CELIA: A Device and Architecture Co-Design Framework for STT-MRAM-Based Deep Learning Acceleration

Hao Yan  
Department of Electrical and Computer Engineering  
University of Texas at San Antonio  
San Antonio, Texas, USA  
hao.yan@utsa.edu

Hebin R. Cherian  
Department of Electrical and Computer Engineering  
University of Texas at San Antonio  
San Antonio, Texas, USA  
hebin.cherian@my.utsa.edu

Ethan C. Ahn  
Department of Electrical and Computer Engineering  
University of Texas at San Antonio  
San Antonio, Texas, USA  
chiyui.ahn@utsa.edu

Lide Duan  
Department of Electrical and Computer Engineering  
University of Texas at San Antonio  
San Antonio, Texas, USA  
lide.duan@utsa.edu

ABSTRACT

A large variety of applications rely on deep learning to process big data, learn sophisticated features, and perform complicated tasks. Utilizing emerging non-volatile memory (NVM)'s unique characteristics, including the crossbar array structure and gray-scale cell resistances, to perform neural network (NN) computation is a well-studied approach in accelerating deep learning tasks. Compared to other NVM technologies, STT-MRAM has its unique advantages in performing NN computation. However, the state-of-the-art research has not utilized STT-MRAM for deep learning acceleration due to its device- and architecture-level challenges. Consequently, this paper enables STT-MRAM, for the first time, as an effective and practical deep learning accelerator. In particular, it proposes a full-stack solution across multiple design layers, including device-level fabrication, circuit-level enhancement, architecture-level data quantization, and system-level accelerator design. The proposed framework significantly mitigates the model accuracy loss due to reduced data precision in a cohesive manner, constructing a comprehensive STT-MRAM accelerator system for fast NN computation with high energy efficiency and low cost.

CCS CONCEPTS

• Computer systems organization → Neural networks; Processors and memory architectures; • Hardware → Memory and dense storage;

KEYWORDS

STT-MRAM, deep learning acceleration, device and architecture co-design.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICS '18, June 12–15, 2018, Beijing, China  
© 2018 Association for Computing Machinery  
ACM ISBN 978-1-4503-5783-8/18/06...$15.00  
https://doi.org/10.1145/3205289.3205297

1 INTRODUCTION

Deep learning has recently shown extensive usage in a wide variety of applications, such as image/speech recognition, self-driving cars, financial services, healthcare, etc. For example, convolutional neural networks (CNNs) have led to great successes in image classification, improving the top-5 accuracy of ImageNet [29] from 84.7% in 2012 to 96.4% in 2015. The recent successes of machine learning and deep learning are due to [38]: (1). the explosive development of information available for model training; (2). the fast increase of computing capacity; and (3). the development of large, generic open source frameworks. In particular, machine learning is becoming a major driving force for high performance computing [4]. Therefore, accelerating the training and inference of deep learning models has been a recent focus in designing various computing systems ranging from embedded devices to data center servers.

Deep learning applications are highly computation and memory intensive, dynamically and frequently accessing large amounts of data with low locality. As a result, the main memory has become a fundamental bottleneck in both performance and energy efficiency when running these applications. Conventional DRAM-based main memory is facing critical challenges in both latency and scalability. The DRAM latency has remained nearly unchanged in recent generations [22], and it has been extremely difficult to scale DRAM to have larger capacities [19]. Consequently, emerging non-volatile memories (NVM), including phase-change memory (PCM) [21], spin transfer torque magnetoresistive RAM (STT-MRAM) [12], and metal-oxide-based resistive RAM (RRAM) [39], are being investigated as replacements for DRAM. A wide variety of benefits can be obtained in NVM-based main memories, including almost zero idle power, no data refreshes, non-destructive reads, nearly infinite data retention time, etc. These advantages make NVM a tempting solution to replace the memory hierarchy in computer systems.
In addition to data storage, NVMs have also been used to perform neural network (NN) computation through their crossbar structure [6] [31] [33]. In such NVM-base NN accelerators, input data are represented using wordline voltages, and synaptic weights are programmed into cell conductance; the resulting bitline current gives the NN calculation result. In these designs, however, data precision is reduced in synaptic weights since only a limited number of resistance states are available in a NVM cell to represent the synaptic weight programmed in it. Similarly, input data precision is also reduced in the digital-to-analog conversion (DAC) from binaries to wordline voltages. As a consequence, the inference accuracy of the deep learning application is ultimately reduced.

In this paper, we tackle this accuracy reduction problem via device and computer architecture co-design. We propose a comprehensive framework named Convolution-Enhanced Learning In Accelerators (CELIA), and highlight two key features of it. First, CELIA is the first STT-MRAM-based NN accelerator. Existing NVM accelerators [27] [6] [31] [33] are all based on RAM (memristors). Note that SPINDLE [27] also uses memristors to store weights in its crossbar arrays. Using STT-MRAM crossbars as the accelerator incurs unique benefits and also challenges (Section 3) that will be addressed by CELIA. Second, CELIA enables a full-stack solution for deep learning acceleration across multiple design layers. At the device level (Section 4), CELIA fabricates STT-MRAM devices to have intermediate resistance states in a cell; at the circuit level (Section 5.4), CELIA connects two STT-MRAM cells in parallel to create sufficient resistances needed by the application; at the architecture level (Section 5.2), CELIA proposes a novel data quantization scheme that can better utilize the enabled resistance states to represent the deep learning model; and at the system level (Section 6), CELIA showcases the detailed design of a deep learning accelerator with improved performance and energy efficiency.

Consequently, our proposed framework significantly mitigates the model accuracy loss due to reduced data precision in a cohesive manner, constructing a comprehensive STT-MRAM accelerator system for fast NN computation with high energy efficiency and low cost. CELIA is particularly useful for data center designs for machine learning due to its processing-in-memory (PIM) characteristics, which co-locate computation with memory storage to enable large volume of low cost hybrid (computation and storage) devices. The contributions of this paper can be summarized as:

- **Creating multiple resistance states in a STT-MRAM cell.** Our device-level work relies on the strong dependence of the STT-MRAM cell resistance on the applied bias voltage. We carefully control the bias voltage sweep and its cycling history. This increases the number of resistance states in a STT-MRAM cell, thus improving the data precision of the synaptic weight stored in the cell and ultimately achieving higher application inference accuracy.

- **Non-uniform data quantization for synaptic weights.** Motivated by the observation that the most important quantization points of synaptic weights are not uniformly distributed, we propose a non-uniform data quantization scheme to identify the most important synaptic weight quantization points to the model inference. Hence, the same limited number of resistance states, when non-uniformly configured, can better represent the application model and minimize the inference accuracy loss due to reduced data precision of synaptic weights.

- **A system design for deep learning acceleration.** We further propose a comprehensive accelerator system to conduct CNNs in a pipelined manner. This framework exhibits unique innovations, including a novel crossbar allocation scheme, an energy efficient digital-to-analog converter (DAC), an analog-to-digital converter (ADC), accumulation and pooling, etc.

## 2 BACKGROUND

### 2.1 Deep Learning Basics

Deep learning applications utilize big artificial neural networks (NNs) to perform machine learning tasks. Typical examples include deep neural networks (DNNs) and convolutional neural networks (CNNs). A DNN is a neural network with more than three (typically 5 to over 1000) layers. A CNN is composed of several layers of neurons connected in a specific pattern and specialized for classifying images. As illustrated in Figure 1, a typical CNN model includes interleaved convolutional (CONV) and pooling layers, followed by a number of fully connected (FC) layers.

A CONV layer takes a number of input feature maps, convolving with fixed-sized filters (i.e., kernels) to generate a number of output feature maps. As shown in the figure, each pixel in an output feature map is the summation of dot-products between a filter (consisted of a block of synaptic weights) and a same-sized matrix of pixels from each input feature map. A different set of filters are used in calculating the same pixel in another output feature map; and the matrices of input data move around the input feature maps when they are used to calculate different output pixels. A non-linear activation function, e.g., ReLU or Sigmoid, is usually applied to the convolution result. A pooling layer down-sizes the feature maps by shrinking a block of input pixels into a single output pixel. For example, max pooling simply uses the pixel with the largest value to represent a block of input pixels. As a result, after a few interleaved CONV and pooling layers, there are an increasingly larger number of feature maps with an increasingly smaller size. These feature maps are finally flattened to form a feature vector used as the input to a FC layer, which performs fully connected NN computation (i.e., multi-layer perception) to generate an output feature vector. The values in the output feature vector of the last FC layer indicate the probability of different categories that the input
The computations incurred in CNNs are mostly matrix/vector operations. First, STT-MRAM has orders of magnitude longer cell lifetimes than PCM and RRAM. PCM employs a destructive write mechanism that limits its endurance to less than $10^{12}$ cycles [18]; cutting-edge RRAM cells also demonstrate endurance of at most $10^{12}$ cycles [11]. In contrast, bit flipping in a STT-MRAM cell is done in a non-destructive way, resulting in programming endurance of $10^{15}$ cycles [17]. This highly cyclable feature of STT-MRAM can enable continuous cell reconfiguration to process and learn new features for deep learning.

Second, STT-MRAM has great potential to induce complex and tunable resistance dynamics through the mechanism of spin transfer torque [8]. Similar to PCM and RRAM, STT-MRAM can emulate synapses using intermediate resistance states of the MTJ. Additionally, the MTJ resistance can oscillate and spike, exhibiting complex dynamics to implement other important biological functions such as nanoscale neurons. In contrast, this is not readily possible in other NVM technologies where additional passive circuit elements such as capacitors or inductors are necessary to oscillate the resistance.

Furthermore, STT-MRAM is compatible with CMOS [30], and is now in a close-to-market position towards its commercialization. For example, high-density (256 MB) STT-MRAM storage has recently been demonstrated by Everspin [10]. The enhanced maturity of STT-MRAM will enable more practical experiments for neuromorphic computing.

**[Challenges.]** Despite the various benefits, STT-MRAM has been little exploited for accelerating deep learning. Instead, the state-of-the-art NVM accelerators have largely focused on using RRAM [27] [6] [31] [33]. This is primarily due to the challenges in maintaining NN model accuracy across the device and architecture levels. As illustrated in Section 2.2, NVM-based NN computation requires programming synaptic weights into NVM cell conductance (resistance). Since a NVM cell only has a limited number of resistance states, the synaptic weight that it represents has largely reduced data precision, which in turn lowers the NN model accuracy. This situation is exacerbated in STT-MRAM because of its device-level characteristics.

At the device level, STT-MRAM encounters a significant challenge that prevents it from being practically used for NN computation. The ON/OFF resistance ratio, defined as $R_{AP}/R_P$, is extremely low in STT-MRAM. Compared to a typical ON/OFF ratio of $10^{6}$ to $10^{8}$ in RRAM and PCM, the highest ON/OFF ratio ever reported for STT-MRAM was just about 7 [13]. This significantly limits the possible resistance range of intermediate resistance states in a cell. Our
adjusting (very small amount of weights are quantized to it). In other words, the above challenges across multiple design layers, enabling a full-stack solution for STT-MRAM-based deep learning acceleration for the first time. Section 4 conducts device-level experiments that demonstrate multiple (4, at this point) resistances states in a hardware STT-MRAM cell. By connecting two STT-MRAM cells in parallel to store a synaptic weight (Section 5.4), CELIA obtains sufficient (16) resistance states for the proposed architecture-level weight quantization scheme (Section 5) to achieve negligible model accuracy loss. Finally, CELIA provides a detailed system-level design (Section 6) showing significantly improved performance and energy efficiency over a state-of-the-art RRAM accelerator [33].

4 CREATING MULTIPLE RESISTANCE STATES IN A STT-MRAM CELL

4.1 Device-Level Solution

Our device-level work aims at creating multiple resistance states in a STT-MRAM cell. As mentioned in Section 3, this is particularly challenging in STT-MRAM due to its low ON/OFF resistance ratio. Our proposed solution is inspired by two observations: (1) the STT-MRAM cell’s anti-parallel (AP) state resistance is strongly dependent on the applied bias voltage; and (2) the AP state resistance also shows dependence on the voltage sweeping history. These two observations are demonstrated in Figure 6, which shows the STT-MRAM cell’s resistance-voltage (R-V) switching characteristics obtained from our device-level experiments.

We first conducted a bias voltage sweep from $-1V$ to $1V$ then back to $-1V$, generating a typical hysteretic R-V characteristic shown as the purple trend in the figure. As can be seen, $R_{AP}$ reaches its maximum value at the zero bias, and decreases as the bias voltage increases in either positive or negative polarity direction. Conventionally, this strong dependence on the bias voltage has long been considered as a challenge in retrieving the stored data, since only a low voltage near zero can be applied in order to maximize the read-out signal. In this work, however, we take advantage of this unique behavior to utilize the bias dependence to create enough room for intermediate resistance states. In contrast, the parallel (P) state resistance barely changes during the voltage sweep. The SET switching (AP-to-P state transition) occurs at around $0.5V$, whereas the RESET switching (P-to-AP state transition) occurs at around $-0.6V$. These state transition regions are too abrupt to create reliable intermediate resistance states. One prior study [24] created multiple resistance levels in the AP-to-P state transition region by controlling the domain-wall motion in the magnetic free layer;

**Figure 5**: An illustration of uniform data quantization.

**Figure 6**: The STT-MRAM cell resistance-voltage (R-V) characteristics obtained with three bias voltage sweeps.
however, due to the stochastic nature of the domain wall pinning, the fabricated magnetic memristor synapse exhibited large cycle-to-cycle and device-to-device variations that may impede its practical usage in neuromorphic computing. Hence, our solution will target the AP state region in the R-V characteristic.

The second observation suggests that it is possible to obtain different resistance values at the AP state by controlling the voltage sweep and its cycling history. We have conducted another two voltage sweeps that generate the black trend (for Sweep #2) and the blue trend (for Sweep #3) in Figure 6. The second sweep changes the voltage from $0.4V$ to $-0.2V$, resulting in a resistance value that is lower than that in the initial full sweep. On the contrary, the third sweep varies the voltage from $-0.2V$ to $0.4V$, and yields another resistance value higher than that in the first sweep. Therefore, by carefully controlling the bias voltage sweeping history (including the starting voltage and the sweeping direction), creating new resistance states in a STT-MRAM cell is possible. These new resistance states may be attributed to the increase or decrease in the number of spin-polarized electrons in the free layer, depending on the direction of the current flow.

### 4.2 Device-Level Experiments

Our device-level experiments were conducted on fully functional, nanoscale STT-MRAM devices fabricated through our industry collaborator. These STT-MRAM samples feature the cutting-edge perpendicular MTJ (p-MTJ) technology with a TMR ratio (i.e., $(R_{AP} - R_{P})/R_{P}$) of greater than 100% and a very low resistance-area (RA) product of about $100\mu m^2$. The p-MTJ stack was deposited on $300 mm$ wafers by an Applied Materials’ Endura sputtering system.

Figure 7 demonstrates four distinct resistance states obtained in our experiment with a STT-MRAM cell. They all belong to the AP state of the cell, and are each configured with a different bias voltage sweep. More importantly, these new states show non-uniform resistance values with, for instance, the “01” state being closer to the “00” state than the “10” state. These non-uniform intermediate resistance states, which are configured by controlling the voltage sweeping direction and the time delay between adjacent sweep operations, will naturally enable the non-uniform weight quantization scheme proposed in Section 5.

We are actively working on getting even more resistance states and investigating the precise physical mechanism of our device-level solution. It is important to note that the purpose of our device work is not directly comparing the number of resistances in STT-MRAM with other NVMs, but an important proof-of-concept for feasible STT-MRAM-based deep learning acceleration. The showcased multiple resistance states, along with the circuit-level enhancement of combining two STT-MRAM cells (Section 5.4), can already provide sufficient resistance states to the architecture-level weight quantization to achieve near-zero model accuracy loss. It is also important to note that, although the newly created resistances gather at the AP state, they can be easily scaled by a constant scaling factor at a higher design layer to fulfill the design needs of the whole system.

### 5 Non-Uniform Weight Quantization

With the intermediate resistance states provided by the device level (Section 4), this section seeks to better utilize them to represent the original NN model, so that the accuracy loss of the model inference can be minimized. As specified in Section 3, our proposed solution is motivated by the observation that the conventional, uniformly distributed quantization points are not equally important. Therefore, we propose identifying the most important quantization points in a non-uniform manner.

#### 5.1 Ineffectiveness of Existing Works

Uniform quantization schemes, including the static and dynamic approaches, have been used in state-of-the-art NN accelerator designs. The static uniform quantization [3] [4] directly relies on the fixed-point number representation (Section 3), using fixed integer and fractional lengths for the whole model; therefore, all synaptic weights in the NN model have the same data precision. The dynamic uniform quantization [26] [6] allows tuning the fractional length ($f_l$) across different CNN layers to minimize the difference between the original and quantized weight values. Regardless, these two approaches both use uniform quantization points that cannot capture the highly complex nature of modern CNN models.

A few existing works also used non-uniform quantization. LogNet [23] assigns quantization points based on a logarithmic distribution: the first point is at 1/2 of the maximum weight value, and each addition point is at the mid-point between zero and the previous point. SPINDLE [27] uses a very similar quantization function $\tanh()$ that also assigns more points towards zero. Although being non-uniform, the log and other similar quantization schemes still suffer from low accuracy in quantizing CNNs due to several reasons. First, they rely on a fixed quantization function, such as $\log()$ and $\tanh()$, thus being unaware of the varying weight distribution of different CNN models. In other words, the same set of quantization points are used independently of the application. Second, even with increased data precision, they simply assign the additional quantization points towards zero. As we have already articulated in Section 3, a quantization point that is close to zero has limited impact on the model accuracy. We will show in Section 5.3 that these existing uniform and non-uniform quantization schemes result in significant accuracy loss with a largely reduced bit width of synaptic weights.

---

---

Figure 7: The four resistance states obtained at the AP state using different voltage sweeps.
Moreover, Deep Compression [9] provides a learned approach to group synaptic weights and assign quantization points learned from data. However, it involves additional, expensive model re-training in order to identify these points. It also requires a lookup table to encode the learned full-length quantization values. In contrast, our solution proposed next also learns from data, but does not need additional model training nor any extra model structure.

5.2 Proposed Solution

In our non-uniform quantization scheme, we construct an importance function $g(x) \cdot |x|$, where $g(x)$ is the weight distribution of the NN model (as exemplified in Figure 5), to approximate the importance of different quantization values to the model accuracy. This is shown as the blue solid curve in Figure 8. This bimodal function curve is in accordance with our previous intuition that quantization points that are too close to or too far from zero are less important. The proposed importance function takes into account both the weight value and the weight amount of the model, indicating that the most important quantization points are around the peaks of the function curve.

To identify the most important quantization points, we evenly partition the area between the function curve and the x-axis into $2^{bw} + 1$ regions. As exemplified in Figure 8, for a bit width of 3, the entire area is partitioned into 9 regions. For the central region, its quantization point is forced to be zero; for each of the other regions, the quantization point is the x-axis value that divides the region into halves (analogous to the center of mass for this region). We set a quantization point at zero because: (1). a great amount of weights are close to zero, and quantizing them to other points will result in too high residuals; (2). quantizing at zero can enable future optimizations that make use of the model sparsity; and (3). extreme quantization such as ternary networks [25] [41] also keeps zero as a critical point.

Furthermore, we generalize the importance function to $g(x) \cdot |x|^k$, so that the weight value can be prioritized differently by adjusting $k$. Figure 9 illustrates the function curves for the cases of $k = 1$, $k < 1$, and $k > 1$. In our model, we test different values of $k$ in the range of [0, 2] with a stride of 0.1, and pick the one that gives the highest model accuracy.

5.3 Accuracy Experiments

We have implemented the proposed non-uniform quantization scheme, and compared it against a number of existing schemes using six large CNN models from Caffe [14]. Detailed experimental setup and benchmark descriptions can be found in Section 6.4. Figure 10 shows how the model inference accuracy varies with the reduced bit width of synaptic weights. "static" and "dynamic" are the static [3] [4] and dynamic [26] [6] approaches using uniform quantization; "log" is the log quantization scheme [23]; "k = 1" and "optimal k" are our proposed non-uniform quantization schemes. The sign bit is excluded from the bit width shown on the x-axis, because separate crossbar arrays will be used for positive and negative operations in our design (see Section 6.3). In other words, the amount of crossbars used for NN computation is doubled so that
the sign bit can be saved in the weight representation. In all these results, the input data is uniformly quantized to 5 bits, because: (1) our digital-to-analog converter (DAC) specified in Section 6.2 requires uniformly quantizing the input neurons; and (2) reducing the bit width of input data to fewer than 5 bits results in unacceptable accuracy loss.

As can be seen, the static and dynamic uniform quantization schemes demonstrate significant accuracy loss starting from 7 bits. The log quantization shows little accuracy variation from 7 bits to 3 bits. This is expected since when more quantization points are available it simply assigns them towards zero. Nevertheless, the log quantization still shows significant accuracy loss in Cifar10, AlexNet, and CaffeNet; although it is comparable to our scheme in the other three workloads, note that LeNet is a relatively small network where all evaluated schemes perform well. Finally, our proposed scheme with the optimal k demonstrates negligible accuracy loss for a bit width of as low as 4 bits, consistently in all the evaluated workloads.

### 5.4 Device and Architecture Co-Design

Our accuracy experiments in Section 5.3 reveal that the proposed non-uniform quantization scheme can achieve near-zero accuracy loss in state-of-the-art CNN models as long as the bit width of synaptic weights is at least 4 bits. This indicates the need of 16 resistance states in a single cell. However, the device-level work (Section 4) can only provide four resistance states in a STT-MRAM cell at this point. In order to bridge the gap between the device and architecture solutions, we further propose a circuit-level enhancement that connects two STT-MRAM cells in parallel to implement an accelerator cell that stores a synaptic weight.

This is depicted in Figure 11, where the two STT-MRAM cells in connection have resistances of $R_1$ and $R_2$, respectively. Therefore, the resulting accelerator cell has a resistance of $(R_1 \times R_2)/(R_1 + R_2)$. If $R_1$ is configured to have four non-uniform resistance states and $R_2$ is configured to have a different set of four non-uniform resistance states, the accelerator cell will have 16 resistance states that are non-uniform as well.

Prior studies also composed multiple NVM cells to represent a synaptic weight. For example, PRIME [6] used two 4-bit RRAM cells to represent one 8-bit synaptic weight. However, PRIME’s composing scheme is in the digital domain, requiring complicated digital circuits. In contrast, our accelerator cell design relies on analog currents being composed for computation, and therefore only needs one half of analog-to-digital converters (ADCs) and many simplified inter-cell circuits compared to PRIME.

Our proposed design also enables easy reconfiguration of synaptic weights stored in accelerator cells. To configure $R_1$ and $R_2$ in the two STT-MRAM cells, the switch in Figure 11 is closed first and $R_1$ is configured in both cells; after that, the switch is open and $R_2$ is configured in the cell on the right. Configuring a resistance state in a STT-MRAM cell can be done via performing a voltage sweep with certain conditions, as described in Section 4.

### 6 A DEEP LEARNING ACCELERATOR SYSTEM

In addition to the device and architecture innovations, we also include a detailed accelerator system design in CELIA to achieve a full-stack solution for STT-MRAM-based deep learning acceleration. As shown in Figure 12 (a), the accelerator system is composed of a number of compute units connected to on-chip buffers and global buses. A compute unit (CU) has a pipelined architecture given in Figure 12 (b). This overall system architecture is similar to recent RRAM accelerators PRIME [6] and PipeLayer [33], which organize computational crossbar arrays in a hierarchy of banks/chips/ranks similar to DRAM. This section uses PipeLayer as the baseline, and introduces several enhancements. Note that the proposed accelerator design is generic for different NVMs. This is the same in existing accelerators [31] [6] [33] whose components are also independent of NVMs.

In the proposed system, original input images are first fetched from off-chip memory, quantized uniformly to a bit width of 5 bits (see Section 5.3), and stored in on-chip buffers for future reuse of intermediate results. Before input data enter the CU pipeline, synaptic weights have already been programmed into the crossbar array using our proposed non-uniform quantization with a bid width of 4 bits (Sections 5). The entire CU pipeline is in the analog domain with digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) as the entry and exit points, respectively. The detailed designs of different pipeline stages will be described in the following subsections. The buffers are logically shared across CUs, but can be physically distributed to each CU in the hardware design. The CUs are allocated to different layers in a CNN model based on the computation need of each layer and the total available CUs in the system.

#### 6.1 A Crossbar Allocation Scheme with Input Reuse

Before introducing our new crossbar allocation scheme, we need to take a close look at how a convolution operation is performed in
the crossbar array. Background on convolution can be referred to
in Section 2.1. Figure 13 (a) illustrates such a convolution operation:
windows of data, each being from an input feature map, are flattened
and concatenated to form an input vector; the same number of
convolution kernels, as colored in Figure 13(a), are also flattened
and concatenated to form a weight vector; calculating the dot-
product of these two vectors gives one pixel value in an output
feature map. In a crossbar array, as shown in Figure 13(b), the input
vector is applied horizontally as wordline voltages; the weight
vector is programmed to one vertical bitline of cells (colored as
“1” and the “0”), thus largely reducing the design complexity and
energy consumption of the DAC.

In addition, we incorporate a bit flipping scheme in the DAC: in
a column of input bits, the DAC flips all the bits if there are more
1s than 0s (assuming that the high voltage is used to represent “1”).
This scheme further reduces the energy consumption of the DAC
at the expense of one extra bit per input bit column to indicate
flipping.

6.2 An Energy Efficient DAC Design
Digital-to-analog converters (DACs) are responsible for transform-
ing the digital data fetched from buffers into analog input voltages
to the crossbar array. Similar to ISAAC [31], we propose a DAC
design taking one bit of information at a time. This is illustrated
in Figure 15, where the inputs are represented as 5-bit binaries. In
each cycle, only one bit column (with each bit from one input) is
converted by the DAC to analog voltages. The shift-and-add logic
(described in Section 6.3) will aggregate multiple bit columns in
the analog domain. This binary streaming approach only needs a
simple inverter to generate high and low voltages (representing the
“1” and the “0”), thus largely reducing the design complexity and
energy consumption of the DAC.

We term such a weight matrix a “pixel weight matrix (PWM),
because applying one input vector to this weight matrix calculates
the pixels at the same location across all output feature maps. As
shown in Figure 13(b), a PWM has a size of $n_{in} \times k \times k \times n_{out}$,
where $n_{in}$ and $n_{out}$ are the numbers of input/output feature maps
in the current layer, and $k \times k$ is the kernel size. Note that a PWM
contains all the synaptic weights of the current layer. When the
input data windows slide to the next location, a new input vector is
formed and applied to the PWM to calculate the next set of output
pixels. Consequently, a NVM accelerator can choose to apply input
vectors sequentially to the same PWM, or replicate multiple PWMs
to improve data processing parallelism [33].

A naive crossbar allocation scheme, as shown in Figure 14(b),
simply replicates PWMs without any overlaps in both dimensions.
This is to avoid interference across inputs/outputs. In the shown
example, we assume $n_{in} = 1$, $n_{out} = 5$, and $k = 3$, and the resulting
PWM is $9 \times 5$. As shown in Figure 14(b), a 30 $\times$ 30 crossbar array can
only accommodate three such PWMs, and only half of the wordlines
generate effective outputs. Figure 14(a) shows the corresponding
input sliding windows of data (with pixels numbered from 1 to 15)
for the three PWMs. Since a convolution operation usually slides
the input window with a stride of 1, we easily observe significant
overlap across nearby input windows.

Consequently, we propose a new crossbar allocation scheme
with input reuse. As illustrated in Figure 14(c), three PWMs reuse
the input data that they share. As a result, another three PWMs can
be accommodated in the crossbar array, and all the wordlines now
have useful outputs. This effectively doubles the data processing
parallelism (performance). Our scheme examines reusing the input
across different numbers of PWMs, and chooses the allocation that
maximizes the effective output ratio in the crossbar array.

6.3 Other Components

[Accumulation Logic]. The accumulation logic is responsible for
the following tasks. First, it uses an amplifier to implement I-to-V
logic to generate the result voltage based on the result current. In
this process, it checks whether the input bits have been flipped
in the DAC, and outputs the correct result voltages accordingly.
Second, it accumulates the result voltages from multiple input bit
columns, and outputs the results for the complete inputs. An circuit
implementation of these two tasks is given in Figure 16. The shift-
and-add logic is needed to assemble the outputs due to the DAC
design (Figure 15) taking input bit columns in a sequential manner.
Third, the accumulation logic handles the signed arithmetic, by
using separate crossbar arrays for positive and negative synaptic
weights [6]. As shown in Figure 17, it relies on a subtraction circuit
to calculate the difference between the result voltages from the
two crossbars. Finally, it performs ReLU activation ($y = \max(0, x)$),
by outputting a zero voltage for a negative result and keeping the
original value for a positive result.

[Crossbar-Based Pooling]. To perform max pooling, we follow
the same design proposed in PRIME [6]. As shown in Figure 18,
it relies on the separate positive/negative crossbars (similar to han-
dling signed arithmetic) to calculate the difference between each
pair of inputs. These results are then used as a control signal to a
multiplexer to determine the largest input.

[ADC]. Similar to PipeLayer [33], we also use the integrate-and-
dump design as our analog-to-digital converter (ADC). It is important
to note that the ADC in our accelerator system is the exit point

Figure 13: (a): A convolution example. (b): The correspond-
ing “pixel weight matrix”. (c): A Crossbar Pooling Example.
Figure 14: (a): Sliding windows of input data used in convolution. (b): Naive crossbar allocation. (c): The proposed crossbar allocation with input reuse.

Figure 15: The proposed DAC design.

Figure 16: The shift-and-add logic.

Figure 17: Signed arithmetic and ReLU activation.

Figure 18: The pooling logic.

of the computer unit pipeline. Hence, all the operations occurring in the crossbar array, accumulation logic, and pooling logic are in the analog domain. This reduces the amount and complexity of the ADCs needed in the system.

Buffers. To be compatible with the crossbar arrays, the on-chip buffers in the system are implemented with STT-MRAM. STT-MRAM has long been used in implementing low-level caches [36] and the main memory [20] [40]. In particular, a recent study [34] specifically utilizes STT-MRAM as a suitable storage device for energy-efficient neural network accelerators. Furthermore, we observe that the data usage in these buffers has low locality with short data reuse duration (i.e., data is read only once after it is written to the buffer). This is due to the fact that the results generated by one layer in a CNN model will only be used as the inputs to the next layer. Therefore, we can reduce the data retention time of the STT-MRAM buffer in exchange of improved energy efficiency [32] [37] [15] [40]. We leave this enhancement as our future work.

6.4 System-Level Experiments

Experimental Setup. In our system-level experiments, we evaluate the performance and energy consumption of CELIA, and compare it against PipeLayer [33]. PipeLayer is used as the baseline because it is the latest NVM-based NN accelerator design as of now. Six workloads from Caffe [14] are used in our experiments: LeNet, CIFAR-10, AlexNet, CaffeNet, VGG16, and VGG19. The first two were trained in-house, while the rest were pre-trained models available in Caffe Model Zoo [1]. Table 1 lists the numbers of layers in each workload (note that VGG16 and VGG19 only count CONV and FC layers in their names).

To evaluate performance, we implement a cycle-accurate performance simulator based upon DRAMSim2 [28] and NVSim [7]. For a fair comparison with PipeLayer, we assume that both systems have the same amount of NVM cells. To evaluate energy consumption,
Table 1: The numbers of layers in used CNN models.

<table>
<thead>
<tr>
<th>CNN Model</th>
<th>CONV</th>
<th>Pooling</th>
<th>FC</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>AlexNet</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>CaffeNet</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>VGG16</td>
<td>13</td>
<td>5</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>VGG19</td>
<td>16</td>
<td>5</td>
<td>3</td>
<td>24</td>
</tr>
</tbody>
</table>

Figure 19: Performance (speedup) comparison.

Figure 20: Speedup (due to crossbar allocation only) sensitivity on the crossbar array size.

we use unit energy data from ISAAC [31], PipeLayer [33], and also running NVSim. We will report the energy consumption of various components in CELIA, including crossbar arrays, DACs, ADCs, buffers, etc.

[Performance]. Figure 19 shows the performance comparison between PipeLayer and CELIA. Speedups over PipeLayer (higher is better) are shown. CELIA consistently outperforms PipeLayer in all workloads evaluated, achieving an overall speedup of 3.28x than PipeLayer. The performance is improved primarily due to two reasons. First, our proposed crossbar allocation scheme (Section 6.1) more effectively utilizes the crossbar arrays, especially in CONV layers with relatively small pixel weight matrices (e.g., in CIFAR-10). This is shown as a separate bar in Figure 19, which varies across benchmarks with an average speedup of 1.64x. Second, PipeLayer needs four 4-bit RRAM cells to represent a 16-bit synaptic weight, whereas a synaptic weight in CELIA only needs two STT-MRAM cells. As a consequence, the effective computation capacity is doubled in CELIA, resulting in a consistent speedup of 2 (see Figure 19).

Figure 20 further lists the sensitivity of the speedup (due to crossbar allocation only) of CELIA over PipeLayer on different crossbar array sizes, with a fixed total amount of crossbars. As can be seen, the speedup increases with a larger crossbar size. This is because a larger crossbar array can enable more flexibility of using our proposed crossbar allocation scheme. The default crossbar array size is 256 × 256 in both PipeLayer and CELIA.

[Energy Consumption]. Figure 21 compares the energy consumption of PipeLayer and CELIA, showing a breakdown by different components. Overall, CELIA’s energy consumption is only 16% of that of PipeLayer, achieving an energy efficiency improvement of 6.25x. As can be seen, the major reductions are in DACs, buffers, crossbars, and ADCs. The energy reduction in DACs is due to: (1) our DACs with only two voltages (much reduced power); (2) the proposed bit flipping scheme; and (3), the improved performance (reduced running time). The energy reduction in buffers is because PipeLayer stores 16 bits for an input neuron whereas CELIA only stores 5 bits. The energy reduction in crossbars is because STT-MRAM cells have much lower resistances than RRAM cells. The energy reduction in ADCs is because: (1). CELIA uses half amount of ADCs due to combining two STT-MRAM cells in the analog domain; and (2). the performance improvement.

7 CONCLUSIONS
This paper proposes: (1). a device-level optimization for STT-MRAM to generate intermediate resistances in a cell; (2). a non-uniform data quantization scheme to minimize the model accuracy loss due to reduced data precision; and (3). a STT-MRAM deep learning accelerator with detailed component designs. The proposed framework is the first deep learning accelerator using STT-MRAM-based crossbar arrays, and also exhibits a full-system solution that bridges the gap between the STT-MRAM technology (as a promising synaptic element) and a magnetic neuromorphic computer.

ACKNOWLEDGMENTS
This work is supported in part by National Science Foundation (NSF) under Grant CCF-1566158. The authors would like to thank the anonymous reviewers and the shepherd for providing invaluable feedback for this work.
REFERENCES