2 approach with a 4-bit, fixed point precision with 512 memory entries, we can achieve a 96.00% classification accuracy for the Omniglot 5-way, 1-shot classification task [48] By comparison, a 32-bit floating point-based cosine similarity based MANN has a 99.06% accuracy for the same task. With this approach, essentially only a few TCAM lookups are required versus $M \cdot D$ multiplications required by the attentional memory with a 32-bit floating point-based cosine similarity search (where M is the number of memory entries, and D is the number of dimensions of the feature embeddings) [48].

With binary comparators (i.e., an XNOR gate), to find the K nearest neighbors of the search vector among the stored vectors, multiple consecutive searches need to be performed [48]. Thus, each cube-based query point might represent a TCAM lookup/array reference, multiple references may be required to identify KNN, and we still incur energy and latency overheads from charging/discharging the match lines (MLs) in a TCAM array multiple times (i.e., with each search) [48]. It is much more desirable for the TCAM to compute any requisite distance norm with just a single search – while simultaneously computing the degree of match.

2) Locality Sensitive Hashing: As the mismatch between the search vector and stored vector increases (or decreases), a match-line (ML) discharges faster (or slower) [48], [55]. Therefore, it is possible to compute the degree of match by sensing the discharge rates of the MLs. Note that a TCAM only computes the Hamming distance (HD) norm between the query and the stored vectors, through direct measurement of the number of mismatched bits. To address this problem, LSH functions can be used to hash real value feature vectors to a binary signature [56] as an intermediate step, and then utilize a TCAM array to find the HD between the binary signatures (Fig. 5). LSH encodes the feature vectors such that similar vectors will be mapped with the same signature. The stored vector with the minimum HD from the query vector is chosen as the nearest vector. With this approach, CAMs can be employed to implement the external memory in a MANN, and calculate the distance between the requested search vector and all the memory entries with just a single parallel search across the array, which can significantly improve both energy and latency in memory search operations versus a cosine similarity calculation performed on a GPU backed by DRAM.

One- and few-shot learning tasks given this context have been considered for the Omniglot dataset [57]. The last fully connected layer in a CNN (that generates feature vectors) is replaced with an LSH function layer [58] (Fig. 5). The LSH layer is used to convert features from a real-valued feature vector to a binary signature As the LSH and the original fully connected layer have similar computational demands, the replacement does not incur overhead in either storage or computation. LSH with random projections is used such that feature vectors (e.g., images) associated with the same class have similar binary signatures. Thus, a greater (or smaller) HD between the binary signatures implies less (or more) similarity between the features represented by the signatures. By storing the binary signatures in the TCAM, one can search in parallel for the supporting class that is closest to a given query. The number of LSH hashing planes is a hyper-parameter and is tuned until further increase does not further improve accuracy.

The TCAM-based MANN exhibits classification accuracies that approach (and sometimes match) those obtained with the conventional cosine similarity calculation when implemented on a GPU backed by external DRAM (Fig. 5 (inset)). Furthermore, we observe 24X and 2,582X and reductions in energy and latency, respectively, for memory search operation when a 16T CMOS TCAM replaces DRAM. As such, alternative hardware architectures have the potential to substantially improve inference energy and latency.

C. Paths Forward

Per Sec. I, [9] has reported an FeFET TCAM array prototype that implements an in-memory HD compute function between the query vector, and all the vectors stored in a TCAM array. The FeFET TCAM cell is comprised of just 2 FeFETs – as opposed to a 16T CMOS TCAM cell. Studies

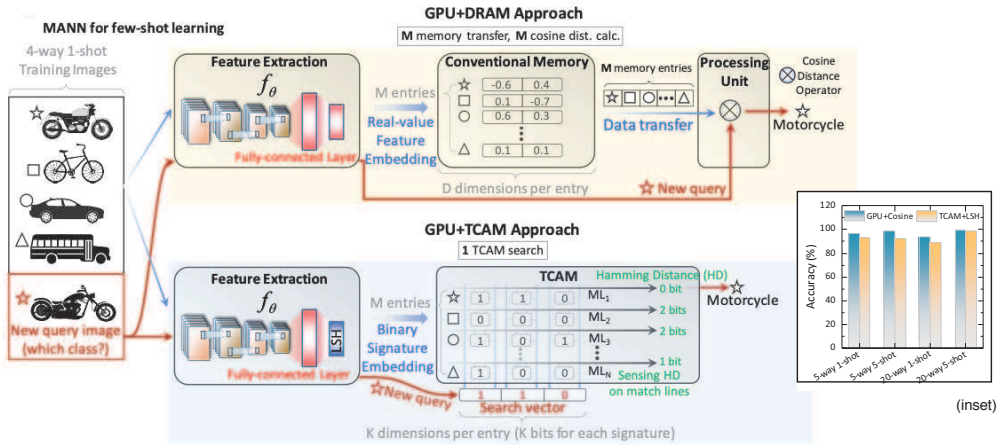

Fig. 5: GPU-based vs. TCAM-based MANNs (from [9]); (inset) classification accuracy (cosine distance vs. LSH).

suggest that replacing 16T CMOS TCAMs with 2 FeFET TCAMs can further reduce the latency and energy for memory search operations in MANNs by 1.1X and 2.4X respectively. A more compact FeFET design could also enable larger MANN memories. That said, issues related to FeFET endurance, how to best exploit cell non-volatility, and improved accuracy (among others) must still be considered. For example, per Fig. 5 (inset), when TCAM-based distance metrics are employed, not all few-shot learning problems approach iso-accuracy. One potential solution is to choose hash values such that nearby feature vectors (in a Euclidean sense) are mapped to hash values that are close in HD. The efficacy of this approach with more sophisticated datasets (e.g., omniglot versus mini-ImageNet) must also be investigated.

V. Emerging Machine Learning Workloads

Personalized recommendation is an important and emerging class of ML workloads. Recommendation is the task of serving personalized content to users based on their preferences. Personalization forms the backbone of many, pervasive internet services such as search, e-commerce, social media, news, and entertainment [59]. Central to these services is the ability to accurately rank content based on the preferences of individual users. Following recent advances from the ML and AI communities, state-of-the-art recommender systems rely on deep learning methods [59]–[61]. In fact, NNs for recommender systems (not CNNs/RNNs) now comprise an overwhelming majority of cycles devoted to AI inference in datacenters [62]–[64]. The efficient deployment of emerging NN based recommender systems will require the design of novel hardware platforms.

A. Unique neural network architecture

To recommend content based on users' personal preferences, NNs for recommendation employ a unique architecture. While CNNs and RNNs operate over continuous, *dense* features (i.e., vectors, matrices, and images), inputs to recommendation models include both dense and categorical, *sparse* features. Categorical features are key to enabling highly accurate recommendation models. While categorical features represent users' interactions with possible items to be recommended, a typical user only "interacts" with a handful of (potentially millions) of items (e.g., videos on Youtube, goods on Alibaba, social media posts on Facebook, etc.). These interactions are represented as multi-hot encoded vectors, which not only makes training more challenging, but also requires (i) intrinsically different operations (e.g., embedding tables) and (ii) model architectures (to process which require novel compute and memory systems for efficient execution)

Fig. 6 is a simplified diagram of an **execution flow** for a NN based recommendation system. As previously mentioned, inputs to the model are a collection of dense and sparse features. Dense features are processed by a series of DNN layers (typically multi-layer perceptron (MLP) layers). Sparse features are first transformed to a dense representation via embedding tables. (While the sparse to dense projection could be implemented as MLP layers or matrix factorization, the compute demands for doing so are large when compared to simpler hash-based embedding lookups). Based on the multi-hot encoded vector, the corresponding rows of the embedding table (which represent learned latent feature vectors) are pooled. The aggregated latent feature vectors are concatenated across embedding tables, along with the output of the dense-feature DNN stack, and processed by the final predictor DNN stack. The output of the network is the predicted "click-through-rate" of the user interacting with a given item.

B. Research challenges and opportunities

The efficient deployment of NN-based recommendation models poses a number of open research challenges. Compared to CNNs and RNNs, recommendation models incur orders of magnitude larger storage capacity and irregular memory access patterns. As such, the **memory system** is of the utmost importance. First, state-of-the-art recommendation models range

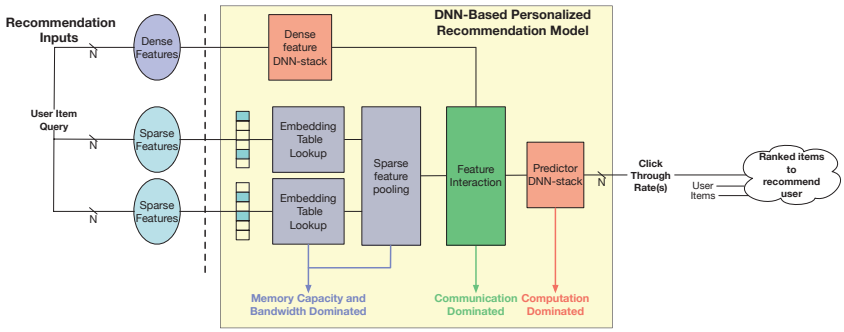

Fig. 6: Simplified model-architecture of NN based recommendation systems.

from hundreds of MBs to tens of GBs in capacity [59]. This is a result of large embedding tables where the number of rows scales with the range of possible items to recommend (e.g., millions) and the column dimension is set by the latent feature size (e.g., tens). While recent work has applied reduced precision to compress embedding tables by up to 16×, a recommendation model's capacity remains well above the capability of traditional local, on-chip storage in state-of-the-art hardware platforms [65].

Embedding table operations exhibit orders of magnitude lower compute intensity as compared to CNN and MLP operations [59]. Namely, the large capacity and irregular memory accesses (i.e., from the multi-hot encoded sparse input features), results in performance constraints due to the associated memory references. Accelerating these operations will require the co-design of the memory system with the recommendation models and could leverage techniques such as caching, prefetching, and near memory processing [66].

In addition to unique memory system challenges, recommendation models incorporate a **variety of NN architectures** – i.e., the recommendation model in Fig. 6 can be configured to represent a variety of NN architectures [59]–[61], [67]–[69]. If the model is configured to have large, dense feature DNN-stacks and predictor DNN stacks, it will be compute dominated. Alternatively, a larger number of embedding tables and multi-hot encoded sparse input vectors results in a model that is constrained by memory capacity and bandwidth. Furthermore, emerging recommendation models rely on explicitly modeling sequences of user interactions and interests with RNNs and attention. Given the diversity of employed NN topologies, accelerators for emerging recommendation systems must carefully balance specialization with flexibility.

Finally, in addition to improving inference efficiency, the **training** of recommendation models also poses challenges. Training highly accurate recommendation systems to provide personalized user experiences requires large datasets with millions of users and interactions that are on the order of tens of GBs [70]. Given the compute and storage requirements of the recommendation model and the size of training datasets, state-of-the-art recommendation models are typically trained across many machines [62]. Thus, efficient training requires carefully balancing compute, memory, and network communication. Furthermore, to capture evolving user interests, recommendation systems are periodically updated [62]. For instance, Facebook's recommendation models are re-trained on hourly and daily intervals. As each training run spans many hours, open research challenges remain with respect to designing hardware friendly meta-learning and few-shot learning techniques for recommendation systems.

VI. Conclusion

We have reviewed efforts toward the realization of analog crossbar architectures that can be used to support existing NN models such as RNNs and CNNs, as well as emerging network models such as MANNs. As described herein, CAM arrays have also been proposed to support emerging models such as MANNs. (Moreover, CAM-based architectures may also benefit from crossbar arrays, as "helper networks" to generate feature embeddings that may take the form of a CNN, etc.) Furthermore, NN models for problems such as recommendation systems face challenges in that often there is not a single, dominating kernel function that could serve as a potential acceleration target. We hope this discussion will further engage the design automation community with respect to architectural design space exploration efforts, etc. to address computational workloads associated with new NN models.

Acknowledgments

This work was supported in part by ADA, ASCENT, and C-BRIC – three of the six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] M. McCloskey et al. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.
[2] A. Santoro, et al. Meta-Learning with Memory-Augmented Neural Networks. In *ICML*, pages 1842–1850, 2016.
[3] A. Graves, et al. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
[4] A. Graves, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.
[5] O. Vinyals, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.
[6] Ł. Kaiser, et al. Learning to Remember Rare Events. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
[7] A. Ranjan, et al. X-MANN: A Crossbar based Architecture for Memory Augmented Neural Networks. In *DAC*, page 130, 2019.
[8] A. Graves, et al. Neural Turing Machines. abs/1410.5401, 2014.

[9] K. Ni, et al. Ferroelectric Ternary Content Addressable Memory for One-Shot Learning. *Nature Electronics*, 2:521–529, November 2019.

[10] S. Shukla, et al. A scalable multi-teraops core for ai training and inference. *IEEE Solid-State Circuits Letters*, 1(12):217–220, Dec 2018.

[11] N. Wang, et al. Training deep neural networks with 8-bit floating point numbers. In *Advances in neural information processing systems*, pages 7675–7684, 2018.

[12] X. Sun, et al. Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks. In H. Wallach, et al., editors, *Advances in Neural Information Processing Systems 32*, pages 4901–4910. 2019.

[13] J. Choi, et al. Accurate and efficient 2-bit quantized neural networks. *2nd SysML Conference*, 2019.

[14] T. Gokmen et al. Acceleration of deep neural network training with resistive cross-point devices: Design considerations. *Frontiers in Neuroscience*, 10:333, 2016.

[15] B. Widrow. An adaptive 'Adaline' neuron using chemical 'memistors', 1960. TR No. 1553-2, Stanford Electronics Laboratories, Stanford University, Stanford, Calif.

[16] K. Steinbuch. Die lernmatrix. *Kybernetik*, 1(1):36–45, 1961.

[17] Z. Xu, et al. Parallel programming of resistive cross-point array for synaptic plasticity. *Procedia Computer Science*, 41:126–133, 2014.

[18] G. W. Burr, et al. Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices*, 62(11):3498–3507, 2015.

[19] T. Gokmen, et al. Training deep convolutional neural networks with resistive cross-point devices. *Frontiers in Neuroscience*, 11:538, 2017.

[20] M. J. Rasch, et al. Training large-scale anns on simulated resistive crossbar arrays. abs/1906.02698, 2019.

[21] M. R. Mahmoodi et al. An Ultra-low Energy Internally Analog, Externally Digital Vector-matrix Multiplier Based on NOR Flash Memory Technology. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, pages 22:1–22:6, 2018.

[22] N. Gong, et al. Signal and noise extraction from analog memory elements for neuromorphic computing. *Nature Comm.*, 9(1):2102, 2018.

[23] E. Eleftheriou, et al. Deep learning acceleration based on in-memory computing. *IBM J. of Res. and Development*, 63(6):7:1–7:16, Nov 2019.

[24] S. Ambrogio, et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, 558(7708):60, 2018.

[25] S. R. Nandakumar, et al. Mixed-precision architecture based on computational memory for training deep neural networks. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2018.

[26] W. Kim, et al. Reliability benefits of a metallic liner in confined PCM. In *2018 IEEE International Reliability Physics Symposium (IRPS)*, pages 6D.5–1–6D.5–5, March 2018.

[27] I. Giannopoulos, et al. 8-bit precision in-memory multiplication with projected phase-change memory. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 27.7.1–27.7.4, Dec 2018.

[28] V. Joshi, et al. Accurate deep neural network inference using computational phase-change memory. abs/1906.03138, 2019.

[29] H.-Y. Chang, et al. AI hardware acceleration with analog memory: Microarchitectures for low energy at high speed. *IBM Journal of Research and Development*, 63(6):8:1–8:14, Nov 2019.

[30] H. Kim, et al. Zero-shifting technique for deep neural network training on resistive cross-point arrays. abs/1907.10228, 2019.

[31] C. Li, et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nature Communications*, 9(1):2385, 2018.

[32] R. Mochida, et al. A 4M Synapses integrated Analog ReRAM based 66.5 TOPS/W Neural-Network Processor with Cell Current Controlled Writing and Flexible Network Architecture. In *2018 IEEE Symposium on VLSI Technology*, pages 175–176, June 2018.

[33] T. Gokmen, et al. The marriage of training and inference for scaled deep learning analog hardware. In *IEEE International Electron Devices Meeting (IEDM)*, 2019.

[34] E. A. Cartier, et al. Reliability challenges with materials for analog computing. In *2019 IEEE International Reliability Physics Symposium (IRPS)*, pages 1–10, March 2019.

[35] T. Gokmen et al. Algorithm for training neural networks on resistive device arrays. *arXiv preprint arXiv:1909.07908*, 2019.

[36] S. Oh, et al. Ferroelectric materials for neuromorphic computing. *APL Materials*, 7(9):091109, 2019.

[37] H. Ishiwara. Proposal of adaptive-learning neuron circuits with ferroelectric analog-memory weights. *Japanese Journal of Applied Physics*, 32(Part 1, No. 1B):442–446, jan 1993.

[38] X. Sun, et al. Exploiting Hybrid Precision for Training and Inference: A 2T-1FeFET Based Analog Synaptic Weight Cell. In *International Electron Devices Meeting (IEDM)*, pages 3.1.1–3.1.4, Dec 2018.

[39] S. Oh, et al. Improved endurance of hfo2-based metal- ferroelectric-insulator-silicon structure by high-pressure hydrogen annealing. *IEEE Electron Device Letters*, 40(7):1092–1095, July 2019.

[40] M. Frank, et al. Analog resistance tuning in TiN/HfO_2/TiN ferroelectric tunnel junctions. In *49th IEEE SISC*, 2018.

[41] E. J. Fuller, et al. Li-ion synaptic transistor for low power analog computing. *Advanced Materials*, 29(4):1604310, 2017.

[42] J. Tang, et al. ECRAM as Scalable Synaptic Cell for High-Speed, Low-Power Neuromorphic Computing. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 13.1.1–13.1.4, Dec 2018.

[43] D. Bishop, et al. Time resolved conductance in electrochemical systems for neuromorphic computing. In *International Conference on Solid State Devices and Materials*, page 23, 2018.

[44] S. Kim et al. Metal-oxide based, CMOS-compatible ECRAM for deep learning accelerator. In *IEEE IEDM*, pages 35.7.1–35.7.4, 2019.

[45] A. S. et al. One-shot Learning with Memory-Augmented Neural Networks. *CoRR*, abs/1605.06065, 2016.

[46] J. Weston et al. Memory Networks. *CoRR*, abs/1410.3916, 2014.

[47] S. Sukhbaatar et al. End-to-end memory networks. *CoRR*, abs/1503.08895, 2015.

[48] A. F. Laguna, et al. Design of Hardware-Friendly Memory Enhanced Neural Networks. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1583–1586. IEEE, 2019.

[49] A. F. Laguna, et al. Ferroelectric FET Based In-Memory Computing for Few-Shot Learning. In *GLSVLSI*, pages 373–378.

[50] S. Hochreiter, et al. Learning to Learn Using Gradient Descent. In *2001, International Conference Artificial Neural Networks (ICANN) Vienna, Austria, August 21-25, 2001 Proceedings*, pages 87–94, 2001.

[51] S. Hochreiter et al. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[52] L. Kaiser, et al. Learning to Remember Rare Events. abs/1703.03129, 2017.

[53] A. Bremler-Barr, et al. Ultra-Fast Similarity Search Using Ternary Content Addressable Memory. In *Proceedings of the 11th International Workshop on Data Management on New Hardware, DaMoN 2015*, pages 12:1–12:10, 2015.

[54] A. Bremler-Barr et al. Space-Efficient TCAM-Based Classification Using Gray Coding. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 1388–1396, May 2007.

[55] M. Imani, et al. Approximate Computing Using Multiple-Access Single-Charge Associative Memory. *IEEE Transactions on Emerging Topics in Computing*, 6(3):305–316, July 2018.

[56] A. Andoni et al. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *CACM*, 51(1):117, 2008.

[57] B. M. Lake, et al. Human-Level Concept Learning Through Probabilistic Program Induction. *Science*, 350(6266):1332–1338, 2015.

[58] Y. LeCun, et al. Deep learning. *Nature*, 521(7553):436, 2015.

[59] U. Gupta, et al. The architectural implications of facebook's dnn-based personalized recommendation. *arXiv preprint arXiv:1906.03109*, 2019.

[60] X. He, et al. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[61] H.-T. Cheng, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10. ACM, 2016.

[62] K. Hazelwood, et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. IEEE, 2018.

[63] C.-J. Wu, et al. Deep learning: It's not all about recognizing cats and dogs.

[64] J. Park, et al. Deep learning inference in facebook data centers: Characterization, performance optimizations and hardware implications. *arXiv preprint arXiv:1811.09886*, 2018.

[65] A. Ginart, et al. Mixed dimension embeddings with application to memory-efficient recommendation systems, 2019.

[66] Y. Kwon, et al. TensorDIMM: A Practical Near-Memory Processing Architecture for Embeddings and Tensor Operations in Deep Learning. In *International Symposium on Microarchitecture*, pages 740–753, 2019.

[67] G. Zhou, et al. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068. ACM, 2018.

[68] G. Zhou, et al. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5941–5948, 2019.

[69] M. Naumov, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv:1906.00091*, 2019.

[70] J. Ni, et al. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, 2019.