3DICT: A Reliable and QoS Capable Mobile Process-In-Memory Architecture for Lookup-based CNNs in 3D XPoint ReRAMs

Qian Lou
Indiana University Bloomington
louqian@iu.edu

Wujie Wen*
Florida International University
wwen@fiu.edu

Lei Jiang
Indiana University Bloomington
jiang60@iu.edu

ABSTRACT

It is extremely challenging to deploy computing-intensive convolutional neural networks (CNNs) with rich parameters in mobile devices because of their limited computing resources and low power budgets. Although prior works build fast and energy-efficient CNN accelerators by greatly sacrificing test accuracy, mobile devices have to guarantee high CNN test accuracy for critical applications, e.g., unlocking phones by face recognitions. In this paper, we propose a 3D XPoint ReRAM-based process-in-memory architecture, 3DICT, to provide various test accuracies to applications with different priorities by lookup-based CNN tests that dynamically exploit the trade-off between test accuracy and latency. Compared to the state-of-the-art accelerators, on average, 3DICT improves the CNN test performance per Watt by 13% ~ 61× and guarantees 9-year endurance under various CNN test accuracy requirements.

CCS CONCEPTS

• **Hardware → Non-volatile memory**; **Hardware accelerators**;

KEYWORDS

3D vertical ReRAM, Lookup-based CNN, process-in-memory

ACM Reference Format:
Qian Lou, Wujie Wen, and Lei Jiang. 2018. 3DICT: A Reliable and QoS Capable Mobile Process-In-Memory Architecture for Lookup-based CNNs in 3D XPoint ReRAMs. In *IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD '18), November 5–8, 2018, San Diego, CA, USA.* ACM, New York, NY, USA, 8pages.
https://doi.org/10.1145/3240765.3240767

1 INTRODUCTION

Deep CNNs emerge as one of the most effective solutions to a wide range of problems, e.g., object recognition [19] and machine translation. Deep CNN models trained by huge datasets such as ImageNet are extremely relevant and critical to intelligent mobile devices such as virtual reality equipments, smart surveillance systems and self-driving cars, since they have demonstrated high test accuracy. However, CNN tests involve both huge volumes of computationally intensive convolutions and large amounts of memory intensive weights. For example, even for a CNN created in 2012, AlexNet [19], a test requires 724M floating point (FP) multiply-accumulate (MAC) operations and 61 million parameters. The essential computing effort of CNN tests prevents CPUs and GPUs from achieving a high enough frame rate on mobile devices with tight power budgets [14].

Although binary CNNs [9, 25, 26, 30] reduce weight and convolution computing overhead by binarizing weights and inputs into 1s and -1s, they work well with only small databases, i.e., MNIST and CIFAR-10, and degrade deep CNN accuracies by 22%~26% when testing ImageNet. In near future, multiple workloads on a mobile device will initiate multiple CNN tests on sensor streams such as video, audio, location tracking and thermal video [14]. Delivering highly accurate CNN tests for all simultaneous applications will exhaust available hardware and power resources of mobile devices. Therefore, a mobile CNN accelerator *must* allow a graceful trade-off between test accuracy, latency and energy consumption. It has to guarantee highly accurate CNN tests for a small portion of critical tasks, e.g., secure logins by face recognitions, but adapt to a large number of real-time tasks using less accurate CNN tests with lower computing overhead. However, no existing mobile accelerator can exploit such trade-off to dynamically control CNN test quality-of-service (QoS).

Recently, Lookup-based CNN (LCNN) [2] is presented to reduce the test convolution overhead by encoding convolutions through few lookups to a small dictionary. By using different sizes of dictionaries, a LCNN test costs different amounts of computing overhead and achieves various accuracies. Moreover, recent hardware works [5, 13, 16, 28] propose a ReRAM-based Dot-product Engine (RDE) to improve the convolution computing efficiency by $> 10^3\times$ over traditional ASIC designs. A RDE efficiently executes convolutions by analog signals and produces output digital results by analog-to-digital converters (ADCs). However, it is *challenging* to deploy LCNNs onto RDEs particularly in low power mobile devices. Because of the tight mobile power budget, a mobile accelerator can have only a limited number of power hungry ADCs and short endurance multi-level cell (MLC) ReRAM arrays. Since it is impossible to spread all dictionary entries of a LCNN into such few ReRAM arrays, frequent writes updating dictionary entries in these arrays are inevitable. As a result, the naïve mobile LCNN accelerator design cannot obtain reasonable test performance. Moreover, with

*Dr. Wen is partially supported by NSF CNS-1840813.

ICCAD '18, November 5–8, 2018, San Diego, CA, USA

ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3240767

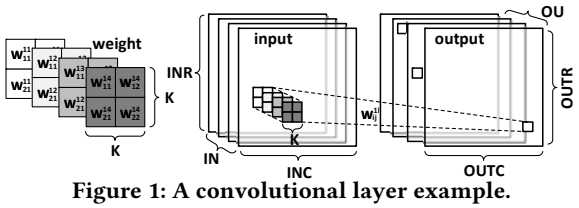


Figure 1: A convolutional layer example.

non-stop LCNN tests, all RDEs in the accelerator may be worn-out within 5 days. In this paper, we propose a reliable and energy-efficient 3D XPoint ReRAM-based Process-In-Memory (PIM) architecture, 3DICT, to deliver various test accuracies to different applications based on their QoS requirements. Our contributions are summarized as follows:

- We create a QoS-capable mobile PIM accelerator, 3DICT, to exploit the trade-off between CNN test accuracy and latency.
- We propose a 3D XPoint single level cell RDE to prolong the endurance of 3DICT with trivial hardware and power overhead. We further present two architectural optimizations including hybrid convolution and lookup-discard-scale to further accelerate LCNN tests.
- We evaluate and compare 3DICT against the state-of-the-art accelerators. Our experimental results show 3DICT improves the CNN test performance per Watt by 13% ~ 61× with various accuracy requirements over its counterparts.

```
//normal convolution                          INR // input row #
for(row=0; row<OUTR; row++)                                          modifications for
    for(col=0; col<OUTC; col++)                                      small convolution
                                                                     DL // a smaller value
      for(outn=0; outn<OU; outn++){
          for(inn=0; inn<IN; inn++)
// input col #                 for(i=0; i<K; i++)                    // small convolution
    INC                           for(j=0; j<K;j++)                 semiD[outn][row][col]+=
                                      output[outn][row][col]+=          dict[outn][inn]*
                                      weight[outn][inn][i][j]*          input[inn][row][col];
                                      input[inn][SW*row+i][SW*col+j];
      }
```

Figure 2: The pseudo code of a (small) convolution.

2 BACKGROUND

2.1 Convolutional Neutral Network

The state-of-the-art CNNs [15, 19] are supervised learning algorithms invoking a feedforward function during *tests* and a back-propagation process for *trainings*. In mobile devices, image or object recognitions are performed in real-time, and hence the CNN test speed matters. In contrast, we can train CNNs through multiple powerful GPUs in the cloud. In this paper, we focus on accelerating CNN tests in mobile devices. A typical CNN consists of multiple types of layers including convolutional, activation, pooling and fully-connected layers [18]. A convolutional layer example is exhibited in Figure 1. The layer receives IN input channels, each of which has INC columns and INR rows. Each input channel is convolved by a shifting window with a $K \times K$ weight filter to generate an element in one $OUTC \times OUTR$ output channel. The stride of the shifting window is SW ($< K$). Totally, OU output channels are generated by the convolutional layer. The pseudo code of a convolutional layer can be seen in Figure 2. The convolutional layers in AlexNet cost > 90% of the test execution time [35].

2.2 ReRAM-based dot-product engine

2D RDE. Recent works [5, 13, 16, 28] propose a ReRAM-based Dot-product Engine (RDE) to improve the performance per Watt of dot-products by > 10^3× over traditional ASIC designs. A RDE example

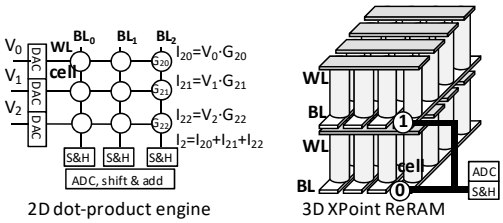


Figure 3: ReRAM-based dot-product engines.

is shown in Figure 3, where the array consists of wordlines (WLs), bitlines (BLs) and cells. Each cell on a BL is programmed into a certain resistance (R), e.g., $cell_{2x}$ on BL_2 is written to R_{2x}, where $x = 0, 1, 2$. The cell conductance (G) is the inverse of the cell resistance ($1/R$), e.g., $cell_{2x}$ has a conductance of $G_{2x} = 1/R_{2x}$. A voltage (V_x) can be applied to each WL, so that the current (e.g., I_{2x}) passing through a cell ($cell_{2x}$) to the BL is the product of the voltage and the cell conductance ($V_x \cdot G_{2x}$). Based on the Kirchhoff's law, the total current (e.g., I_2) on a BL (BL_2) is the sum of currents passing through each cell on the BL (BL_2), so $I_2 = \sum_{x=0}^{2}(V_x \cdot G_{2x})$. All BLs in the array produce the current sums simultaneously with the same voltage inputs along WLs. In this way, a vector-matrix multiplication between the input vector V and the conductance matrix G stored in the array is computed by a RDE each step. The conversion between analog and digital signals is necessary for RDEs to communicate with other digital circuits. A digital-analog converter (DAC) unit converts digital inputs into corresponding voltages that are applied to each WL. A sample-and-hold (S&H) circuit captures the bitline current, converts the current to a voltage and sends the voltage to an expensive ADC unit. The ADC costs 66.4% of power consumption and 73.2% of area overhead in a RDE [28]. To accelerate CNN tests, prior works [5, 28] encode CNN weights as cell conductances in a RDE and apply input voltages to all BLs, so the BL currents can represent the elements in an output channel, where the same inputs convolve with different weight filters.

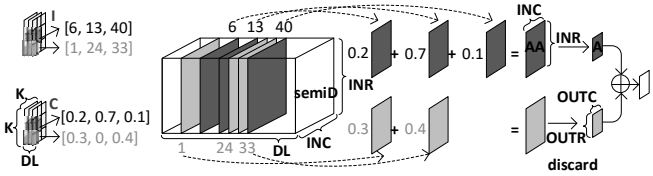


Figure 4: Lookup-scale-discard operations.

3D RDE. A RDE favors a long BL, since more ($V \cdot G$)s can be summed along one BL. However, a long BL introduces significant parasitic resistance degrading the voltage integrity [13]. Recent works [16, 28] connect only 128 cells to a BL to mitigate the parasitic resistance in a RDE. To increase the dot-product throughput, as Figure 3 shows, a 3D XPoint ReRAM-based dot-product engine [3] integrates multiple 2D RDEs into a 3D XPoint structure, where each layer has an independent copy of BLs and WLs. Although every S&H circuit accumulates currents on multiple BLs, each of which is from one layer, to improve dot-product throughput, the 3D RDE requires a ADC with larger resolution to prevent overflows when converting the current to a digital value. However, increasing ADC resolution significantly boosts its power consumption and area overhead. For instance, compared to an 8-bit ADC, a 10-bit counterpart [24] increases the area by 6× and the power consumption by 10×. So existing 3D RDEs actually degrade the dot-product performance per Watt per mm^2.

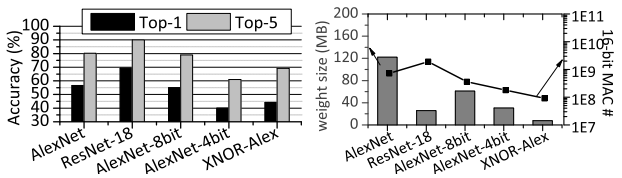
2.3 Lookup-based CNN

Basics. Lookup-based Convolution Neural Network [2] (LCNN) reduces the convolution overhead by encoding convolutions by few lookups to a dictionary trained to cover the space of weights in CNNs. LCNN has the same pooling and activation layers as full precision CNNs. But for each *convolutional* or *fully-connected* layer in LCNN, all weight filters are concentrated to three components: a shared dictionary (*dict*), an index tensor (I) and a coefficient tensor (C). The *dict* of a LCNN layer consists of DL row vectors of length IN, where DL is the dictionary length. The size of the dictionary (DL) varies for different LCNN layers, but it is always smaller than the total number of vectors in weight filters of a CNN layer ($OU \times K \times K$, where OU is the CNN output channel number and K is the CNN weight kernel width). Both I and C consist of DL channels, each of which has $K \times K$ elements. The computation of a LCNN convolutional layer consists of two steps: *small convolution* and *lookup-scale-discard*. Figure 2 exhibits the small convolution pseudo code. In a small convolution, the input is first convolved with the dictionary *dict*. *Different* from the CNN convolution, the small convolution output *semiD* comprises DL channels of size $INR \times INC$, where INR and INC are the width and height of a input channel, and DL indicates the dictionary size. Lookup-scale-discard operations are described in Figure 4. A lookup-scale-discard fetches channels of the small convolution output *semiD* based on its corresponding entry in I, e.g., [6, 13, 40]. Then, it scales the lookup results by their factors in C, i.e., [0.2, 0.7, 0.1], and accumulates all scaled lookup results into an intermediate matrix AA. Based on its corresponding entry in I, some elements in the intermediate $INR \times INC$ matrix AA are discarded, so that a smaller result matrix A with $OUTR \times OUTC$ is produced. At last, the element-wise sum of all result matrices is the output of the LCNN layer. More details can be found in [2].

Advantages. By adjusting the dictionary size (DL), a LCNN endows mobile devices the capability of dynamically tuning the test latency and accuracy for multiple applications with different QoS requirements. Moreover, LCNN also reduces the CNN training effort in cloud servers. First, the dictionary allows a network to learn from very few training examples on novel categories [2]. The *few-shot learning* ability is extremely important to mobile devices that need to adapt to frequently changed user preferences. When a user needs to classify a new set of categories, instead of reinitializing all layers randomly, LCNN can inherit dictionaries of a previously trained model and only learn the index tensor (I) and the coefficient tensor (C) from scratch by a few examples. Second, LCNN accelerates the trainings by limiting the number of learnable parameters without changing the network architecture [2]. Training deep neural networks is computationally expensive and requires hundreds of thousands of iterations. However, LCNN can create a network by training dictionaries on a shallow CNN and reusing it in the deeper CNN. For the deeper CNN, LCNN needs to train only the index tensor (I) and the coefficient tensor (C).

3 PRIOR ART AND MOTIVATION

The QoS support is critical for mobile devices to improve the overall performance and energy efficiency of multiple concurrent applications performing CNN tests. But state-of-the-art mobile CNN accelerators [1, 4, 12, 18, 23, 28] cannot gracefully exploit the trade-off between CNN test accuracy, latency and energy consumption.



(a) CNN accuracy. (b) CNN overhead.

Figure 5: Comparison between various CNN models.

The first successful deep CNN, AlexNet, has attained 56.6% top-1 and 80.2% top-5 accuracies shown in Figure 5(a) when testing images from the ImageNet dataset by 16-bit MAC operations. However, as Figure 5(b) exhibits, one AlexNet test requires 122MB weights and 724M MAC operations. Such huge computing overhead of AlexNet makes state-of-the-art mobile accelerators can barely support multiple workloads with the minimal real-time requirement, 30 FPS. The ASIC-based mobile accelerator, Eyeriss [4], can achieve only 13 FPS when processing full precision AlexNet tests. Although deeper CNNs such as VGG-16 and ResNet-18 increase the test accuracy, it is more difficult for mobile accelerators to execute these models, since they requires either more memory-intensive weights or computing-intensive MAC operations. For example, compared to AlexNet, ResNet-18 increases the number of MAC operations by 1.6×. Therefore, storing and switching between multiple full precision CNN models online is not an attractive way to exploit the trade-off between test accuracy, latency and energy consumption on mobile devices, because of the tight power budget.

A recent mobile accelerator, ENVISION [23], can compute one 16-bit MAC, two 8-bit MACs, or four 4-bit MACs in a MAC unit simultaneously. The computing overhead of two 8-bit MACs or four 4-bit MACs is equivalent to that of a 16-bit MAC. Although decreasing the bit-resolution in a MAC unit increases MAC computing throughput, it degrades the CNN test accuracy a lot. In Figure 5(b), the MAC (in terms of 16-bit MAC #) and weight overheads of AlexNet with 4-bit weights, inputs and activations are only 25% of those of full precision 16-bit AlexNet. As Figure 5(a) shows, compared to original AlexNet, the top-1 test accuracy of AlexNet-4bit is reduced by 30%. Prior accelerators, YodaNN [1] and XNOR-POP [18], use simplified CNN models (e.g., BinaryConnect [9] and XNOR-Net [26]) binarizing weights, inputs and activations to boost CNN test speed by performing only low hardware overhead operations such as additions, subtractions, XNORs and popcounts. Although the computing effort (in terms of 16-bit MAC # in Figure 5(b)) and the weights are greatly reduced by the simplified model XNOR-Alex, the test accuracy is also significantly reduced by 21% for top-1 and 13.7% for top-5 (Figure 5(a)). For critical applications such as unlocking phones by face recognitions on mobile devices, such large accuracy degradation is not acceptable. Moreover, even combining full precision AlexNet and XNOR-Alex models still cannot enable QoS in a mobile device, since only extremely high and low test accuracies can be expected.

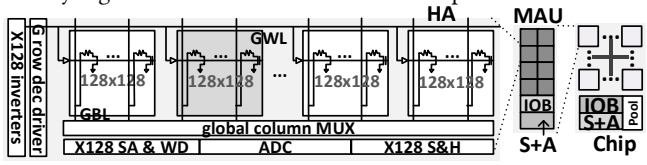


Figure 6: A hierarchical array architecture.

4 3DICT

In this section, we first formulate a mobile baseline configuration with the existing RDE design. Then, we illustrate the naïve implementation of LCNNs on our mobile baseline and identify the short endurance reliability problem of the existing RDE design. To prolong the endurance of our mobile LCNN accelerator, we propose a 3D XPoint RDE that uses SLC cells and 2-layer shared BLs to conduct dot-product operations. We further present two architectural optimizations, hybrid convolution and lookup-discard-scale, to accelerate the LCNN test performance of our 3D XPoint ReRAM-based PIM accelerator.

4.1 Constructing Baseline Configuration

Because of the limited battery lifetime of mobile devices, in this paper, we adopt $\sim$ $400mW$ peak power budget and an existing RDE design [28] to construct our mobile baseline. With such small peak power budget, there can be only 128 8-bit 1.28GSps ADCs in our mobile baseline. As Figure 6 shows, we use the hierarchical array (HA) structure [31] to share an ADC, 128 1-bit DACs (inverters), and 128 S&H circuitries among 8 2-layer 128×128 2-bit MLC ReRAM 3D XPoint arrays by global bitlines and wordlines. To correctly conduct dot-products, only 1 out of 8 arrays in a HA can be activated and use one shared ADC, 128 DACs and 128 S&H circuitries. Instead of a 2D array, our baseline adopts 2-layer 3D XPoint arrays to increase the array capacity but maintain the same dot-product throughput that is decided by the BL length and ADC for each array. During each dot-product, we treat each 3D XPoint array as a 2D array by using only one layer. The prior study [28] has already proved that a 128×128 2-bit MLC array balances the trade-off between the dot-product accuracy, power consumption and ADC/DAC overhead. Our baseline also adopts the 10MHz RDE pipeline [28], where each multiplication between a 16-bit 128-entry vector and a 16-bit 128×128 matrix costs 22 cycles, i.e., 16-cycle for RDE do-products (1-bit input per cycle), 1-cycle for ADC, 1-cycle for activation and 4-cycle for I/O operations. To form a multiply-accumulate unit (MAU), 8 HAs share a shift-add (S+A) unit and a ReRAM-based I/O buffer (IOB). Our baseline has only 16 MAUs connected by a 128-bit bus. Furthermore, the small peak power budget also limits the write bandwidth, i.e., only 16 128-bit writes can concurrently proceed. We also have a 64MB ReRAM read-only array to store various sizes of dictionaries for entire CNNs to provide different test accuracies. Each high density sub-array in the 64MB array is 1MB, so we cannot use these sub-array to perform dot-products. Otherwise, the summed current along BLs will be destroyed by the large parasitic resistance of these long BLs. Because of the non-volatility of ReRAM, array peripheral power gating [36] is applied to eliminate the 64MB read-only array power consumption during its idle period. The detailed configuration and design overhead are shown in Section 4.8.

4.2 3DICT Naïve Implementation

4.2.1 Small convolution. As Figure 7 shows, we can perform small convolutions the same way as that computes normal convolutions in [5, 28]. The LCNN dictionaries *dict* are written into RDEs, while the voltages representing *input* are applied on WLs. At last, the summed currents are sampled, held and converted on BLs by ADCs. Our baseline has only 1024 (2048) 128×128 ReRAM arrays (layers), while the LCNN-AlexNet with the smallest dictionary requires 3123 128×128 ReRAM layers to store all dictionary entries. The full precision AlexNet requires more than 1 million such ReRAM layers to record all weights. Therefore, it is impossible to conduct small convolutions without dictionary updates. We store as many dictionary entries of last several fully-connected layers of each LCNN in RDEs of our baseline, and write dictionary entries of every other layer into RDEs before the small convolution of that layer happens. The dictionary updates take turn to occur in all RDEs, so that all RDEs wear evenly.

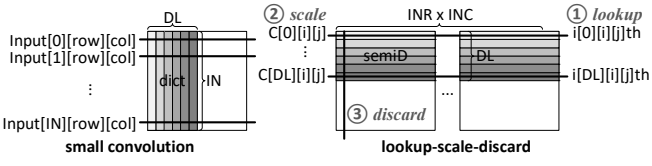
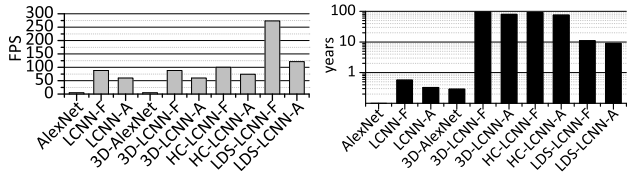


Figure 7: 3DICT naïve LCNN implementation.

4.2.2 Lookup-scale-discard operation. The small convolution produces DL channels of $INR \times INC$ *semiD* for each LCNN layer. All lookup-scale-discard operations occur on *semiD* as shown in Figure 7. Each channel of $INR \times INC$ *semiD* is converted to an one-dimensional vector and written to a row of an array. If $INR \times INC$ is larger than the row size, a *semiD* channel spans across multiple arrays. DL *semiD* channels occupy DL rows. The procedure of a lookup-scale-discard can be summarized as: ① based on each element of Index tensor I, the $i[outn][i][j]$th row is activated; ② the voltage representing the corresponding element of coefficient tensor C is applied on the WL of that row. The summed currents on BLs indicate the intermediate matrix AA in Figure 4; and ③ based on I, only the currents indicating the elements in the result matrix A are translated to digital signals by ADCs; other currents are skipped by ADCs and hence discarded.



(a) Performance. (b) Endurance.

Figure 8: Baseline performance and endurance.

4.2.3 Performance and endurance. Figure 8 exhibits the performance and endurance of our baseline computing LCNN-AlexNet tests. Our baseline can also directly execute full precision *AlexNet* tests. However, it achieves only 5 FPS and lasts only 1 day when constantly executing full precision tests, because of the short MLC ReRAM cell endurance (10^7 writes) and frequent weight updates. We define the endurance of our baseline accelerator as the interval from its first test to the time when its first row is worn-out. Our baseline with the naïve LCNN implementation obtains 88 (60) FPS and only about 6 (4) months lifetime when constantly testing LCNN-AlexNet *LCNN-F* (*LCNN-A*) with the smallest (largest) dictionary. The detailed dictionary configuration can be found in Section 4.6. Compare to a mobile CPU or GPU, this naïve implementation has better performance per Watt. The detailed comparison is shown in Section 6. Although LCNN-AlexNet improves the test performance by $10\times \sim 16\times$ on our baseline over original CNNs,

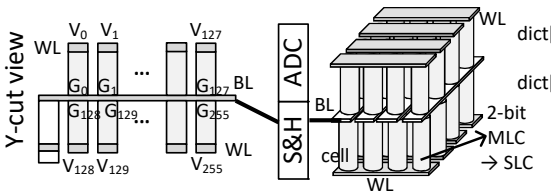


Figure 9: 3Dict array structure.

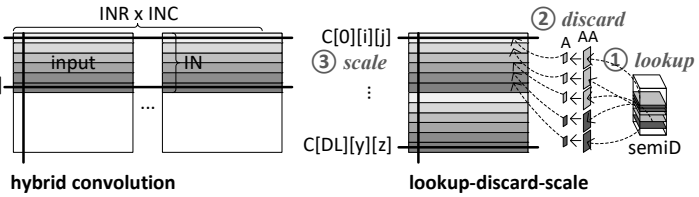
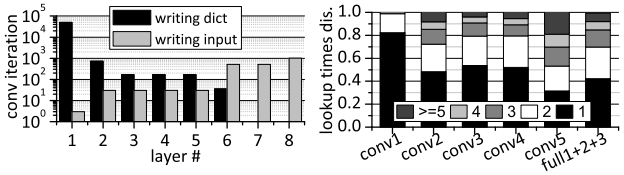


Figure 10: 3DICT architectural optimization.

the traditional RDE-based accelerator cannot guarantee reasonable lifetime when testing both CNN models.

4.3 Long Endurance 3D XPoint SLC RDE

To extend the endurance of RDEs, we propose a 3D XPoint single level cell (SLC) RDE using a ReRAM cell to record only 1 bit of a weight/dictionary. The 3D XPoint SLC RDE is shown in Figure 9. Compared to a MLC cell that can tolerate 10^7 writes, each SLC cell can sustain for $10^{10} \sim 10^{12}$ writes [32]. Therefore, our SLC RDE increases the lifetime by $10^3\times \sim 10^5\times$ over the traditional RDE design [28]. However, compared to the 2-bit MLC RDE, our SLC RDE has to occupy a doubled number of BLs, each of which is connected to 128 cells, to perform the same task. Since our SLC RDE also adopts the same 8-bit 12.8GSps ADC that translates only 128 BLs in each cycle, it reduces the dot-product computing throughput by 50% over the traditional RDE. Because of the tight power budget in mobile settings, doubling the ADC or/and array number in our baseline is not a practical solution to alleviate the dot-product performance degradation.



(a) Small con. iteration #. (b) Lookup #.

Figure 11: Optimization motivation in our naïve baseline.

To mitigate the dot-product throughput degradation, our SLC RDE adopts a 3D XPoint array structure shown in Figure 9. A prior 3D RDE [3] consists of multiple ReRAM layers, each of which has an independent copy of WLs and BLs to separate the dot-product results between each layer. However, in a conventional 3D XPoint ReRAM chip [34], two vertically connected layers share a set of WLs or BLs to improve the vertical integration scalability. Our 3D XPoint SLC RDE shares one set of BLs between two 128×128 ReRAM layers. Totally, in the 3D XPoint array, 256 cells (128 cells on a layer) are connected to a BL. Since a BL in this 3D structure has the same length as that in the 2D counterpart [28], the 3D XPoint BL resistance does not increase or degrade the summed current signal on the BL. By applying two sets of 128 input voltages on the WLs of the top and bottom layers respectively, each BL sums 256 1-bit×1-bit currents that can be translated to a digital value by our baseline 8-bit ADC. On the contrary, to avoid overflows, the prior 3D RDE design [3] requires higher bit-resolution ADCs consuming 7× chip area and 11× power consumption [24], since it uses 2-bit MLC cell in each layer and accumulates currents from multiple layers. With our long endurance 3D XPoint SLC RDEs, as Figure 8(b) shows, constantly testing AlexNet (*3D-AlexNet*) still fails our baseline within 1 year, while the nonstop execution of LCNN in our baseline (*3D-LCNN-F*) can last for nearly 95 years even exceeding the mobile device lifetime. Next, we will propose two architectural optimizations to accelerate the execution of LCNNs by sacrificing the endurance of 3D XPoint SLC RDEs.

4.4 Hybrid Convolution

As Figure 2 exhibits, small convolutions happen between *dict* and *input*. Since our baseline has too few RDE arrays to store all dictionary entries of a LCNN, at least for the first several layers, *dict* entries have to be written into RDE arrays before small convolutions happen. So the total latency of small convolutions of a LCNN layer includes the latency of writing *dict* entries and the RDE convolution latency. Although the size of *dict* is only $IN \times DL$, $INR \times INC$ convolution iterations are required to produce *semiD*. On the contrary, if we write *input* with the size of $INC \times INR \times IN$ into 3D RDE arrays shown in Figure 10, *semiD* can be calculated through DL convolution iterations. We convert each *input*[*inn*] into an one-dimensional vector with $INC \times INR$ elements and write it to a row in an array or spanning across multiple arrays. The entire *input* may occupy IN rows. Figure 11(a) highlights the convolution iteration number comparison between writing *dict* and *input* for each layer of LCNN-AlexNet. For 5 convolution (3 fully-connected) layers, writing *input* (*dict*) needs more writes but introduces less convolution iterations. Therefore, we propose Hybrid Convolution (HC) to write *input* for convolution layers but write *dict* for fully-connected layers to reduce the iteration number and the convolution latency. Our HC increases the LCNN-AlexNet performance with the smallest (largest) dictionary, *HC-LCNN-F* (*HC-LCNN-A*), to 100 (74) FPS (Figure 8(a)) and obtains 92 (75) years lifetime (Figure 8(b)) on our baseline with 3D XPoint SLC RDEs.

4.5 Lookup-Discard-Scale

Figure 7 shows the procedure of lookup-scale-discard operations. Although DL channels of *semiD* occupy DL rows, most lookup-scale-discard operations access, scale and accumulate only few channels of *semiD*. As Figure 11(b) shows, 82% lookup-scale-discards of the first layer of LCNN-AlexNet access and scale only 1 *semiD* channel. On average, only 8% lookup-scale-discards access, scale and accumulate more than 4 channels of *semiD*. Since each BL in our 3D XPoint SLC RDEs connects to 256 rows and can accumulate at most 256 currents during each lookup-scale-discard, most lookup, scaling and accumulation capability on a BL is wasted. We propose Lookup-Discard-Scale (LDS) to fully exploit the lookup, scaling and accumulation capability of BLs. The procedure of a LDS operation can be viewed in Figure 10 and described as: ① based on each element of Index tensor I, a channel of *semiD* (the intermediate matrix AA) is found; ② based on the element of I, only $OUTR \times OUTC$ out of $INR \times INC$ elements in an intermediate matrix AA are selected to form a result matrix A and written

into a row of a RDE array. All other elements in the intermediate matrix AA are discarded; and ③ after all result matrix As of Index tensor I are written into RDEs, the voltages representing the corresponding elements of coefficient tensor C are applied on the WLs of the corresponding rows. The summed currents on BLs are convert to the digital outputs of this LCNN layer by ADCs. Our LDS and HC improve the LCNN-AlexNet performance with the smallest (largest) dictionary, *LDS-LCNN-F* (*LDS-LCNN-A*), to 273 (121) FPS (Figure 8(a)) and achieves 11 (9) years endurance (Figure 8(b)) in our baseline with 3D XPoint SLC RDEs.

4.6 QoS Support

To enable the capability of QoS control, 3DICT stores 7 dictionaries with various sizes for a CNN, e.g., AlexNet, in a 64MB ReRAM read-only array. The mobile host platform such as a CPU controlled by programmers can actively inform our 3DICT to use a particular dictionary to perform LCNN tests. Each dictionary deliveries a LCNN test accuracy expectation and a performance guarantee. The performance and LCNN test accuracy achieved by these 7 dictionaries can be found in Figure 12. Our 3DICT is the first CNN accelerator that enables a graceful trade-off between LCNN test accuracy and latency. With the smallest dictionary (4.3MB), 3DICT provides 68% top-5 test accuracy and 273 FPS test performance. In contrast, 3DICT having the largest dictionary (9.6MB) achieves 78% top-5 test accuracy and 121 FPS test throughput.

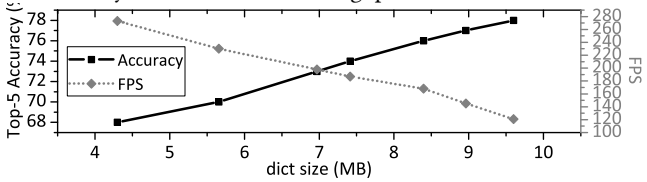


Figure 12: 3DICT QoS Support on AlexNet.

When constantly using one dictionary, the dictionary entries of the last several full-connected layers are left in our 3D XPoint SLC RDEs. So, before computing full-connected layers, a LCNN test needs to update only the RDEs that are modified by first several convolutional layers. However, if the host CPU decides to change to another dictionary different from that are used in the previous LCNN test, 3DICT requires 246μs to update all 1024 (2048) RDE arrays (layers). On average, the performance of 3DICT with various dictionaries reduces 1 FPS, because of this extra updating latency.

4.7 Variations on Array Latency and Endurance

Previous works [31, 33] observe that particularly during RESETs, ReRAM arrays suffer from serious sneak path currents that significantly prolong the RESET latency. The amount of sneak path current is decided by the writing cell position in the array and how many low resistance state (LRS) cells along its BL and WL. However, our 3D XPoint SLC arrays do not have the sneak path current problem, since they adopt Ag/Hafnia-based access devices [22] with 10^{10} ON/OFF current ratio to suppress the sneak path current. Moreover, the size of our arrays is only 128×128. Based on the evaluation in [31], such small array size is immune to sneak path currents. We also adopted the fine-grained RESET/SET current provision scheme [17] to eliminate over-RESETs and over-SETs in our small arrays. Therefore, we assume each SLC cell in our arrays can tolerate at least 10^{10} writes.

4.8 Design Overhead

The power and area overhead of 3DICT is shown in Table 1. To compare against ISAAC [28], we adopt its ADC, DAC, S&H, S&A, pooling, activation logic design and the same 32nm process technology. The router and bus are modeled and estimated through Cadence Virtuoso with 32nm PTM technology. Instead of eDRAMs, we used ReRAMs as I/O buffers and the dictionary storage. We used NVSim [10] to estimate the latency, power and area of ReRAM arrays. To support our 3D XPoint SLC RDE design, 256 1-bit DACs (inverters) are required in each HA. We adopted a ReRAM cell model from [31], where a RESET costs 26.4μW and 10ns while a SET requires 11.3μW and 15ns. So with only $\sim$ 400mW power budget, at most 16 arrays with 128-bit rows can be written simultaneously. Since a 2-bit MLC write costs a RESET and multiple SETs, we assume the MLC write latency is 100ns.

Table 1: DictR power and area

Name	Component	Spec	Power (mW)	Area (mm^2)
Hierarchical Array (HA)	ADC×1	8-bit 1.28GSps	2	0.0012
	DAC×256	1-bit, inverter	1	0.00025
	S&H×128	sample & hold	0.0125	0.000005
	Array×8	128 × 128 2-layer	0.3	0.0002
Sub-total			3.3125	0.001655
Multiply Accumulate Unit (MAU)	HA×8		26.5	0.01324
	S&A×4	shift & add	0.2	0.00024
	ReRAM 1KB	I/O buffer	0.15	0.0005
Sub-total			26.85	0.01398
3DICT	MAU×16		429.6	0.224
	Sigmoid×2	activation	0.52	0.0006
	S&A×1	shift & add	0.4	0.00006
	MaxPool×1	pooling	0.31	0.00024
	Router and bus	connection	3	0.04
	ReRAM 1KB	I/O buffer	0.15	0.0005
dict storage	ReRAM 64MB	power gating	3.6 (0)	0.16
Total			433.98	0.4254

Table 2: CNN benchmarks (C: convolutional; S: pooling; F: fully-connected; $Orig$: original; $LCNN-A$: LCNN accurate; $LCNN-F$: LCNN fast; $XNOR$: XNOR-Net**).**

Name	DataBase	Topology	Top-5 Accuracy(%)			
			$Orig$	$3DICT-A$	$3DICT-F$	$XNOR$
LeNet-5	MNIST	3C,2S,1F	99.1	98.8	97.2	97.2
MLP	MNIST	5F	98.5	98.1	97.1	96.9
CNP	MNIST	3C,2S,1F	97.0	96.8	96.2	96.1
SCNN	MNIST	2C,2F	99.0	98.2	97.7	97.8
MCNN	MNIST	3C,3S,3F	96.8	96.1	95.7	95.7
AlexNet	ImageNet	5C,3S,2F	80.2	78.1	68.7	69.2
ResNet-18	ImageNet	18C,2S,1F	89.2	84.6	76.8	73.2

Table 3: Simulated scheme comparison.

Name	Description	$Power_{acc}$	$Power_{mem}$
Envision [23]	complex CNNs	62mW [23]	1.91W [18]
YodaNN [1]	BinaryConnect	248mW [1]	1.91W [18]
Eyeriss [4]	complex CNNs	278mW [4]	1.91W [18]
TETRIS [12]	HMC PIM	8.42W [12]	0
XNOR-POP [18]	XNOR-Net PIM	2.15W [18]	0
mISAAC	mobile ReRAM CNN	435.58mW	0
3DICT	ReRAM LCNN	435.58mW	0

5 EXPERIMENT METHODOLOGY

Workload. We studied six CNNs including LeNet-5 [20], CNP [11], SCNN [29], MCDNN [7], AlexNet [19] and ResNet-18 [15], and a Multilayer Perceptron(MLP) [20]. LeNet-5, CNP, MLP, SCNN, and MCDNN were trained with MNIST to identify simple handwritten digits, while AlexNet and ResNet-18 were trained with ImageNet to recognize complex objects. Compared to other deep CNNs with more layers and weights, ResNet-18 obtains approximately

the same or even better test accuracy [15]. We trained all networks by Torch7 [8]. More network details can be viewed in Table 2, where besides the network topologies, we also show the test accuracy of each network. Compared to full precision CNNs (*Orig*), XNOR-Net [26] reduces the test accuracy (top-5) of AlexNet and ResNet-18 by 13.7% and 17.9% respectively. By adopting different dictionaries, LCNN can explore the trade-off between test accuracy and speed. Therefore, we show 3DICT testing LCNNs with two dictionary configurations: 3DICT-A achieves high test accuracy (only 1% $\sim$ 5% degradation, compared to original CNNs) and slow test speed by using the largest dictionary (9.6MB); and 3DICT-F attains low test accuracy (similar to that of XNOR-Net) and fast test speed through computing with the smallest dictionary (4.3MB). More details on the dictionary configuration can be viewed in Section 4.6.

Schemes. We compared 3DICT against 6 counterparts shown in Table 3. We selected 4 mobile CNN accelerators including Envision, YodaNN, Eyeriss and XNOR-POP. Both Envision and Eyeriss are designed for processing full precision CNNs, while YodaNN takes advantage of BinaryConnect [9] to test CNNs with only additions and subtractions. XNOR-POP is a Wide-IO2-based PIM using XNOR-Net [26] to test CNNs with XNORs and popcounts. We also chose a 3D DRAM-based CNN PIM TETRIS [12] running full precision CNNs. We customized a mobile version of ReRAM PIM ISAAC to run full precision CNNs by using our baseline configuration in Section 4.1. We also adopted a mobile SoC platform Jetson KT1 including a 4-core ARM-A57 CPU and a 256-CUDA-core NVIDIA GPU. The mobile CPU consumes $1W$, while the mobile GPU costs $7W$. Since the 1GB Wide-IO2 DRAM is an indispensable component of XNOR-POP, for a fair comparison, we assume the same DRAM configuration for all other platforms (except PIMs) that require a main memory system to buffer weights.

Accelerator modeling. We used a heavily modified deep learning accelerator simulator FODLAM [27] to study the performance, power and energy consumption of all accelerators. Based on a user-defined accelerator configuration and a neutral network description, FODLAM can generate the performance, power and energy details of the accelerator running that network. The FODLAM model has been correlated and validated by real accelerator chips such as Eyeriss [4]. We also implemented micro-architectural pipeline details of all accelerators into FODLAM. All neutral network descriptions are obtained through Torch7 [8].

6 EVALUATION

Comparison against a CPU and a GPU. Figure 13 shows the performance comparison between a CPU, a GPU and our 3DICT. Directly executing full precision AlexNet tests on the mobile CPU or GPU cannot achieve competitive performance. When testing full precision AlexNet, the ARM CPU only has 1 FPS, while the mobile GPU attains 18 FPS, both of which are smaller than the minimal real-time performance requirement 30 FPS. Testing LCNN-AlexNet cannot help the mobile ARM CPU reach 30 FPS neither. However, the mobile GPU obtains 30 $\sim$ 284 FPS when testing LCNN-ALexNet with various sizes of dictionary. Our 3DICT achieves 273 (113) FPS when computing LCNN-AlexNet tests with the smallest (largest) dictionary. Compared to the mobile GPU, our 3DICT improves the performance per Watt of LCNN-AlexNet tests with various sizes of dictionary by 21× $\sim$ 85×.

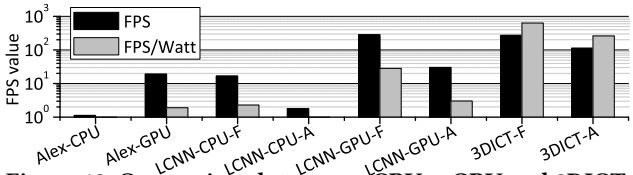


Figure 13: Comparison between a CPU, a GPU and 3DICT.

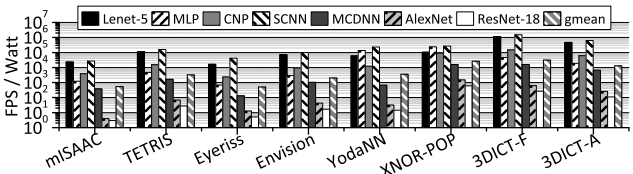


Figure 14: Performance per watt comparison.

Performance per watt comparison against accelerators. The performance per watt comparison of all accelerators is shown in Figure 14. For complex CNNs like AlexNet and ResNet-18, only XNOR-POP and 3DICT achieve all > 30 FPS per watt. Low power ASIC-based mobile accelerators, e.g., Eyeriss, Envision and YodaNN, has limited hardware resources and thereby failing to provide strong enough CNN test processing power. Moreover, to buffer weights, they also require an independent main memory system consuming extra power. The mobile ISAAC cannot efficiently test full precision CNNs, since it has to frequently update weights into 2-bit MLC ReRAMs. TETRIS fails to obtain high performance per Watt, since it relies on power-hungry 3D DRAM structure and interface. Compared to 3DICT-F, XNOR-POP increases the FPS per watt on AlexNet by 137% and ResNet-18 by 130%. This is because large WideIO2 rows can execute the most intensive operations in XNOR-POP, XNORs, in huge throughput with low power consumption. However, XNOR-POP does not have the ability to explore the trade-off between CNN test speed and accuracy. It can only execute fast CNN tests with low accuracy. Therefore, 3DICT-F wins the second best performance per watt among all accelerators when processing complex CNNs. For simple CNNs such as LeNet-5 and CNP, all accelerators can obtain > 100 FPS per watt. Particularly, 3DICT-F achieves the best performance per watt. XNOR-POP cannot compete with 3DICT-F on simple CNNs, since the number of weights in simple CNNs is too small to fully utilize the large DRAM row. Compared to XNOR-POP, averagely 3DICT-F improves the test performance per Watt by 13.7%. Compared to 3DICT-F, 3DICT-A increases the test accuracy by 14.7% (to 78.1%) but reduces the FPS per Watt by 41%.

Energy comparison against other accelerators. The energy comparison of all accelerators is shown in Figure 15. Among all accelerators, XNOR-POP and 3DICT spend the least energy in testing an image, since their throughput is high and power consumption is low. XNOR-POP costs less energy when testing complex CNNs such as AlexNet and ResNet, while 3DICT-F consumes less energy during tests of simple CNNs, e.g., Lenet-5 and CNP. On average, 3DICT-F reduces the CNN test energy by 12.1% over XNOR-POP. Compared to 3DICT-F, 3DICT-A increases the test energy by 141%.

Endurance of 3DICT. As Figure 8(b) exhibits, with both architectural optimizations, 3DICT-F obtains 11-year endurance while 3DICT-A has 9-year lifetime when constantly testing AlexNet. In LCNNs, compared to convolution layers, fully-connected layers have much larger dictionaries. AlexNet relies on three fully-connected

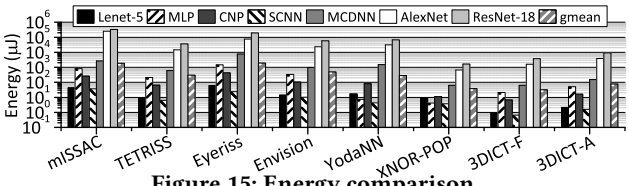


Figure 15: Energy comparison.

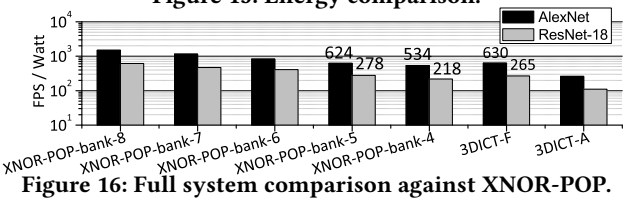


Figure 16: Full system comparison against XNOR-POP.

layers, but ResNet has only one fully-connected layer. Therefore, when execute nonstop ResNet-18, 3DICT-F achieves 23-year endurance and 3DICT-A attains 19-year lifetime. For simple CNNs and MLPs, the dictionary sizes are much smaller than those of complex CNNs such as AlexNet. The endurance of 3DICT executing simple CNNs and MLP exceeds 20 years averagely.

Full system comparison against XNOR-POP. Although XNOR-POP has better FPS per Watt over 3DICT-F, it occupies all Wide-IO2-based main memory banks during CNN tests. Recent works [6, 21] show that the main memory is a key hardware for mobile devices to buffer and exchange the data from all components including the CPU, GPU, display, audio and modem. Therefore, the main memory frequently receives concurrent accesses from all components [6]. However, when the open row in an XNOR-POP bank serves memory requests, it is impossible to test CNNs in the bank, since key peripherals such as TSVs, decoders, and sense amplifiers are occupied by the accesses. Figure 16 exhibits the performance per Watt degradation of XNOR-POP in a mobile system when several banks are serving memory accesses from other components. When four banks are occupied by the accesses from other components, the FPS per Watt of XNOR-POP is reduced to 534 for AlexNet and 218 for ResNet-18, both of which are worse than those of 3DICT-F. In contrast, 3DICT can perform LCNN tests independently and is not influenced by the main memory system.

7 CONCLUSION

In this paper, we present a reliable and QoS-capable 3D XPoint ReRAM-based PIM design, 3DICT, to accelerate LCNN tests for low power mobile devices by hybrid convolutions and lookup-dicard-scale operations. With 3D XPoint SLC RDEs, 3DICT can guarantee $>$ 5 years lifetime in mobile devices. By adopting different sizes of dictionary, 3DICT flexibly provides various QoS options on CNN test latency and accuracy to all tasks involving CNN tests and running on the same mobile device. Compared to state-of-the-art accelerators, 3DICT improves the CNN test performance per Watt by 13% $\sim$ 61$\times$ under various CNN test accuracy requirements.

REFERENCES

[1] R. Andri, *et al.*, "YodaNN: An Ultra-Low Power CNN Accelerator Based on Binary Weights," in *ISVLSI*, pages 236–241, July 2016.
[2] H. Bagherinezhad, *et al.*, "LCNN: Lookup-based Convolutional Neural Network," in *CVPR*, 2017.
[3] B. Chakrabarti, *et al.*, "A multiply-add engine with monolithically integrated 3D memristor crossbar/CMOS hybrid circuit," *Scientific Reports*, 7, 2017.
[4] Y. H. Chen, *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *ISSCC*, 2016.
[5] P. Chi, *et al.*, "PRIME: A Novel PIM Architecture for Neural Network Computation in ReRAM-Based Main Memory," in *ISCA*, 2016.
[6] N. Chidambaram Nachiappan, *et al.*, "GemDroid: A Framework to Evaluate Mobile Platforms," in *SIGMETRICS*, 2014.
[7] D. Ciregan, *et al.*, "Multi-column deep neural networks for image classification," in *CVPR*, 2012.
[8] R. Collobert, *et al.*, "Torch7: A Matlab-like Environment for Machine Learning," in *BigLearn, NIPS Workshop*, 2011.
[9] M. Courbariaux, *et al.*, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *NIPS*, 2015.
[10] X. Dong, *et al.*, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory," *TCAD*, 2012.
[11] C. Farabet, *et al.*, "CNP: An FPGA-based processor for Convolutional Networks," in *FPL*, 2009.
[12] M. Gao, *et al.*, "TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory," in *ASPLOS*, 2017.
[13] P. Gu, *et al.*, "Technological exploration of RRAM crossbar array for matrix-vector multiplication," in *ASPDAC*, 2015.
[14] S. Han, *et al.*, "MCDNN: An Approximation-Based Execution Framework for Deep Stream Processing Under Resource Constraints," in *MobiSys*, 2016.
[15] K. He, *et al.*, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.
[16] M. Hu, *et al.*, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *DAC*, 2016.
[17] L. Jiang, *et al.*, "Enhancing Phase Change Memory Lifetime through Fine-Grained Current Regulation and Voltage Upscaling," in *ISLPED*, 2011.
[18] L. Jiang, *et al.*, "XNOR-POP: A processing-in-memory architecture for binary Convolutional Neural Networks in Wide-IO2 DRAMs," in *ISLPED*, 2017.
[19] A. Krizhevsky, *et al.*, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.
[20] Y. Lecun, *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 86(11), Nov 1998.
[21] S. E. Lee, *et al.*, "Accelerating mobile augmented reality on a handheld platform," in *ICCD*, 2009.
[22] R. Midya, *et al.*, "Anatomy of Ag/Hafnia-Based Selectors with 10^{10} Nonlinearity," *Advanced Materials*, 29(12), 2017.
[23] B. Moons, *et al.*, "Envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network processor in 28nm FDSOI," in *ISSCC*, 2017.
[24] B. Murmann, "An ADC Performance, Power and Area Survey from 1997 to 2017," http://web.stanford.edu/~murmann/adcsurvey.html.
[25] L. Ni, *et al.*, "An Energy-Efficient Digital ReRAM-Crossbar-Based CNN With Bitwise Parallelism," *JESSCDC*, 2017.
[26] M. Rastegari, *et al.*, "XNOR-Net: Imagenet Classification Using Binary Convolutional Neural Networks," in *ECCV*, 2016.
[27] A. Sampson and M. Buckler, "FODLAM: a first-order deep learning accelerator model," https://github.com/cucapra/fodlam.
[28] A. Shafiee, *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA*, 2016.
[29] P. Y. Simard, *et al.*, "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis," in *ICDAR*, 2003.
[30] T. Tang, *et al.*, "Binary convolutional neural network on RRAM," in *ASP-DAC*, 2017.
[31] W. Wen, *et al.*, "Speeding up crossbar resistive memory by exploiting in-memory data patterns," in *ICCAD*, 2017.
[32] H. S. P. Wong, *et al.*, "Metal-Oxide RRAM," *Proceedings of the IEEE*, 2012.
[33] C. Xu, *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *HPCA*, 2015.
[34] T. y. Liu, *et al.*, "A 130.7-mm^2 2-Layer 32-Gb ReRAM Memory Device in 24-nm Technology," *JSSC*, 2014.
[35] C. Zhang, *et al.*, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," in *FPGA*, 2015.
[36] P. Zhou, *et al.*, "A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology," in *ISCA*, 2009.