# RAELLA: Reforming the Arithmetic for Efficient, Low-Resolution, and Low-Loss Analog PIM: No Retraining Required!

Tanner Andrulis
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA
andrulis@mit.edu

Joel S. Emer
Massachusetts Institute of Technology, Nvidia
Cambridge, Massachusetts, USA
jsemer@mit.edu

Vivienne Sze
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA
sze@mit.edu

## ABSTRACT

Processing-In-Memory (PIM) accelerators have the potential to efficiently run Deep Neural Network (DNN) inference by reducing costly data movement and by using resistive RAM (ReRAM) for efficient analog compute. Unfortunately, overall PIM accelerator efficiency is limited by energy-intensive analog-to-digital converters (ADCs). Furthermore, existing accelerators that reduce ADC cost do so by changing DNN weights or by using low-resolution ADCs that reduce output fidelity. These strategies harm DNN accuracy and/or require costly DNN retraining to compensate.

To address these issues, we propose the RAELLA architecture. RAELLA adapts the architecture to each DNN; it lowers the resolution of computed analog values by encoding weights to produce near-zero analog values, adaptively slicing weights for each DNN layer, and dynamically slicing inputs through speculation and recovery. Low-resolution analog values allow RAELLA to both use efficient low-resolution ADCs and maintain accuracy without retraining, all while computing with fewer ADC converts.

Compared to other low-accuracy-loss PIM accelerators, RAELLA increases energy efficiency by up to 4.9× and throughput by up to 3.3×. Compared to PIM accelerators that cause accuracy loss and retrain DNNs to recover, RAELLA achieves similar efficiency and throughput without expensive DNN retraining.

## CCS CONCEPTS

• **Computer systems organization** → **Analog computers**; **Neural networks**; • **Hardware** → *Emerging architectures.*

## KEYWORDS

processing in memory, compute in memory, analog, neural networks, accelerator, architecture, slicing, ADC, ReRAM

**ACM Reference Format:**
Tanner Andrulis, Joel S. Emer, and Vivienne Sze. 2023. RAELLA: Reforming the Arithmetic for Efficient, Low-Resolution, and Low-Loss Analog PIM: No Retraining Required!. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23), June 17–21, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3579371.3589062


*ISCA '23, June 17–21, 2023, Orlando, FL, USA.*

ACM ISBN 979-8-4007-0095-8/23/06.
https://doi.org/10.1145/3579371.3589062

## 1 INTRODUCTION

Processing-In-Memory (PIM) is a promising solution to the high compute and energy cost of Deep Neural Network (DNN) inference. By computing in memory [34], PIM accelerators avoid expensive off-chip movement of the DNN weights [59]. Furthermore, PIM accelerators often utilize Resistive-RAM (ReRAM) devices and ReRAM crossbars [5, 54, 56] for dense and efficient analog compute [34].

Unfortunately, while ReRAM crossbars can compute efficiently and with high density, overall PIM accelerator energy is often dominated by the analog-to-digital converters (ADCs) that read computed analog values from crossbars. Due to ADC overhead, some PIM accelerators [47, 54] do not significantly improve energy over non-PIM accelerators [4] despite the opportunities in PIM.

Some prior works attempt to reduce this ADC overhead by reducing the resolution of the ADC, which exponentially decreases ADC energy [65]. Architectures often partition, or *slice*, the bits in DNN inputs and weights into multiple lower-resolution slices and compute with different slices in multiple steps [54]. Although sliced arithmetic can use lower-resolution ADCs, ADCs must process the results of each slice, so *these strategies replace each high-resolution ADC convert with multiple low-resolution ADC converts*, and therefore ADCs still dominate overall energy.

Other PIM accelerators reduce ADC energy, but do so at the expense of DNN accuracy. Some designs prune DNNs [8, 26, 48, 75, 80] to reduce DNN weight count, so we call these designs *Weight-Count-Limited*. They reduce the computation count and ADC converts required, but also introduce accuracy loss. Alternatively, other designs use efficient lower-resolution ADCs to process high-resolution analog values from crossbars [5, 7, 24]. We call these designs *Sum-Fidelity-Limited* as the resolution difference reduces output fidelity and introduces error. These architectures requantize DNNs to tolerate ADC resolution limitations, which again causes accuracy loss.

To reduce this accuracy loss, both Weight-Count-Limited and Sum-Fidelity-Limited architectures retrain DNNs. This is a problem; DNN training has a very high computational cost [43], can require cumbersome hyperparameter tuning to achieve high accuracy [19], and may be impossible if the training data is private [50, 62]. Furthermore, cutting-edge DNNs often require particular training schemes [6], which may not be compatible with the retraining scheme required by an architecture.

To avoid accuracy loss without imposing retraining, we look at fidelity limitations. We define *fidelity* as the ability of the ADC to represent the full resolution of computed analog values. Architectures lose fidelity and generate errors when the computed analog value resolution is higher than the ADC resolution. Each

Models of RAELLA are available at https://github.com/mit-emze/raella

DNN produces many distributions of analog values, and prior Sum-Fidelity-Limited approaches modify DNNs to reshape these analog value distributions to fit a resolution-limited ADC range. In contrast, we observe that we can reshape analog value distributions with an adaptable architecture, rather than changing the DNN.

Using this key insight, we propose the RAELLA architecture to enable efficient PIM inference without retraining. RAELLA modifies arithmetic and slicing, shaping computed value distributions to produce low-resolution analog results. This allows RAELLA to use efficient low-resolution ADCs while maintaining high fidelity and low DNN accuracy loss. The main contributions of RAELLA are:

- Center+Offset encoding to accumulate more values in the analog domain while keeping small, low-resolution sums. Specifically, RAELLA shifts DNN weights to equalize the average magnitude of the positive and negative weight slices in each crossbar column. As analog-domain calculations are accumulated, positive and negative results negate to produce near-zero sums that can be converted with high fidelity.
- Adaptive Slicing of DNN weights at compilation time to balance density, efficiency, and fidelity. Storing more bits in each ReRAM device is denser and more efficient but creates higher-resolution analog values. For each DNN layer, RAELLA adapts the number of ReRAM devices per weight and the number of bits in each ReRAM device. This enables RAELLA to use the densest and most efficient strategies possible while keeping computed analog values low-resolution.
- Dynamic Slicing of DNN input activations at runtime for both efficient and high-fidelity computation. RAELLA speculates with an efficient strategy that processes with more bits in each input slice. RAELLA detects and recovers from incorrect results using a less efficient, higher-fidelity strategy that processes inputs with more slices using fewer bits each. This allows RAELLA to further reduce the number of ADC conversions without reducing fidelity.

Compared to other low-accuracy-loss PIM accelerators [54], RAELLA can both lower ADC resolution and run DNNs with up to 14× fewer ADC conversions without sacrificing fidelity.

We evaluate RAELLA on seven representative DNNs against three state-of-the-art PIM accelerators. Compared to other low-accuracy-loss PIM accelerators, RAELLA improves energy efficiency by up to 4.9× (geomean 3.9×) and throughput by up to 3.3× (geomean 2.0×). Compared to Weight-Count-Limited and Sum-Fidelity-Limited accelerators that require DNN retraining to recover accuracy, RAELLA provides similar efficiency and throughput while avoiding expensive DNN retraining.

## 2 BACKGROUND AND MOTIVATION

We first give a brief overview of DNN inference, Processing-In-Memory (PIM), and slicing to lower the resolution of analog operands. We then explore how ADCs limit PIM, how to reduce ADC energy, and the limitations of prior approaches.

### 2.1 Deep Neural Network (DNN) Inference

Modern DNNs are dominated by matrix-vector operations in convolutional and fully connected layers [59]. For inference, 8-bit (8b) per-channel quantized DNNs with 8b inputs/weights and 16b partial sums (psums) are widely available and can achieve high accuracy [22, 25, 37, 51, 82]. RAELLA supports this type of quantization.

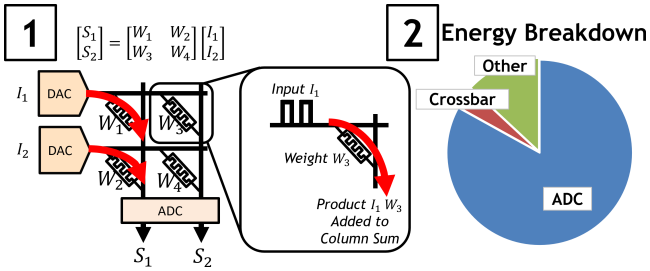


**Figure 1: Basic PIM crossbar. 1 $2 \times 2$ MVM. Each operand is a single slice. 2 Energy breakdown of an ISAAC-based design.**

DNNs may have billions of multiply-accumulate operations (MACs) over millions of weights [16]. This makes PIM an attractive choice for DNN inference acceleration. PIM can operate directly in weight memory to reduce data movement [5] and use Resistive-RAM (ReRAM) for dense and efficient analog compute.

### 2.2 ReRAM Properties

The functional unit of PIM systems is the ReRAM crossbar. ReRAM crossbars accelerate DNN layers by computing dense in-memory matrix-vector multiplications. Furthermore, ReRAMs are small and offer high storage density [45]. This density allows ReRAM-based systems to store and run on-chip pipelines that compute DNN layers sequentially [54, 56] without costly accesses to off-chip memory [59]. A disadvantage of ReRAM is high write energy [34]. Write cost is amortized in inference as ReRAM is nonvolatile, so written weights can be reused for many inferences [54].

Fig. 1 shows a basic $2 \times 2$ matrix-vector multiplication executed on a $2 \times 2$ ReRAM crossbar. A matrix of weights $W$ is programmed in the ReRAMs. Elements of the input vector $I$ are fed to digital-to-analog converters (DACs), which convert the inputs to analog values. Each ReRAM device multiplies the input on the row with its programmed weight. Products are accumulated in each column to produce analog *column sums*, which are converted by an analog-to-digital converter (ADC) to produce the digital result $S$.

ReRAMs have been shown to be programmable with up to 5b [1] and 512×512 crossbars have been shown to compute up to 8b column sums [17] under analog noise limitations. These resolution limits necessitate *slicing* to compute higher-resolution DNN layers.

### 2.3 Arithmetic with Slices

To run 8b DNN inference using lower-resolution devices, PIM architectures partition, or *slice*, input and weight bits into *input slices* and *weight slices*. A slice is a subset of bits from an operand, and multiplying two slices yields a *sliced product*.

There are two types of slicing. Temporal slicing processes slices in separate cycles (*e.g.*, bit-serial being the extreme with one bit per slice) and spatial slicing processes slices in separate ReRAMs across parallel crossbar columns. In most PIM accelerators that slice, temporal slicing is used for inputs and spatial slicing is used for weights. We refer to a vector of weights and their slices as a

**Dot Product:** 2b input $i_h i_l \cdot$ 2b weight $w_h w_l$

| **Sliced Input** | | | ✓ | | ✓ |
|---|---|---|---|---|---|
| **Sliced Weight** | | | | ✓ | ✓ |
| Cycle | Column | | | | |
| 1 | 1 | $i_h i_l \cdot w_h w_l$ | $i_h \cdot w_h w_l$ | $i_h i_l \cdot w_h$ | $i_h \cdot w_h$ |
| 1 | 2 | - | - | $i_h i_l \cdot w_l$ | $i_h \cdot w_l$ |
| 2 | 1 | - | $i_l \cdot w_h w_l$ | - | $i_l \cdot w_h$ |
| 2 | 2 | - | - | - | $i_l \cdot w_l$ |
| **Bits/MAC** | | 4 | 2 | 2 | 1 |
| **Converts/MAC** | | 1 | 2 | 2 | 4 |

**Table 1: How Slicing Works & Tradeoffs. A 2b input/weight are multiplied and each may be sliced into two 1b slices. High and low order bits are $i_h$, $w_h$ and $i_l$, $w_l$. Each column/cycle computes the sliced product shown. More slices reduce bits/slice and bits/MAC, permitting a cheaper, lower-resolution ADC. However, cycles, columns, and ADC converts are needed to process each slice. More slices increase ADC Converts/MAC.**

*weight filter* if they are mapped to the same set of columns in one crossbar and they contribute to one dot product for a DNN layer.

Table 1 shows an example of sliced arithmetic. Each weight slice is mapped spatially to one crossbar column, while each input slice is processed temporally in one cycle. For each column and cycle, an ADC converts the column sum. The result is shifted and added digitally, allowing PIM architectures to calculate full 16b psums despite low-resolution analog limitations [14, 54].

Table 1 shows tradeoffs relating to slicing. Many costs increase with more slices: each additional input slice increments $Cycles/Input$ while each additional weight slice increments $Columns/Weight$. $ADC\ Converts$ scales with the product of input and weight slice counts. The benefit of more slices is that we can use fewer bits per slice, thus reducing MAC resolution and required ADC resolution. We can also decrease *ADC Converts* by using larger crossbars that accumulate more analog values across more rows, but this also increases the required ADC resolution.

## 2.4 ADCs Limit PIM Accelerators

Fig. 1 shows the power breakdown of an 8b PIM architecture based on the foundational ISAAC [54]. PIM crossbars are dense and efficient, but are limited by ADC costs. Crossbars can compute 8b MACs with < 100fJ, but overall energy is dominated by ADCs. Crossbars are dense, but architectures can spend 5 [24] to 50 [54] times more area on ADC than crossbars. Crossbars can compute with high parallelism, scaling to 1024 rows [32], but the area and energy of ADCs scale exponentially with resolution [65]. Prior work has been limited to as few as 16 activated rows [75] to reduce column sums and ADC resolution requirements.

As ReRAM is dense and low power, RAELLA trades off more ReRAM for lower-resolution ADCs. Furthermore, by reducing resolution, we use more crossbar rows/columns with less ADC area/energy scaling. This higher parallelism yields higher throughput and efficiency for the full RAELLA accelerator.

## 2.5 Reducing ADC Cost

To run efficient PIM inference, we must reduce ADC area and energy. To do so, we present the Titanium Law of ADC energy.[1][2]

Table 2 shows the Titanium Law equation for ADC energy and breaks down its factors. ADC energy is the product of four terms:

- *Energy/Convert* is determined by ADC efficiency and scales exponentially with ADC resolution [65].[3]
- *Converts/MAC* is determined by the number of crossbar rows, input slices, and weight slices.
- *MACs/DNN* is determined by the DNN workload.
- *1/Utilization* corresponds to how many crossbar rows are used by the DNN. A utilization of one means all rows used.

Given these factors, Table 2 shows how to reduce ADC energy by changing hardware attributes. First, notice the tradeoff generated by *Energy/Convert* and *Converts/MAC* in the first/second rows of the table. Although it may seem that slicing and resizing the crossbar can directly reduce ADC energy, this approach has limited benefits. This is because, to reduce *Converts/MAC*, we must either (1) increase the crossbar rows and compute more sliced products per ADC convert, (2) increase bits per weight slice, which reduces the number of columns needed to store each weight and reduces the number of ADC converts needed to process each column, or (3) increase bits per input slice, which reduces the number of cycles required and ADC converts to process column sums over all cycles. The limitation, however, is that in all cases we will accumulate larger and higher-resolution column sums. To preserve fidelity, a higher-resolution ADC is needed, which increases *Energy/Convert* and negates our benefits. The converse is true for reducing *Energy/Convert*; preserving high fidelity requires increasing *Converts/MAC*.

The final column of Table 2 shows the consequences of reducing each of the Titanium Law terms. Of the six consequences, three are ineffective for reducing ADC energy. *Converts/MAC* and *Energy/Convert* trade off with each other. *1/Utilization* cannot be reduced below one.

Architectures that reduce the ADC energy choose options that end in the consequence **Accuracy Loss or Retraining**. Fig. 2 shows how these architectures change DNN operands and lose accuracy. Weight-Count-Limited architectures, in the Ⓦ-marked cells of Table 2, prune/reshape DNN weights to lower *MACs/DNN*. Unfortunately, changing weights causes DNN accuracy loss unless the DNN is retrained. On the other hand, Sum-Fidelity-Limited architectures, in the Ⓢ-marked cells, use more rows, more bits per input/weight slice, and low-resolution ADCs to reduce both *Converts/MAC* and *MACs/DNN*. But because they generate large, high-resolution column sums and use low-resolution ADCs, they lose column sum fidelity. This creates errors in outputs and causes accuracy loss unless the DNN is requantized and retrained.

Counterintuitively, 8b ADCs are not always sufficient for 8b-quantized DNNs in Sum-Fidelity-Limited architectures. Psums are 16b after MACs of 8b inputs and weights, and high-accuracy linear quantization strategies need the full 16b psum range [51, 69, 82], so we would like all 16b to have high fidelity. Sum-Fidelity-Limited architectures may generate > $8b$ column sums and capture them with an 8b ADC. When the ADCs of these architectures lose bits from column sums, they lose bits from the overall psum. This limits

[1] Inspired by the Iron Law [68] and titanium-based ReRAM devices [13].

[2] While ADC energy is the focus here, a similar analysis can be performed for area by substituting *Converts/MAC* with *#ADCs/Throughput*.

[3] *Energy/Convert* can also be reduced with clever new ADC designs, but there is an efficiency limit [35] due to analog noise. This requires innovations on both the ADC and architecture sides.

**The Titanium Law:** $\frac{ADCEnergy}{DNN} = \frac{Energy}{Convert} \times \frac{Converts}{MAC} \times \frac{MACs}{DNN} \times \frac{1}{Utilization}$

| Term | Hardware Attribute | How to Reduce | Tradeoff | Consequence |
|---|---|---|---|---|
| Energy/Convert | ADC Resolution | Reduce ADC Resolution | Fewer Crossbar Rows or Bits/Slice | High *Converts/MAC* |
| | | | Ⓢ Fidelity Loss, Psum Errors | Ⓢ**Accuracy Loss or Retraining** |
| *Converts/MAC* | Crossbar Rows | Increase Crossbar Rows or Bits/Slice | High-Resolution ADC | High Energy/Convert |
| | | | Ⓢ Fidelity Loss, Psum Errors | Ⓢ**Accuracy Loss or Retraining** |
| *MACs/DNN* | # Weights | Prune/Reshape Weights | Ⓦ Eliminated/Changed Weights | Ⓦ**Accuracy Loss or Retraining** |
| *1/Utilization* | Mapping | Improve Mapping | Flexibility Cost, Utilization ≤ 1 | Limited Benefits |

**Table 2: The Titanium Law of ADC energy and how to reduce ADC energy components. Of the possible consequences, three are ineffective, and three cause accuracy loss or require DNN retraining. Sum-Fidelity-Limited architectures choose Ⓢ marked cells and Weight-Count-Limited architectures choose Ⓦ marked cells.**

| Architecture | High-Cost ADC | Limits Weight Count | Fidelity Loss | Needs DNN Retraining |
|---|---|---|---|---|
| ISAAC [54] | **Yes** | - | - | **No** |
| AtomLayer [47] | **Yes** | - | - | **No** |
| FORMS [80] | **No** | **Yes** | - | **Yes** |
| SRE [75] | **No** | **Yes** | - | **Yes** |
| ASBP [48] | **No** | **Yes** | - | **Yes** |
| TIMELY [24] | **No** | - | **High** | **Yes** |
| PRIME [5] | **No** | - | **High** | **Yes** |
| RAELLA | **No** | - | **Low** | **No** |

**Table 3: Comparison to prior works. Previous approaches pay high ADC costs or use strategies that cause DNN accuracy loss, requiring retraining to recover.**

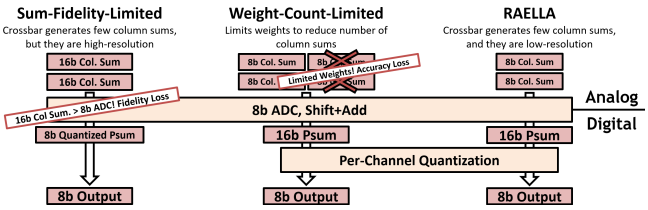


**Figure 2: Loss-causing architectures alongside RAELLA. Although they decrease *Converts/MAC*, Sum-Fidelity-Limited architectures lose fidelity at the ADCs and force hardware-restricted quantization. Weight-Count-Limited architectures limit DNN weights. RAELLA's arithmetic and slicing strategies maintain high fidelity with low *Converts/MAC*.**

high-accuracy quantization strategies to using only the subset of bits that the hardware calculated, rather than the full 16b. This hardware-enforced limitation can cause accuracy loss.

## 2.6 Motivation

Prior works combat accuracy loss by retraining DNNs. FORMS [80], a Weight-Count-Limited architecture, achieves a 2.0× *MACs/DNN* reduction on ResNet18 by pruning and retraining. TIMELY [24], a Sum-Fidelity-Limited architecture, achieves up to a 512× *Converts/MAC* reduction over [54] by using large crossbars and many bits per input/weight slice. However, TIMELY also loses 16b of fidelity from each column sum and recovers accuracy with DNN requantization and retraining. Table 3 shows a gap in recent PIM works: some PIM architectures are inefficient and do not reduce high ADC costs, while others that reduce ADC costs cause DNN accuracy loss and retrain to compensate.

Retraining DNNs can be a challenge due to high computational cost [43], cumbersome hyperparameter tuning [19], and the potential lack of access to training datasets [50, 62]. Additionally, highly efficient DNNs such as highly-reduced-precision models [6, 9, 12] often depend on their own training/quantization procedures. If an architecture requires different training/quantization procedures, it may be difficult or impossible to run these cutting-edge DNNs. The motivation behind RAELLA is to deliver efficient inference and avoid accuracy loss without retraining or modifying DNNs.

# 3 RAELLA: LOW RESOLUTION, HIGH FIDELITY

To be efficient, we would like to reduce ADC resolution. But if column sum resolution is greater than ADC resolution, we lose fidelity and DNN accuracy. We identify three architectural tradeoffs that create high-resolution column sums:

- More sliced products per ADC convert → fewer ADC converts, higher-resolution column sums.
- More bits per weight slice → fewer weight columns, fewer ADC converts, higher-resolution column sums.
- More bits per input slice → fewer input cycles, fewer ADC converts, higher-resolution column sums.

Here, we give an overview of RAELLA's strategies targeting these three tradeoffs. We start with a baseline that uses a $512 \times 512$ crossbar and 4b input/weight slices. Shown in Fig. 3, this setup will produce a very wide distribution of column sums that range from zero to tens of thousands. It requires 17b to represent these column sums. RAELLA's strategies tighten the column sum distribution until it can be represented with a signed 7b range of $[-64, 64)$.

To capture this 7b range without losing fidelity, we set RAELLA's ADC to always capture the seven least-significant bits (LSBs) of column sums. That is, if a single crossbar row is on and produces a sliced product of one, the ADC will read the column sum and output the value one. This small step size preserves full fidelity for in-range column sums.[4] The drawback is that a small step size means

[4]This strategy contrasts with the approach of many Sum-Fidelity-Limited architectures, which drop LSBs from computations [5, 7, 24] While dropping LSBs permits a lower saturation chance, it also necessarily loses fidelity in every psum.

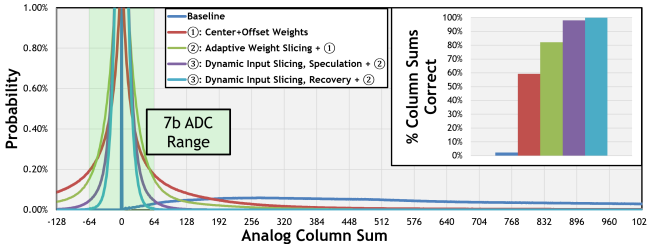


**Figure 3: Column sum distribution with each of RAELLA's strategies while running ResNet18 on ImageNet. RAELLA reduces column sum resolution from 17b to 7b and reduces ADC saturation rate from 98% to 0.1%.**

a small range; the ADC saturates and loses fidelity if the column sum is outside $[-64, 64)$. With 4b input/weight slices, even a single crossbar row can produce sliced products up to $(2^4 - 1)^2 = 225$, which would be saturated at 63. RAELLA must avoid saturation while summing 512 rows at once.

By reshaping the column sum distribution, RAELLA's strategies reduce the probability of saturation to near-zero. This is how RAELLA achieves high fidelity with a low-resolution ADC: RAELLA's ADC only loses fidelity if column sums are large, and the following strategies make column sums small. For each strategy, we report the ADC saturation rate running ResNet18 on ImageNet.

## 3.1 Center+Offset Weights

*3.1.1 Problem.* Standard ReRAM crossbars compute unsigned sliced products. If each sliced product is $\geq 0$, then accumulating many sliced products will generate large-valued, high-resolution column sums.

*3.1.2 Solution.* We shift weights by a center value such that approximately half of the weights are above the center and half are below. As a result, when we slice weights and compute with them in a crossbar, approximately 50/50% of the sliced products come from positive/negative weights. We then sum the signed sliced products in-crossbar. Positive and negative sliced products negate, yielding small-valued column sums even as many sliced products are accumulated. To maximize the beneficial negation that occurs, Center+Offset chooses centers that balance the magnitude of positive/negative slices in each crossbar column.

*3.1.3 Tradeoff.* RAELLA trades off higher crossbar area to implement signed arithmetic in-crossbar. Crossbars are dense, so the area tradeoff is worthwhile to reduce ADC cost. RAELLA also uses additional storage and low-cost digital circuitry to store and process center values.

*3.1.4 Result.* With Center+Offset weights, the column sum distribution labeled ① in Fig. 3 is signed and centered around zero. Column sum resolution is $\leq$ 7b 59.2% of the time.

## 3.2 Adaptive Weight Slicing

*3.2.1 Problem.* More bits per weight slice increase the values stored in weight slices, raising column sum values and resolutions.

*3.2.2 Solution.* Shown in Fig. 7, RAELLA adaptively slices weights at compilation time. We can reduce the average values stored in weight slices and reduce column sum resolution by using fewer bits in each weight slice. However, additional weight slices increase the storage footprint and number of ADC converts by increasing the number of columns, so we would like to minimize the number of slices used. During compilation, we measure errors caused by fidelity loss. We choose the number of bits in each weight slice to control errors and minimize the number of slices used. RAELLA can use a different number of bits for each slice, but all weights in a layer use the same slicing.

*3.2.3 Tradeoff.* RAELLA trades off storage density, ADC converts, and compilation-time preprocessing. ReRAMs and ADC converts needed increase with number of weight slices. RAELLA uses a simple preprocessing strategy and reuses DNN weights for many inferences to minimize preprocessing costs.

*3.2.4 Result.* With Adaptive Weight Slicing, the column sums labeled ② in Fig. 3 are more tightly distributed. Column sum resolution is $\leq$ 7b 82.1% of the time.

## 3.3 Dynamic Input Slicing

*3.3.1 Problem.* More bits per input slice increase the values of input slices, raising column sum values and resolutions.

*3.3.2 Solution.* Shown in Fig. 9, RAELLA dynamically slices the inputs at runtime. RAELLA can use fewer bits per input slice to reduce column sums. However, this requires more cycles and more ADC converts. RAELLA uses a dynamic strategy by speculating with an efficient approach of more bits per input slice. RAELLA recovers from large-column-sum saturation errors by using fewer bits per input slice. This approach achieves high efficiency from speculation and high fidelity from recovery.

*3.3.3 Tradeoff.* RAELLA trades off throughput and crossbar energy. While typically speculation is used to increase speed, RAELLA's speculation trades off speed to gain efficiency. Extra cycles are needed to run both speculation and recovery. Additionally, RAELLA's crossbars consume energy for both speculation and recovery. As crossbars are high-throughput and efficient, it is worth the cost to reduce the ADC overhead.

*3.3.4 Result.* With Dynamic Input Slicing, speculation and recovery column sum distributions labeled ③ in Fig. 3 are further tightened. In speculation and recovery cycles, column sum resolution is $\leq$ 7b 98.0% and 99.9% of the time, respectively.

## 3.4 Accepting Fidelity Loss

*3.4.1 Problem.* With all of RAELLA's optimizations, the column sum resolution can still be greater than ADC resolution. We use a 7b ADC and produce >7b column sums 0.1% of the time. These cause the ADC output to saturate at its min/max of -64/63 and propagate incorrect values to the psum.

*3.4.2 Solution.* DNNs are inherently noise-tolerant [21, 46] so a low error rate is acceptable. Table 4 shows that RAELLA's fidelity errors cause low loss for a variety of DNNs.

RAELLA uses a 512-row crossbar and a 7b ADC. Even with minimal 1b input and 1b weight slices, column sum resolution may be 9b, so it is impossible to guarantee perfect fidelity. With minimal weight slice sizing, RAELLA reduces ADC-related fidelity errors to a rate on the order of one error in ten million psums, or one incorrect psum per ResNet50 [16] inference.

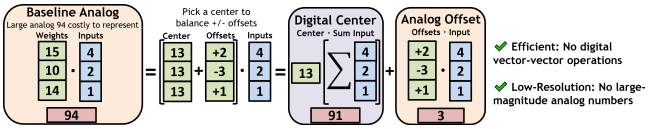


**Figure 4: Center+Offset weights. Standard dot products create high-resolution values which are difficult to represent in analog. Center+Offset digitally subtracts a center from weights, computing with near-zero-average offsets in-crossbar.**

# 4 IMPLEMENTING RAELLA'S STRATEGIES

## 4.1 Implementing Center+Offset Weights

Shown in Fig. 4, we represent DNN weights as a center value plus or minus a small offset. We select centers to make positive and negative weight slices approximately the same magnitude for each column. RAELLA computes signed analog arithmetic, and sliced products from the magnitude-balanced positive/negative weight slices negate to produce near-zero column sums. This allows RAELLA to keep small column sums while accumulating sliced products from many crossbar rows. Meanwhile, RAELLA efficiently processes high-resolution centers in the digital domain.

We first discuss why Center+Offset is important for balancing positive/negative weight slices, then show how RAELLA computes arithmetic with Center+Offset encoding and describe how we calculate optimal center values. Finally, we show the hardware for computing dot products with Center+Offset encoded weights.

*4.1.1 Why Balance Slices.* While DNN weight distributions are commonly cited as zero-mean [14, 73], a zero mean for all weight values over a DNN does not necessarily mean any given weight filter is zero-mean, nor that column sums are zero-mean. For that, we need each individual crossbar column to have weight slices with a zero mean. This is often not the case, as individual weight filters and columns of slices randomly converge to different distributions.[5]

A growing body of works are exploring differential encoding, which, like Center+Offset, computes signed analog arithmetic [3, 14, 28, 32, 67, 73, 81]. Differential encoding uses positive slices to represent positive weights and negative slices to represent negative weights; it can benefit from Center+Offset to balance positive/negative weight slices and reduce column sum resolution.

Center+Offset can be especially beneficial for filters where weight slice distributions have noticeable nonzero averages. This can occur in filters where there is a greater number of negative than positive weights, such as the filter shown in Fig. 5. Differential encodings represent these mostly-negative weights with mostly-negative slices, yielding a negative average for the slices in each column. After dot products with hundreds of slices, even slight negative averages can accumulate to create large negative column sums. This effect can significantly increase ADC saturation and cause DNN accuracy loss, as shown in Table 4. By balancing positive/negative slices, Center+Offset reduces per-column biases and protects from accuracy loss.

[5]When we say "filter" we mean a set of weights from one dot product that fit in one crossbar. An output channel of a DNN layer (or "filter" in the traditional sense) may be partitioned over multiple crossbars if its weights do not fit in a crossbar. The important aspect is that each crossbar column produces a unique column sum distribution, regardless of the characteristics of the overall DNN. To account for this, Center+Offset attempts to balance positive/negative weight slices in each column.

*4.1.2 Center+Offset Arithmetic.* Given a weight $w$ and center $\phi$, we calculate positive offset $w_+ = max(w - \phi, 0)$ and negative offset $w_- = max(\phi - w, 0)$. For weights above the center, $w_+$ is the difference between the weight and the center while $w_-$ is zero. The converse is true for weights below the center. Given weight filter $W$ programmed as positive/negative offset vectors $W_+$, $W_-$, RAELLA computes a dot product with input vector $I$ as:

$$W \cdot I = \left(\phi \sum I\right) + (W_+ - W_-)I \tag{1}$$

RAELLA computes Eq. 1 at runtime, with $\phi \sum I$ computed digitally and $(W_+ - W_-)I$ in analog.

*4.1.3 Calculating Optimal Centers.* We calculate centers/offsets with one-time preprocessing before programming RAELLA. We calculate a center for each weight filter independently, as weight distributions and optimal centers vary for different weight filters.

We define an optimization problem to solve for the center value $\phi$. First, we define a slice $S$ as a sequence of inclusive bit indices $[h \ldots l]$ from the most to least significant index $h$ to $l$ (*e.g.*, slice $[7 \ldots 4]$ contains the four most significant bits). Then, we define a slicing function $D(h, l, x)$ that crops signed number $x$ to contain the bits from indices $h$ to $l$ (shifted so bit $l$ is the least significant position), preserving the sign. Given a weight filter $W$ and slices $S_{i \in \{1,2,\ldots,N\}} = [h_i \ldots l_i]$, we solve for the center $\phi$ of $W$ as follows:

$$\underset{\phi \in \{1,2,\ldots,255\}}{\arg\min} \sum_{i=1}^{N} 2^{l_i} \left( \sum_{w \in W} D(h_i, l_i, w - \phi) \right)^4, \tag{2}$$

where $N$ is the total number of slices, and $w$ is a weight in $W$. Eq. (2) balances positive/negative values in each column of weight slices, assigning higher costs for columns with larger nonzero sums. The inner sum $(\sum_{w \in W} D(h_i, l_i, w - \phi))^4$ calculates the cost for a single column, equal to the sum of weight slices in the column raised to the power of four. Four is chosen empirically; we find that too low a power does not sufficiently penalize large sums, while too high a power overvalues the largest-sum column and fails to consider all columns. The outer sum $\sum_{i=1}^{N} 2^{l_i}(\ldots)$ weights cost by bit position (*e.g.*, the most significant bit in an 8b number has a magnitude of $2^7$ and the cost of a 1b slice containing only this bit would also be scaled by $2^7$) and sums costs for all columns. Costs are weighted by bit position as saturations in higher-order slices have a greater impact on the psum.

We calculate centers for each weight filter (*i.e.*, a single dot product) in the crossbar independently. A coarser granularity, such as a single center for a full crossbar (*i.e.*, 100+ different filters), would not be as effective, as different DNN filters can have different weight distributions and require different centers.

RAELLA's per-filter centers have the drawback that each center balances multiple columns for one filter, and therefore may not be optimal for any one column. Ideally, per-column centers would be able to precisely zero the average weight slice value for each column. However, this approach is limited by integer precision centers. Consider the case where a column of slices has an average value of 0.4. We could shift all weight slices in the column by $-1$, but this would worsen the average by shifting it to $-0.6$. Instead, we shift full-precision weight values before slicing, which can reshape (rather than shift) the value distribution for each individual slice.

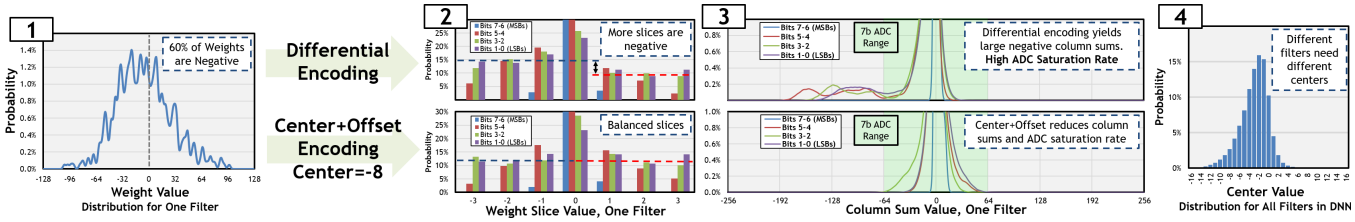


**Figure 5: Differential vs. Center+Offset Encoding. Distributions for an InceptionV3 [61] filter with negative-average weight slices is shown for illustrative purposes. 8b weights / inputs are sliced into four 2b / eight 1b slices. 1 Most of the weights in a filter are negative. 2 Differential encoding represents negative weights with negative slices, yielding mostly-negative slices for the filter. Center+Offset balances positive/negative slices. 3 Dot products with mostly-negative slices yield large negative column sums that cause ADC saturation. Center+Offset reduces column sums. 4 Each DNN filter needs a different center.**

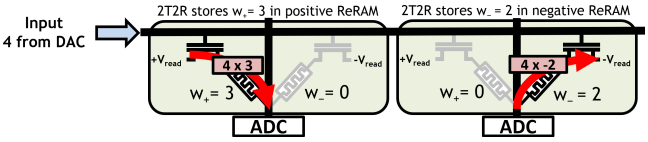


**Figure 6: 2T2R devices compute signed arithmetic in-crossbar. Red shows the magnitude and direction of the current flow. Grayed-out devices are off (set to the high-resistance state).**

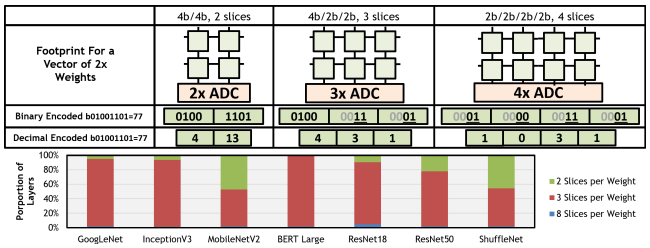


**Figure 7: (Top) Weight Slice Crossbar Footprints. (Bottom) DNN Per-Layer Weight Slicings. Increasing slice count lowers column sums and saturation chance, but increases *Converts/MAC*. Most layers use three slices per weight.**

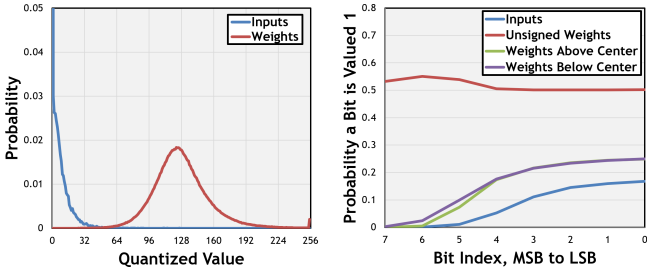


**Figure 8: (Left) DNN input/weight value distributions without slicing and (Right) per-bit densities. The second-to-last layer of ResNet50 is shown, representing a typical DNN layer. Bell-curve-distributed weights can be split about a center into two similar distributions with sparse high-order bits. Unsigned inputs have naturally-sparse high-order bits.**

This distribution reshaping can make smaller adjustments to the average weight slice value in each column.

*4.1.4 Center+Offset In Hardware.* Computing psums with Eq. (1) requires two terms. The first term, the input sum $\phi \sum I$, is calculated digitally. Crossbar columns share input vectors, so the sum calculation is amortized across columns.

The second term is the vector-vector multiplication with offsets. $(W_+ - W_-)I$ is calculated in analog by the crossbar. RAELLA uses 2T2R devices, shown in Fig. 6, to realize analog subtraction in-crossbar.[6] 2T2R, with two ReRAMs (2R) per weight accessed via two access transistors (2T), have been explored as a method to represent signed weights [3, 27, 28, 67, 74]. One ReRAM device is connected to a positive source and the other a negative source, letting 2T2Rs add to or subtract from column sums. For each weight, we program positive/negative offsets $w_+/w_-$ into the two ReRAMs. As one offset is zero for any given weight, one ReRAM device is used in each pair. Added ReRAMs and access transistors increase RAELLA's crossbar size, but crossbars are small, and the increase in system area is only $\sim$ 10%.

## 4.2 Implementing Adaptive Weight Slicing

Adaptive Weight Slicing minimizes the weight slices used for each DNN layer. It uses as many bits as possible in each slice without excess fidelity loss. Fig. 7 shows various slicings available to RAELLA. More bits per slice means fewer slices per weight, denser storage, and fewer ADC converts, but more bits also increase the values stored in each weight slice and raise the chance of high-resolution column sums. RAELLA can use a different number of bits for each slice, but all weights in a layer use the same slicing.

The bit density, or probability that a given bit is 1, affects the values in weight slices. Fig. 8 shows per-bit densities for DNN inputs and weights in a typical DNN layer. Input values generally follow right-skewed distributions, yielding sparse high-order bits. Weight values generally follow rough bell curves. When represented with Center+Offset encoding, this also yields sparse high-order bits. Due to sparsity in the high-order weight bits, in most layers, 4b weight slices can store the highest-order 4b of weights with low values and low column sums. Low-order weight bits are denser and usually require a lower 2b per weight slice. Fig. 7 shows the per-layer slicings of DNNs. Most layers use the 4b-2b-2b setup with three weight slices: one weight slice for the highest-order four bits and two weight slices storing two low-order bits each.

*4.2.1 Error Budgets.* We could choose weight slices to minimize saturation, but this is too conservative. DNNs can tolerate some

[6] Analog subtraction can also be done with circuits [20] 1T2R [81], and SRAMs [40, 78].

error, so we would like to allow a small amount of saturation. Unfortunately, it is difficult to predict how slicing impacts saturation and how saturation affects error in DNN layer outputs. Too-large column sums cause saturation, and column sums are affected by input/weight distributions, input/weight slice distributions, correlations between distributions, the number of weights per filter, and random variance. Furthermore, 16b psums are digitally quantized into 8b outputs [82], which may magnify or shrink the error.

To capture all these factors, we take an empirical approach. We define the *error budget* as the average magnitude error allowed for nonzero outputs of a layer after outputs are fully computed and quantized to 8b. Only nonzero outputs are counted when calculating average magnitude error to give a more consistent calculation for layers with varying output sparsity.[7]

To calculate error, we use ten inputs chosen randomly from the validation set. It suffices to use so few inputs because changing slicings may change the error by an order of magnitude or more, and these differences are easily detected.[8] The order-of-magnitude differences stem from the shape of the column sum distribution (Fig. 3). The distribution tails shrink exponentially, so changes in the distribution width (*i.e.*, due to slicings) have an exponential effect on the saturation rate.

The error budget is set to 0.09 in our tests, corresponding to one in eleven 8b outputs being off by one on average. After quantization, the errors created by ADC saturation are generally small and cause a low accuracy loss, shown in Table 4.

*4.2.2 Choosing Weight Slices.* Weight slices are calculated with the preprocessing procedure shown in Algorithm 1. Preprocessing occurs once when compiling a DNN for RAELLA, taking 10-1000ms per layer on an Nvidia RTX 2060 GPU. After preprocessing, sliced+encoded weights are programmed to crossbars for use with any number of inferences.

For an M-bit weight and up to N bits per ReRAM, we define a *slicing* as a tuple of integers $1 \leq s_0..s_j \leq N$ such that $\sum s_i = M$. For 8b weights, $\leq$ 4b per ReRAM, slicings include (4b,4b), (2b,1b,1b,4b), and (1b,2b,2b,3b). There are 108 slicings in total.

To find the best slicing for a DNN layer, we iterate through all 108 slicings. For each, we Center+Offset encode weights following Section 4.1, simulate the crossbar with ten test inputs, and record error. We choose the slicing that uses the fewest slices and has below-budget error. For slicings with the same number of weight slices, the lower-error slicing is chosen.

We use 1b input slices when comparing weight slicings. We always use the most conservative 1b per weight slice for the last layer. The last layer has an outsized effect on DNN accuracy [6] and a less efficient last-layer slicing has little effect on overall energy/throughput as intermediate layers dominate DNNs (Fig. 7).

*4.2.3 Adaptive Weight Slicing in Hardware.* Given 4b ReRAMs, each can be programmed with $2^4 - 1$ analog levels. To program 3b or 2b slices, we use the lowest $2^3 - 1$ or $2^2 - 1$ levels. Given a 3b weight slice XXX, this corresponds to programming a device with 0XXX. This is only a restriction of the available range and therefore does not require a change to ReRAMs. Crossbars already need shift+add circuits to add column sums across weight and input slices; adaptive slicing requires only changing the shift+add pattern.

The main overhead depends on the number of weight slices. Each additional weight slice increases required ReRAMs and ADC converts. RAELLA can use between two weight slices (4b/slice, most efficient) and eight weight slices (1b/slice, least efficient). Most layers use three weight slices.

[7] This is important when ReLU is folded into quantization. If ReLU zeros an output, it will likely zero any error associated with that output as well. If ReLU zeros many outputs, then average error is lowered while error per nonzero output remains consistent.
[8] In fact, this algorithm usually picks the same slicings when testing just one input. If we test with Gaussian noise as input, then slicings match for > 90% of layers.

**Algorithm 1: Preprocessing Weight Slicing and Centers**

```
Func SliceEncodeWeights(layer, testInputs, errorBudget)
    /* DNN layer preprocessing. Requires a layer (shape,
       quantization, weights), test inputs (activations
       from ten images/tokens in this paper), and a
       scalar error budget (0.09 in this paper).      */
    slicing = FindBestSlicing(layer, testInputs, errorBudget)
    centers = FindOptimalCenters(layer, slicing)
    return slicing, centers

Func FindBestSlicing(layer, testInputs, errorBudget)
    /* Implementation of Adaptive Weight Slicing from
       Sec. 4.2. 10-1000ms per layer.                 */
    expectedOutputs = layer.Run(testInputs)
    possibleSlicings = GetAllPossibleSlicings()
    bestSlicing = possibleSlicings[0]
    bestNSlices = CountSlices(bestSlicing)
    bestError = ∞
    for slicing ∈ possibleSlicings do
        centers = FindOptimalCenters(layer, slicing)
        outputs = layer.SimulateCrossbar(testInputs, slicing,
          centers)
        errors = |expectedOutputs - outputs|
        meanError = Mean(errors[expectedOutputs != 0])
        nSlices = CountSlices(slicing)
        betterSlicing = nSlices < bestNSlices ||
          (nSlices==bestNSlices && meanError < bestError)
        if meanError < errorBudget && betterSlicing then
            bestSlicing = slicing
            bestNSlices = nSlices
            bestError = meanError
    return bestSlicing

Func FindOptimalCenters(layer, slicing)
    /* Solve Center+Offset Eq. (2). <1ms per layer.
       Returns a center for each weight filter.       */
    centers = SolveOptimizationProblem(layer, slicing)
    return centers
```

## 4.3 Implementing Dynamic Input Slicing

Dynamic Input Slicing balances high-efficiency more-bit input slices and high-fidelity fewer-bit input slices. We would like to minimize the input slices and thus ADC converts, while also avoiding fidelity loss due to high-resolution column sums. Unlike with weights, the input slicing can be changed at runtime. This allows us to speculate with an efficient, aggressive slicing and recover with a conservative slicing. In speculation, RAELLA uses three input slices, which has high efficiency but a higher chance of creating

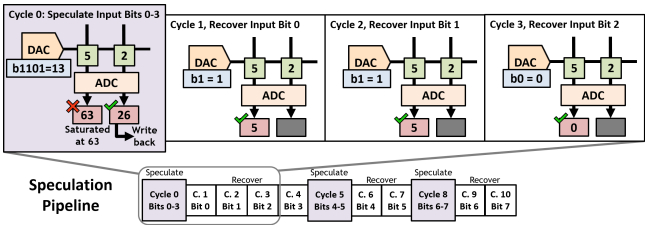


**Figure 9: Speculative Computation. Speculative cycles use 2–4 bits per input slice, with fewer ADC converts per input but a higher saturation chance. Recovery cycles use 1-bit input slices and ADCs only process columns that failed speculation.**

large, high-resolution column sums. In recovery, RAELLA uses the most conservative eight 1b input slices.

The procedure for speculation and recovery is shown in Fig. 9. First, a 4b high-order slice is speculatively fed to the crossbar, and column sums are converted by ADCs. If a column sum is too large, it will saturate at the ADC bounds of $[-64, 64)$. If an ADC output equals either of these bounds, an error is detected and marked as a speculation failure. Next, after all columns are processed, the 4b input slice is resliced into 1b slices and processed again over four recovery cycles. To save energy in recovery, ADCs are power-gated for columns that speculated successfully. In the rare event that an ADC saturates in recovery, we accept fidelity loss and propagate the saturated value. After the four recovery slices are processed, the process repeats for the following speculation and recovery cycles.

*4.3.1 Dynamic Input Slicing In Hardware.* Given a 4b DAC, analog input slices can take one of $2^4 - 1$ analog levels. For an N-bit input slice, we can use the lowest $2^N - 1$ levels. Given a 3b input slice XXX, this corresponds to converting 0XXX. As this is only a restriction of the available range, it does not require changing the DAC hardware.

To track successful/failed speculations, RAELLA stores speculation success flags in a buffer for each crossbar. In recovery, ADCs only convert column sums that failed speculation.

The entire ReRAM crossbar is one unit, so all columns speculate and recover together. As it is highly likely that at least one column will fail speculation, crossbars always run recovery. Therefore, RAELLA's speculation saves energy at the cost of speed (unlike the common use of speculation for speed, *e.g.*, CPU branch prediction).

Speculation also increases crossbar energy, as all columns and ReRAMs run both speculation and recovery cycles. Recovery cycles consume less energy than speculation cycles, as ReRAM devices use less energy with smaller input values [29] and ADCs only process a small fraction of columns in recovery cycles.

*4.3.2 Dynamic Input Slicing System Effects.* RAELLA can run without speculation, processing eight recovery slices alone. With this approach, each column would require eight ADC converts for all eight input slices. With speculation, three ADC converts are needed instead to process three 2-4b speculative input slices. Across our baselines, speculation fails approximately 2% of the time, requiring 2-4 recovery converts depending on which speculative slice failed. Overall, speculation succeeds ~ 98% of the time and reduces ADC converts by ~ 60% over a recovery-only approach. An average of three speculative converts + 0.3 recovery converts are required to process each column.

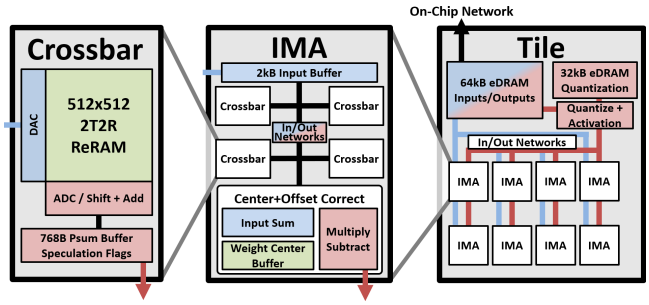


**Figure 10: The RAELLA Architecture. (1) The base unit is a crossbar. (2) Four crossbars make up an IMA. (3) Eight IMAs make up a tile. Components are colored blue for input storage/processing, green for weights, and red for outputs.**

While RAELLA saves ADC converts with speculation, it trades off throughput and crossbar energy. RAELLA's crossbars require eleven cycles to run all three speculation + eight recovery slices. Alternatively, a no-speculation approach could run only the eight recovery slices, increasing throughput but also increasing the number of ADC converts required.

## 5 RAELLA ARCHITECTURE AND PIPELINE

The high-level RAELLA architecture is shown in Fig. 10. RAELLA's organization mostly follows that of ISAAC [54]. We describe the RAELLA architecture from the bottom up, show RAELLA's dataflow, then describe how RAELLA reduces analog nonidealities.

### 5.1 Crossbar

Crossbars consist of $512 \times 512$ 2T2Rs. Each crossbar is programmed with weights from one DNN layer, and each weight filter uses 2-8 crossbar columns based on the DNN layer slicing (Section 4.1).

To process inputs, we use 4b pulse-train DACs for their simple hardware [32] and superior linearity [55]. Pulse-train DACs encode an N-bit input slice with a number of pulses up to $2^N - 1$. The DAC consists of a simple row driver to apply input pulses, a 1b flip-flop to store the current input bit, and an AND gate acting as an enable signal [32]. To output a 4b value, the most significant bit is first loaded into the flip-flop and a global clock generates eight 1ns pulses. The DAC outputs the AND'ed value of the clock and its stored value, equal to eight pulses if the bit is on and zero otherwise. Subsequent bits are loaded sequentially and run for four, two, and one pulse(s), respectively.

DACs activate the 2T2R access transistors and each 2T2R device computes a sliced product that it adds or subtracts from the column sum. Column sums appear as a current on a column, which is buffered and scaled by a current buffer [24] before being captured as capacitor voltages and held by sample+hold circuits [38].

Next, four 7b ADCs [23] convert the 512 column sums in 100ns [54]. 7b signed ADC results are summed by shift+add circuits and accumulated in 16b psum buffers [82].

With the most-dense slicing of two slices per weight, one crossbar may produce up to 256 psums, which are stored in a 256-entry psum buffer. Each entry stores a 16b psum + 8b success flags, for a 768B psum buffer total capacity.

In speculation/recovery cycles, inputs are streamed to crossbars once for each cycle. In speculation, ADCs process all columns. If an ADC saturates, the psum is not updated and the success flag is marked. In recovery, all success flags are checked. ADCs process and write results only for columns that failed speculation.

The crossbar cycle is pipelined in two stages [54]. In the first stage, the DACs supply input pulses, the crossbar computes analog column sums, and the results are latched in sample+hold circuits. 4b pulse train DACs with 1ns/1ns on/off pulse width take 30ns to send up to 15 input pulses. Crossbars settle and produce column sums in less than a nanosecond [17]. In the second stage, ADCs convert sample+hold results in 100ns [54]. The overall crossbar cycle time is 100ns from the slower-stage ADC processing.

RAELLA utilizes input bit sparsity to reduce column sum values and crossbar energy, benefiting from the high bit sparsity of unsigned inputs (Fig. 8). If inputs are signed, RAELLA processes positive/negative inputs in two separate cycles to generate sparsity.

## 5.2 In-Situ Multiply Accumulate

Four crossbars are organized into an In-Situ Multiply Accumulate (IMA) with an input buffer [54]. An input network sends input vectors to crossbars, and if all inputs are shared between two crossbars, the input vector is multicast. To exploit temporal input reuse [47, 59], the input buffer stores reused inputs between crossbar cycles. The four crossbars can process up to $4 \times 512 = 2048$ inputs across all rows, so the buffer is sized 2kB.

To support Center+Offset weights, each IMA includes a weight center buffer and digital addition circuitry to calculate input sums. A running sum is kept for each crossbar. To exploit input reuse [47], we add inputs to the sum when they are first used in crossbar columns and subtract when they are last used. If different crossbar columns use different subsets of the inputs, RAELLA adds/subtracts inputs in a streaming fashion while processing columns.

## 5.3 Tile

Eight IMAs are organized into a tile. Each tile includes a 64kB eDRAM buffer [54] storing 8b inputs/outputs, digital maxpool units, and quantization circuits. RAELLA digitally computes 8b per-channel quantization [82], allocating 32b per output channel to store a FP16 quantization scale and bias [82], or 32kB per tile.

## 5.4 Accelerator & Programming

Like ISAAC [54], every four tiles share a router enabling on-chip communication. When a tile completes a set of outputs, it sends data to the next tile via its router. If a layer has more weights than a tile can store, its weights are split across multiple tiles.

Like other PIM accelerators [24, 47, 54, 80], RAELLA is programmed once for many inferences to mitigate high ReRAM write energy [45]. When compiling a configuration for RAELLA, we use lightweight preprocessing for Center+Offset and Adaptive Weight Slicing, as discussed in Section 4.2.2.

## 5.5 DNN Dataflow

Each DNN layer is mapped to one crossbar if it fits. Otherwise, it will spill over to more crossbars, IMAs, and tiles. RAELLA's interlayer dataflow follows ISAAC's [54] to minimize eDRAM footprint and inter-tile communication requirements. Fig. 11 shows RAELLA's

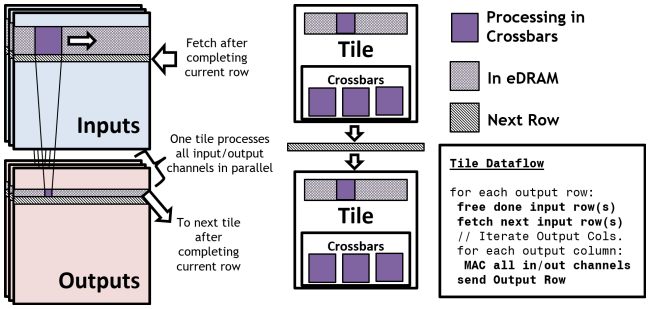


**Figure 11: Dataflow. One row of outputs for a layer are computed at a time. Tiles receive/send inputs/outputs once.**

dataflow. DNN layers are run in a pipeline across parallel tiles. Tiles generate one row of a layer's output tensor at a time, reusing previously-used input rows and fetching only new input rows. As a tile produces rows of the output tensor from top to bottom, input rows are consumed from the previous tile in the same order. Communication and data reuse patterns are coordinated by pattern generators and fixed at program time. Below the tile level, Timeloop [41] is used to find optimal data reuse patterns.

RAELLA replicates weights to increase throughput following previous work [24, 54, 56]. If there is space, weights are replicated in-crossbar to compute multiple convolution steps using a partial Toeplitz expansion [11, 24]. Weights can be further replicated across crossbars, IMAs, or tiles. Replication follows a greedy scheme: while there are tiles left, the lowest-throughput layer is replicated.

## 5.6 RAELLA Reduces Analog Nonidealities

PIM crossbars can suffer from nonidealities such as IR drop and sneak current. RAELLA reduces these relative to ISAAC.

High current traversing long crossbar columns causes IR drop, which can cause accuracy loss [7, 75]. Positive/negative 2T2R devices consume current from their neighbors, reducing IR drop [28, 81]. Furthermore, RAELLA's ADC saturates at 64, or fewer than five ReRAMs in the highest-conductance state. Therefore, RAELLA's columns must only tolerate current from five ReRAMs, compared to an ISAAC-like design that sums current for 128 ReRAMs.

Sneak current, or leakage through off ReRAMs, can cause accuracy loss [7]. Sneak current is zero in 2T2R crossbars as the leakages from positive and negative ReRAMs negate [81].

# 6 EVALUATION

RAELLA is compared to accelerators ISAAC [54], FORMS-8 [80], and TIMELY [24]. ISAAC does not require DNN retraining. FORMS is Weight-Count-Limited and TIMELY is Sum-Fidelity-Limited, so both retrain to recover DNN accuracy.

First, we show the efficiency and throughput gain of RAELLA in a non-retraining setting by comparing RAELLA's energy and throughput with those of ISAAC. We show that RAELLA achieves high throughput and efficiency without changing the DNN models.

Next, we show competitiveness with DNN-retraining architectures by comparing RAELLA to FORMS and TIMELY. We show that RAELLA matches the efficiency/performance of these architectures without needing to retrain.

Then, we show RAELLA's low accuracy loss and compare to FORMS and TIMELY. We also show the accuracy benefits of RAELLA's Center+Offset encoding.

## 6.1 Methodology

Models of RAELLA, ISAAC, and FORMS are created using Accelergy/Timeloop [41, 71, 72] in the 32nm technology node. The architectures are modified to support 8b DNNs as described in Section 6.1.2. Under a $600mm^2$ area budget, RAELLA fits 743 tiles while ISAAC and FORMS fit 1024 tiles each. Results for TIMELY are from the original paper [24]. To compare to TIMELY, we scale RAELLA to TIMELY's 65nm tech node and use TIMELY's analog components (TDC, IAdder, Charging+Comparator) and ReRAM devices [13] in RAELLA. RAELLA's error budget is set to 0.09 in all tests.

*6.1.1 Component Models.* SRAMs are modeled in CACTI [18]. Models of networks, routers, and eDRAM buffers are from ISAAC [54]. eDRAM refresh is not an issue as tiles consume data faster than a refresh period [63]. RAELLA uses the ADC [23] from ISAAC scaled to 7b following [52]. DAC, input driver, and crossbar area/energy are generated using a modified NeuroSim [2, 44]. 2T2R area is pessimistically estimated as the sum area of two ReRAMs and two min-sized transistors, ignoring potential stacking between chip layers [27, 66]. DACs use a flip-flop and an AND gate for each row to generate pulse trains [32], where each pulse is 1ns and each 4b input slice can comprise up to 15 pulses. ReRAM parameters are taken from TIMELY [5, 24], using a 0.2V read voltage and 1kΩ/20kΩ on/off resistance [13, 17]. Current buffers that capture analog column sums are taken from TIMELY [24]. Outputs are quantized with a multiply/truncate and activation functions are fused into quantization [82]. Maxpool units and sampling capacitors consume negligibly little energy/area [24, 54]. One crossbar cycle is 100ns, and crossbars produce a set of psums every 11 cycles (three speculation input slices + eight recovery input slices) unless bottlenecked by the interlayer dataflow. Latency is doubled for signed inputs as positive/negative inputs are processed in separate cycles. With speculation disabled, crossbars require eight cycles and 800ns to produce a set of psums.

*6.1.2 Models of ISAAC and FORMS.* ISAAC [54] and FORMS [80] models are validated against the results presented in their papers with $< 10\%$ energy and throughput error. After validating, we model ISAAC and FORMS using the same components used in RAELLA for a fair apples-to-apples comparison. In particular, the DAC/crossbars are modeled using a modified NeuroSim [2, 44] which captures the data-dependent energy consumption of analog components. We modify both architectures and add quantization hardware to run 8b DNNs. After scaling to 8b, our ISAAC baseline has $\sim 4\times$ higher efficiency and throughput than the original ISAAC while our FORMS baseline has $\sim 2\times$ higher efficiency and throughput than the original FORMS. For FORMS, we use the highest reported pruning ratio. For a fair comparison, we modify ISAAC to support the partial-Toeplitz mappings [11, 24] that RAELLA supports, which increased the throughput of ISAAC by an additional $1-1.9\times$. These mappings were not beneficial to FORMS.

## 6.2 DNN Models and Test Sets

We test on seven representative DNNs. Six are CNNs from the PyTorch [42] Torchvision [31] quantized library: GoogLeNet [60], InceptionV3 [61], Resnet18 [16], ResNet50 [16], ShuffleNetV2 [30], and MobileNetV2 [53]. ShuffleNetV2 and MobileNetV2 are compact with small filters, while the others are large models. We report accuracy for the ImageNet [10] validation set.

Additionally, we test a Transformer [64] BERT-Large [70] on the Stanford Question Answering Dataset [49] to show RAELLA's effectiveness on cutting-edge Transformers. For BERT-Large, we accelerate the feedforward layers. Other works explore accelerating Transformer attention [39, 58, 77]. BERT-Large shows RAELLA's performance with a non-ReLU activation and signed inputs.

## 6.3 Efficiency And Throughput: No Retraining

RAELLA is evaluated and compared to ISAAC running off-the-shelf models of all DNNs. Fig. 12 shows efficiency and throughput results. RAELLA improves energy efficiency 2.9 to 4.9× (geomean 3.9×). Efficiency gains come mainly from ADC energy reduction. RAELLA uses a 7b ADC, while ISAAC uses an 8b ADC. Furthermore, RAELLA uses larger crossbars, more bits per input slice/weight slice, and speculation to reduce ADC converts by 5 to 15×.

RAELLA's throughput benefits come from large $512\times512$ (versus ISAAC's $128\times128$) and denser 2-4 bits per weight slice (versus ISAAC's 2b per weight slice). Larger and denser weight storage and computation give RAELLA a throughput benefit of 0.7 to 3.3× (geomean 2.0×).

Without speculation, RAELLA runs recovery slices only, reducing relative efficiency benefits to 2.8× geomean due to higher ADC energy. Relative throughput increases to 2.7× geomean as crossbars do not run the three speculation slices, and psums are computed in eight crossbar cycles instead of eleven.

RAELLA is more effective with unsigned inputs and larger DNNs. Positive/negative inputs (*e.g.*, those in BERT) are processed in separate cycles, reducing throughput gains, and small filters in ShuffleNet and MobileNet poorly utilize the large crossbars of RAELLA.

## 6.4 Comparison with Retraining Architectures

RAELLA is compared to TIMELY and FORMS-8. We show geomean ResNet18/ResNet50 results since we have data for these DNNs on all baselines. RAELLA runs off-the-shelf models, while FORMS [80] runs pruned-retrained versions and TIMELY [24] runs requantized-retrained versions.

Fig. 13 compares RAELLA's efficiency/throughput to FORMS and TIMELY. RAELLA is able to match the throughput of FORMS and exceed the efficiency of both FORMS and TIMELY. In the TIMELY comparison, we find that 65nm RAELLA is more efficient without speculation. This is because 65nm-RAELLA uses TIMELY's analog components, including TIMELY's highly efficient ADC. Speculation is useful when ADC costs dominate, but the tradeoffs may not be worthwhile if the ADC is not a major contributor to overall energy.

## 6.5 Accuracy Comparison

We compare RAELLA with three baselines. RAELLA Center+Offset is the standard RAELLA, configured with a 0.09 error budget. To showcase the benefits of Center+Offset, we compare it with

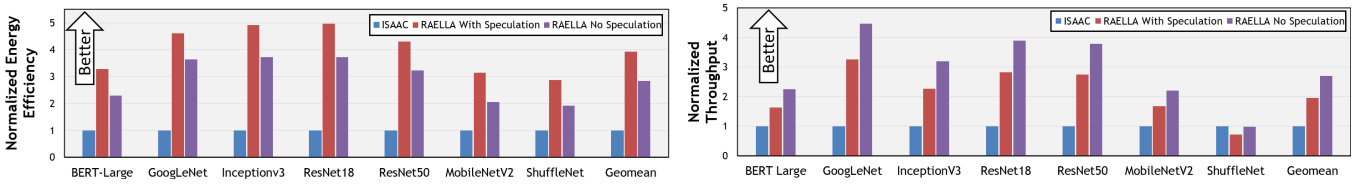


**Figure 12: Efficiency and throughput normalized to the ISAAC architecture. Both architectures run DNNs without retraining. RAELLA with/without speculation increases efficiency 3.9× /2.8× and throughput by 2.0× /2.7× geomean.**

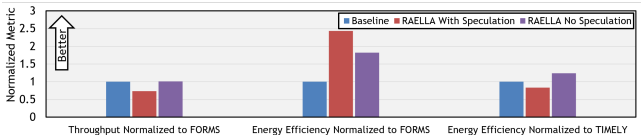


**Figure 13: Comparison with FORMS and TIMELY. FORMS and TIMELY run retrained DNNs. RAELLA offers competitive/superior throughput/efficiency without retraining.**

| | RAELLA Center+Offset | RAELLA Zero+Offset | FORMS [80] | TIMELY [24] |
|---|---|---|---|---|
| **Retrained** | **No** | **No** | **Yes** | **Yes** |
| **Accuracy Drop %. Negative is accuracy gain.** | | | | |
| ResNet18 | 0.06 | 0.16 | 0.62 | ≤ 0.1 |
| ResNet50 | -0.08 | 0.30 | 0.70 | ≤ 0.1 |
| MobileNetV2 | 0.03 | 10.17 | - | - |
| ShuffleNetV2 | 0.14 | 16.36 | - | - |
| GoogLeNet | -0.02 | 1.53 | - | - |
| InceptionV3 | -0.03 | 3.72 | - | - |
| BERT-Large | 0.12 | 0.46 | - | - |

**Table 4: Accuracy Comparison. BERT-Large compares F1 loss, while others compare Imagenet Top-5 loss. Zero+Offset causes high accuracy loss; Center+Offset is essential to preserve accuracy. FORMS and TIMELY retrain, while RAELLA maintains low accuracy loss without retraining.**

RAELLA Zero+Offset, which implements a common-practice differential encoding (described in Section 4.1) by setting centers to zero. We use the same slicings for RAELLA Center+Offset and RAELLA Zero+Offset to match efficiency/throughput. Additionally, we show the reported accuracy of FORMS and TIMELY after retraining.

Table 4 shows the accuracy results. RAELLA with Center+Offest encoding causes little to no accuracy loss. Zero+Offset (differential encoding) causes substantial accuracy degradation due to high ADC saturation rates, as described in Section 4.1. Zero+Offset accuracy drop varies greatly across DNNs due to varying filter weight distributions. TIMELY and FORMS recover from accuracy loss by retraining DNNs.

# 7 ABLATION STUDIES

To isolate the effects of each of RAELLA's strategies, we begin with an ISAAC architecture and apply strategies sequentially. In the energy ablation, we test the efficiency benefits of each of RAELLA's strategies. In the accuracy ablation, we test RAELLA's strategies against increasing analog noise. All test setups maintain high fidelity. The four test setups are the following:

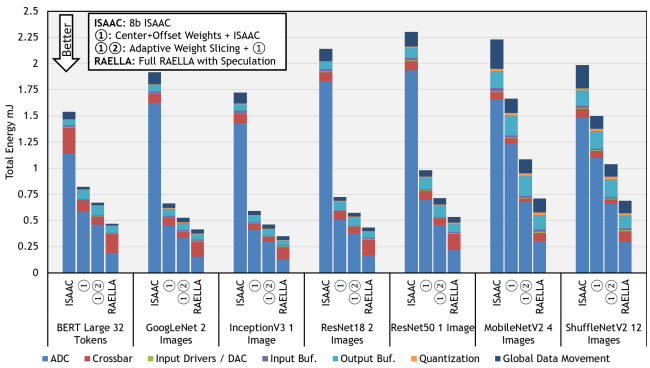


**Figure 14: Energy Ablation. Each of RAELLA's strategies increases PIM architecture efficiency. Batch size is varied across DNNs to keep overall energy in the same range.**

- ISAAC: an 8b ISAAC. 128 × 128 crossbars, unsigned arithmetic. Four 2b weight slices, eight 1b input slices. 8b ADC.
- Center+Offset: previous setup, plus crossbar size increased to 512 × 512 2T2R with Center+Offset arithmetic. ADC resolution is reduced to 7b.
- Center+Offset, Adaptive Weight Slicing: previous setup, plus weight slicings are chosen per-layer following Section 4.2.2. Most layers use three weight slices in a 4b-2b-2b pattern.
- RAELLA: previous setup, plus Dynamic Input Slicing and speculation enabled. RAELLA's registers/networks are added. RAELLA runs a 2-4 bit speculation input slice followed by 2-4 one-bit recovery input slices. In recovery cycles, ADCs do not convert columns where speculation succeeded.

## 7.1 Energy Ablation

Fig. 14 shows the following results:

- ISAAC: ADCs dominate overall energy. *Converts/MAC* is 0.25. Per-component energy breakdown varies depending on DNN input/weight values, crossbar utilization, and digital data movement requirements.
- Center+Offset: enables a 4× scale-up in crossbar rows/columns and reduces ADC resolution. Center+Offset bit sparsity lowers crossbar energy. Large crossbars decrease data movement energy and reduce *Converts/MAC* from 0.25 to 0.063. Digital center processing, which requires one input addition and one multiply/subtract per several hundred MACs, contributes negligible energy.
- Adaptive Weight Slicing: reduces ADC energy ∼ 25% as most layers use three weight slices instead of four. *Converts/MAC* is reduced to 0.047.

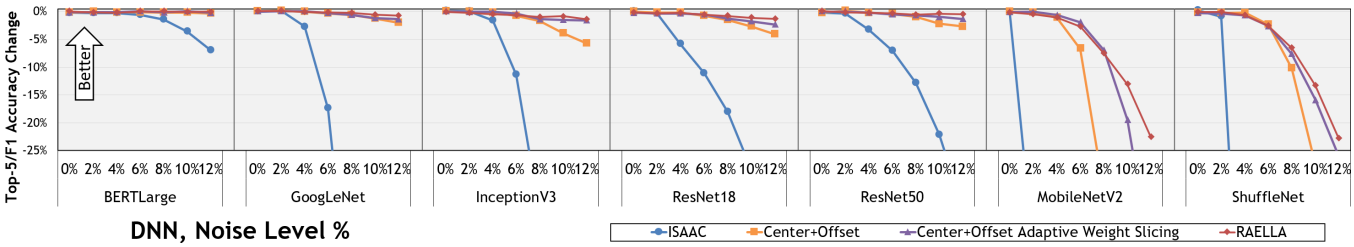


**Figure 15: Accuracy drop at increasing analog noise. Center+Offset and Adaptive Weight Slicing increase noise tolerance. Dynamic Input Slicing maintains accuracy despite speculation failures; recovery prevents accuracy loss.**

- Speculation: reduces ADC energy by 60%. Increases crossbar/DAC energy due to speculation cycles. Increases the input buffer energy due to 2× fetches. Usually decreases output buffer energy due to fewer psum writebacks. *Converts/MAC* is 0.018.

## 7.2 Accuracy and In-Crossbar Noise Ablation

Using the same four ablation setups, we evaluate DNN accuracy running on RAELLA with varying levels of noise. All PIM architectures suffer from analog variation and noise, but RAELLA tolerates noise with lower accuracy loss.

We model variation and noise as a Gaussian distribution that we add to column sums [17]. Given positive/negative sliced product sums $N_+$ and $N_-$, we model the column sum as $\mathcal{N}(\mu, \sigma^2)$. For noise level $E$, we set the mean $\mu$ to the ideal column sum $(N_+ - N_-)$ and the standard deviation $\sigma = E\sqrt{N_+ + N_-}$. After calculating a column sum, it is sent to an ADC and will be saturated at $[-64, 64)$ if out of range. Noise is additive across positive/negative sliced products. We test with up to 12% error, or $\sigma \approx 4$ for 512 $2b \times 2b$ MACs.

We make two changes to ISAAC to improve noise tolerance for a fair comparison. ISAAC's encoding strategy relies on an analog circuit that sums crossbar inputs [54]. This component has been shown to degrade accuracy under noise [73], so we replace it with a digital equivalent. For BERT accuracy, we give ISAAC two cycles to process positive/negative inputs, matching RAELLA. This provides additional noise resistance. Fig. 15 shows the following:

- ISAAC: all DNNs suffer high accuracy loss for noise > 4%. ISAAC uses unsigned weights, which have dense high-order bits (Fig. 8). Dense bits generate larger, higher-noise values, and noise in high-order slices creates large errors in results.
- Center+Offset: this is critical. Offset encoding provides noise resistance [73], and Center+Offset increases bit sparsity and decreases noise. Intuitively, digital center processing moves much of the computation out of the noisy analog domain.
- Adaptive Weight Slicing: accuracy is further improved. RAELLA's empirical slicing strategy is noise-aware, allowing RAELLA to adapt slicing to varying levels of noise. As noise increases, Adaptive Weight Slicing uses fewer bits per weight slice to reduce error, with five weight slices for most layers at the highest tested noise.
- RAELLA: with speculation, RAELLA maintains accuracy similar to that of a no-speculation approach. The recovery step prevents accuracy drop due to failed speculations.

We find that RAELLA can maintain DNN accuracy at higher noise levels, while on ISAAC, all DNNs suffer sharp accuracy loss at lower noise levels. Compact DNNs suffer higher accuracy degradation from errors compared to larger DNNs [76]. BERT uniquely benefits from the sparsity generated by two-cycle positive/negative inputs. This, along with BERT's large size, allows BERT to maintain better accuracy at high noise levels.

RAELLA can adapt to varying noise; adaptive weight slicing automatically trades off storage density and efficiency for correctness by using fewer bits per slice in higher-noise scenarios. This lets RAELLA maintain accuracy without retraining while extracting as much efficiency as possible under noise constraints.

# 8 RELATED WORK

Xiao et al. [73] provide an in-depth and insightful exploration of DNN accuracy versus fidelity, differential encoding, and PIM design space decisions. We urge the reader to read this work for a deeper understanding of the tradeoffs explored in RAELLA.

Multiple works push the bounds of low ADC resolution. TinyADC [79] retrains while pruning DNN weight bits, achieving impressive reductions in column sum resolution. BRAHMS [57] tailors ADC quantization steps for each layer to maximize DNN accuracy under fidelity loss. Guo et al. [15] exploit naturally-low column sums to reduce ADC resolution and scale the number of crossbar rows used based on a column sum prediction. McDanel et al. [33] explore low-resolution ADC quantization and DNN error tolerance. RAELLA achieves much greater ADC resolution reductions than these works (2b-3b vs. 10b).

Newton [36] improves ISAAC by varying ADC resolution, using heterogeneous tiles, and using transformations that reduce computation. These are orthogonal to RAELLA; it would be interesting to see how an accelerator may combine both.

# 9 CONCLUSION

RAELLA shows that PIM accelerators can reduce high ADC costs without retraining or modifying DNNs. By encoding for low-resolution analog outputs and changing slicing patterns, RAELLA can reshape the distributions of computed analog values. RAELLA uses this ability to keep computed analog values low-resolution and high-fidelity while extracting as much efficiency and throughput as possible from each DNN layer. We hope that, by expanding the set of retraining-free strategies available to PIM designers, RAELLA will inspire future hardware strategies, permit novel co-design opportunities, and broaden the scope in which PIM can be used.

# 10 ACKNOWLEDGEMENTS

This work was funded in part by Ericsson, the MIT AI Hardware Program, and MIT Quest.

# REFERENCES


[1] Fabien Alibart, Ligang Gao, Brian D Hoskins, and Dmitri B Strukov. 2012. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology* 23, 7 (jan 2012), 075201. https://doi.org/10.1088/0957-4484/23/7/075201

[2] Pai-Yu Chen, Xiaochen Peng, and Shimeng Yu. 2017. NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures. In *2017 IEEE International Electron Devices Meeting (IEDM)*. 6.1.1–6.1.4. https://doi.org/10.1109/IEDM.2017.8268337

[3] Yuzong Chen, Lu Lu, Bongjin Kim, and Tony Tae-Hyoung Kim. 2020. Reconfigurable 2T2R ReRAM Architecture for Versatile Data Storage and Computing In-Memory. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 12 (2020), 2636–2649. https://doi.org/10.1109/TVLSI.2020.3028848

[4] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2014. DaDianNao: A Machine-Learning Supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. 609–622. https://doi.org/10.1109/MICRO.2014.58

[5] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. 2016. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 27–39. https://doi.org/10.1109/ISCA.2016.13

[6] Jungwook Choi, Swagath Venkataramani, Vijayalakshmi (Viji) Srinivasan, Kailash Gopalakrishnan, Zhuo Wang, and Pierce Chuang. 2019. Accurate and Efficient 2-bit Quantized Neural Networks. In *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia (Eds.), Vol. 1. 348–359. https://proceedings.mlsys.org/paper/2019/file/006f52e9102a8d3be2fe5614f42ba989-Paper.pdf

[7] Teyuh Chou, Wei Tang, Jacob Botimer, and Zhengya Zhang. 2019. CASCADE: Connecting RRAMs to Extend Analog Dataflow In An End-To-End In-Memory Processing Paradigm. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO '52)*. Association for Computing Machinery, New York, NY, USA, 114–125. https://doi.org/10.1145/3352460.3358328

[8] Chaoqun Chu, Yanzhi Wang, Yilong Zhao, Xiaolong Ma, Shaokai Ye, Yunyan Hong, Xiaoyao Liang, Yinhe Han, and Li Jiang. 2020. PIM-Prune: Fine-Grain DCNN Pruning for Crossbar-Based Process-In-Memory Architecture. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC18072.2020.9218523

[9] Philip Colangelo, Nasibeh Nasiri, Eriko Nurvitadhi, Asit Mishra, Martin Margala, and Kevin Nealis. 2018. Exploration of Low Numeric Precision Deep Learning Inference Using Intel® FPGAs. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 73–80. https://doi.org/10.1109/FCCM.2018.00020

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[11] Lei Deng, Ling Liang, Guanrui Wang, Liang Chang, Xing Hu, Xin Ma, Liu Liu, Jing Pei, Guoqi Li, and Yuan Xie. 2020. SemiMap: A Semi-Folded Convolution Mapping for Speed-Overhead Balance on Crossbars. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 1 (2020), 117–130. https://doi.org/10.1109/TCAD.2018.2883959

[12] Andrea Fasoli, Chia-Yu Chen, Mauricio Serrano, Xiao Sun, Naigang Wang, Swagath Venkataramani, George Saon, Xiaodong Cui, Brian Kingsbury, Wei Zhang, Zoltán Tüske, and Kailash Gopalakrishnan. 2021. 4-Bit Quantization of LSTM-Based Speech Recognition Models. In *Proc. Interspeech 2021*. 2586–2590. https://doi.org/10.21437/Interspeech.2021-1962

[13] Ligang Gao, Fabien Alibart, and Dmitri B. Strukov. 2013. A High Resolution Nonvolatile Analog Memory Ionic Devices.

[14] Sujan K. Gonugondla, Charbel Sakr, Hassan Dbouk, and Naresh R. Shanbhag. 2020. Fundamental Limits on the Precision of In-Memory Architectures. In *Proceedings of the 39th International Conference on Computer-Aided Design* (Virtual Event, USA) *(ICCAD '20)*. Association for Computing Machinery, New York, NY, USA, Article 128, 9 pages. https://doi.org/10.1145/3400302.3416344

[15] Mengyu Guo, Zihan Zhang, Jianfei Jiang, Qin Wang, and Naifeng Jing. 2022. Boosting ReRAM-based DNN by Row Activation Oversubscription. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 604–609. https://doi.org/10.1109/ASP-DAC52403.2022.9712520

[16] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778.

[17] Miao Hu, John Paul Strachan, Zhiyong Li, Emmanuelle M. Grafals, Noraica Davila, Catherine Graves, Sity Lam, Ning Ge, Jianhua Joshua Yang, and R. Stanley Williams. 2016. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2897937.2898010

[18] Norman P. Jouppi, Andrew B. Kahng, Naveen Muralimanohar, and Vaishnav Srinivas. 2015. CACTI-IO: CACTI With OFF-Chip Power-Area-Timing Models. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 7 (2015), 1254–1267. https://doi.org/10.1109/TVLSI.2014.2334635

[19] Tinu Theckel Joy, Santu Rana, Sunil Gupta, and Svetha Venkatesh. 2016. Hyperparameter tuning for big data using Bayesian optimisation. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2574–2579. https://doi.org/10.1109/ICPR.2016.7900023

[20] Sangyeob Kim, Sangjin Kim, Soyeon Um, Soyeon Kim, Kwantae Kim, and Hoi-Jun Yoo. 2022. Neuro-CIM: A 310.4 TOPS/W Neuromorphic Computing-in-Memory Processor with Low WL/BL activity and Digital-Analog Mixed-mode Neuron Firing. In *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. 38–39. https://doi.org/10.1109/VLSITechnologyandCir46769.2022.9830276

[21] Michael Klachko, Mohammad Reza Mahmoodi, and Dmitri Strukov. 2019. Improving Noise Tolerance of Mixed-Signal Neural Networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN.2019.8851966

[22] Raghuraman Krishnamoorthi. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *CoRR* abs/1806.08342 (2018). arXiv:1806.08342 http://arxiv.org/abs/1806.08342

[23] Lukas Kull, Thomas Toifl, Martin Schmatz, Pier Andrea Francese, Christian Menolfi, Matthias Braendli, Marcel Kossel, Thomas Morf, Toke Meyer Andersen, and Yusuf Leblebici. 2013. A 3.1mW 8b 1.2GS/s single-channel asynchronous SAR ADC with alternate comparators for enhanced speed in 32nm digital SOI CMOS. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*. 468–469. https://doi.org/10.1109/ISSCC.2013.6487818

[24] Weitao Li, Pengfei Xu, Yang Zhao, Haitong Li, Yuan Xie, and Yingyan Lin. 2020. TIMELY: Pushing Data Movements and Interfaces in PIM Accelerators towards Local and in Time Domain. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture* (Virtual Event) *(ISCA '20)*. IEEE Press, 832–845. https://doi.org/10.1109/ISCA45697.2020.00073

[25] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. 2021. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* 461 (2021), 370–403. https://doi.org/10.1016/j.neucom.2021.07.045

[26] Jilan Lin, Zhenhua Zhu, Yu Wang, and Yuan Xie. 2019. Learning the Sparsity for ReRAM: Mapping and Pruning Sparse Neural Network for ReRAM Based Accelerator. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC '19)*. Association for Computing Machinery, New York, NY, USA, 639–644. https://doi.org/10.1145/3287624.3287715

[27] Eike Linn, Roland Rosezin, Carsten Kügeler, and Rainer Waser. 2010. Complementary resistive switches for passive nanocrossbar memories. *Nature Materials* 9, 5 (01 May 2010), 403–406. https://doi.org/10.1038/nmat2748

[28] Qi Liu, Bin Gao, Peng Yao, Dong Wu, Junren Chen, Yachuan Pang, Wenqiang Zhang, Yan Liao, Cheng-Xin Xue, Wei-Hao Chen, Jianshi Tang, Yu Wang, Meng-Fan Chang, He Qian, and Huaqiang Wu. 2020. 33.2 A Fully Integrated Analog ReRAM Based 78.4TOPS/W Compute-In-Memory Chip with Fully Parallel MAC Computing. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 500–502. https://doi.org/10.1109/ISSCC19947.2020.9062953

[29] Anni Lu, Xiaochen Peng, Wantong Li, Hongwu Jiang, and Shimeng Yu. 2021. NeuroSim Validation with 40nm RRAM Compute-in-Memory Macro. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 1–4. https://doi.org/10.1109/AICAS51828.2021.9458501

[30] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[31] Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the Machine-Vision Package of Torch. In *Proceedings of the 18th ACM International Conference on Multimedia* (Firenze, Italy) *(MM '10)*. Association for Computing Machinery, New York, NY, USA, 1485–1488. https://doi.org/10.1145/1873951.1874254

[32] Matthew J. Marinella, Sapan Agarwal, Alexander Hsia, Isaac Richter, Robin Jacobs-Gedrim, John Niroula, Steven J. Plimpton, Engin Ipek, and Conrad D. James. 2018. Multiscale Co-Design Analysis of Energy, Latency, Area, and Accuracy of a ReRAM Analog Neural Training Accelerator. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8, 1 (2018), 86–101. https://doi.org/10.1109/JETCAS.2018.2796379

[33] Bradley McDanel, Sai Qian Zhang, and H. T. Kung. 2021. Saturation RRAM Leveraging Bit-Level Sparsity Resulting from Term Quantization. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. https://doi.org/10.1109/ISCAS51556.2021.9401293

[34] Sparsh Mittal. 2019. A Survey of ReRAM-Based Architectures for Processing-In-Memory and Neural Networks. *Machine Learning and Knowledge Extraction* 1, 1 (2019), 75–114. https://doi.org/10.3390/make1010005

[35] Boris Murmann. 2013. Energy limits in A/D converters. In *2013 IEEE Faible Tension Faible Consommation*. 1–4. https://doi.org/10.1109/FTFC.2013.6577781

[36] Anirban Nag, Rajeev Balasubramonian, Vivek Srikumar, Ross Walker, Ali Shafiee, John Paul Strachan, and Naveen Muralimanohar. 2018. Newton: Gravitating

Towards the Physical Limits of Crossbar Acceleration. *IEEE Micro* 38, 5 (2018), 41–49. https://doi.org/10.1109/MM.2018.053631140

[37] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. 2019. Data-Free Quantization Through Weight Equalization and Bias Correction. (2019). https://doi.org/10.48550/ARXIV.1906.04721

[38] M. O'Halloran and R. Sarpeshkar. 2004. A 10-nW 12-bit accurate analog storage cell with 10-aA leakage. *IEEE Journal of Solid-State Circuits* 39, 11 (2004), 1985–1996. https://doi.org/10.1109/JSSC.2004.835817

[39] Atsuya Okazaki, Pritish Narayanan, Stefano Ambrogio, Kohji Hosokawa, Hsinyu Tsai, Akiyo Nomura, Takeo Yasuda, Charles Mackin, Alexander Friz, Masatoshi Ishii, Yasuteru Kohda, Katie Spoon, An Chen, Andrea Fasoli, Malte J. Rasch, and Geoffrey W. Burr. 2022. Analog-memory-based 14nm Hardware Accelerator for Dense Deep Neural Networks including Transformers. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. 3319–3323. https://doi.org/10.1109/ISCAS48785.2022.9937292

[40] Shunsuke Okumura, Makoto Yabuuchi, Kenichiro Hijioka, and Koichi Nose. 2019. A Ternary Based Bit Scalable, 8.80 TOPS/W CNN accelerator with Many-core Processing-in-memory Architecture with 896K synapses/mm2. In *2019 Symposium on VLSI Technology*. C248–C249. https://doi.org/10.23919/VLSIT.2019.8776544

[41] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. 2019. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 304–315. https://doi.org/10.1109/ISPASS.2019.00042

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[43] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. https://doi.org/10.48550/ARXIV.2104.10350

[44] Xiaochen Peng, Shanshi Huang, Hongwu Jiang, Anni Lu, and Shimeng Yu. 2021. DNN+NeuroSim V2.0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-Chip Training. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 11 (2021), 2306–2319. https://doi.org/10.1109/TCAD.2020.3043731

[45] Lillian Pentecost, Alexander Hankin, Marco Donato, Mark Hempstead, Gu-Yeon Wei, and David Brooks. 2022. NVMExplorer: A Framework for Cross-Stack Comparisons of Embedded Non-Volatile Memories. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 938–956. https://doi.org/10.1109/HPCA53966.2022.00073

[46] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. https://doi.org/10.48550/ARXIV.1802.05668

[47] Ximing Qiao, Xiong Cao, Huanrui Yang, Linghao Song, and Hai Li. 2018. AtomLayer: A Universal ReRAM-Based CNN Accelerator with Atomic Layer Computation. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC.2018.8465832

[48] Songyun Qu, Bing Li, Ying Wang, and Lei Zhang. 2021. ASBP: Automatic Structured Bit-Pruning for RRAM-based NN Accelerator. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 745–750. https://doi.org/10.1109/DAC18074.2021.9586105

[49] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 2383–2392. https://doi.org/10.18653/v1/D16-1264

[50] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. https://doi.org/10.48550/ARXIV.2204.06125

[51] Babak Rokh, Ali Azarpeyvand, and Ali Reza Khanteymoori. 2022. A Comprehensive Survey on Model Quantization for Deep Neural Networks. *ArXiv* abs/2205.07877 (2022).

[52] Mehdi Saberi, Reza Lotfi, Khalil Mafinezhad, and Wouter A. Serdijn. 2011. Analysis of Power Consumption and Linearity in Capacitive Digital-to-Analog Converters Used in Successive Approximation ADCs. *IEEE Transactions on Circuits and Systems I: Regular Papers* 58, 8 (2011), 1736–1748. https://doi.org/10.1109/TCSI.2011.2107214

[53] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

[54] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 14–26. https://doi.org/10.1109/ISCA.2016.12

[55] Mahmut E. Sinangil, Burak Erbagci, Rawan Naous, Kerem Akarvardar, Dar Sun, Win-San Khwa, Hung-Jen Liao, Yih Wang, and Jonathan Chang. 2021. A 7-nm Compute-in-Memory SRAM Macro Supporting Multi-Bit Input, Weight and Output and Achieving 351 TOPS/W and 372.4 GOPS. *IEEE Journal of Solid-State Circuits* 56, 1 (2021), 188–198. https://doi.org/10.1109/JSSC.2020.3031290

[56] Linghao Song, Xuehai Qian, Hai Li, and Yiran Chen. 2017. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 541–552. https://doi.org/10.1109/HPCA.2017.55

[57] Tao Song, Xiaoming Chen, Xiaoyu Zhang, and Yinhe Han. 2021. BRAHMS: Beyond Conventional RRAM-based Neural Network Accelerators Using Hybrid Analog Memory System. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. 1033–1038. https://doi.org/10.1109/DAC18074.2021.9586247

[58] Katie Spoon, Hsinyu Tsai, An Chen, Malte J. Rasch, Stefano Ambrogio, Charles Mackin, Andrea Fasoli, Alexander M. Friz, Pritish Narayanan, Milos Stanisavljevic, and Geoffrey W. Burr. 2021. Toward Software-Equivalent Accuracy on Transformer-Based Deep Neural Networks With Analog Memory Devices. *Frontiers in Computational Neuroscience* 15 (2021). https://doi.org/10.3389/fncom.2021.675741

[59] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. 2020. *Efficient Processing of Deep Neural Networks*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01766-7

[60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[61] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2818–2826. https://doi.org/10.1109/CVPR.2016.308

[62] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1701–1708. https://doi.org/10.1109/CVPR.2014.220

[63] Fengbin Tu, Weiwei Wu, Shouyi Yin, Leibo Liu, and Shaojun Wei. 2018. RANA: Towards Efficient Neural Acceleration with Refresh-Optimized Embedded DRAM. *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)* (2018), 340–352.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[65] Marian Verhelst and Boris Murmann. 2012. Area scaling analysis of CMOS ADCs. *Electronics Letters* 48 (2012), 314–315.

[66] Ching-Hua Wang, Yi-Hung Tsai, Kai-Chun Lin, Meng-Fan Chang, Ya-Chin King, Chrong-Jung Lin, Shyh-Shyuan Sheu, Yu-Sheng Chen, Heng-Yuan Lee, Frederick T. Chen, and Ming-Jinn Tsai. 2010. Three-dimensional 4F2 ReRAM cell with CMOS logic compatible process. In *2010 International Electron Devices Meeting*. 29.6.1–29.6.4. https://doi.org/10.1109/IEDM.2010.5703446

[67] Linfang Wang, Wang Ye, Chunmeng Dou, Xin Si, Xiaoxin Xu, Jing Liu, Dashan Shang, Jianfeng Gao, Feng Zhang, Yongpan Liu, Meng-Fan Chang, and Qi Liu. 2021. Efficient and Robust Nonvolatile Computing-In-Memory Based on Voltage Division in 2T2R RRAM With Input-Dependent Sensing Control. *IEEE Transactions on Circuits and Systems II: Express Briefs* 68, 5 (2021), 1640–1644. https://doi.org/10.1109/TCSII.2021.3067385

[68] Wikipedia. 2022. Iron law of processor performance — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Iron%20law%20of%20processor%20performance&oldid=1112639388. [Online; accessed 22-November-2022].

[69] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. 2020. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. *ArXiv* abs/2004.09602 (2020).

[70] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. 2020. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation.

[71] Yannan Nellie Wu, Joel S. Emer, and Vivienne Sze. 2019. Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942149

[72] Yannan Nellie Wu, Vivienne Sze, and Joel S. Emer. 2020. An Architecture-Level Energy and Area Estimator for Processing-In-Memory Accelerator Designs. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 116–118. https://doi.org/10.1109/ISPASS48437.2020.00024

[73] T. Patrick Xiao, Ben Feinberg, Christopher H. Bennett, Venkatraman Prabhakar, Prashant Saxena, Vineet Agrawal, Sapan Agarwal, and Matthew J. Marinella. 2021. On the Accuracy of Analog Neural Network Inference Accelerators [Feature]. *IEEE Circuits and Systems Magazine* 22 (2021), 26–48.

[74] J. Joshua Yang, Dmitri B. Strukov, and Duncan R. Stewart. 2013. Memristive devices for computing. *Nature Nanotechnology* 8, 1 (01 Jan 2013), 13–24. https://doi.org/10.1038/nnano.2012.240

[75] Tzu-Hsien Yang, Hsiang-Yun Cheng, Chia-Lin Yang, I-Ching Tseng, Han-Wen Hu, Hung-Sheng Chang, and Hsiang-Pang Li. 2019. Sparse ReRAM Engine: Joint Exploration of Activation and Weight Sparsity in Compressed Neural Networks. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*. 236–249.

[76] Tien-Ju Yang and Vivienne Sze. 2019. Design Considerations for Efficient Deep Neural Networks on Processing-in-Memory Accelerators. 22.1.1–22.1.4. https://doi.org/10.1109/IEDM19573.2019.8993662

[77] Amir Yazdanbakhsh, Ashkan Moradifirouzabadi, Zheng Li, and Mingu Kang. 2022. Sparse Attention Acceleration with Synergistic In-Memory Pruning and On-Chip Recomputation. https://doi.org/10.48550/ARXIV.2209.00606

[78] Shihui Yin, Zhewei Jiang, Jae-Sun Seo, and Mingoo Seok. 2020. XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks. *IEEE Journal of Solid-State Circuits* 55, 6 (2020), 1733–1743. https://doi.org/10.1109/JSSC.2019.2963616

[79] Geng Yuan, Payman Behnam, Yuxuan Cai, Ali Shafiee, Jingyan Fu, Zhiheng Liao, Zhengang Li, Xiaolong Ma, Jieren Deng, Jinhui Wang, Mahdi Bojnordi, Yanzhi Wang, and Caiwen Ding. 2021. TinyADC: Peripheral Circuit-aware Weight Pruning Framework for Mixed-signal DNN Accelerators. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*. 926–931. https://doi.org/10.23919/DATE51398.2021.9474235

[80] Geng Yuan, Payman Behnam, Zhengang Li, Ali Shafiee, Sheng Lin, Xiaolong Ma, Hang Liu, Xuehai Qian, Mahdi Nazm Bojnordi, Yanzhi Wang, and Caiwen Ding. 2021. FORMS: Fine-grained Polarized ReRAM-based In-situ Computation for Mixed-signal DNN Accelerator. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 265–278. https://doi.org/10.1109/ISCA52012.2021.00029

[81] Jinshan Yue, Yongpan Liu, Fang Su, Shuangchen Li, Zhe Yuan, Zhibo Wang, Wenyu Sun, Xueqing Li, and Huazhong Yang. 2019. AERIS: Area/Energy-Efficient 1T2R ReRAM Based Processing-in-Memory Neural Network System-on-a-Chip. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC '19)*. Association for Computing Machinery, New York, NY, USA, 146–151. https://doi.org/10.1145/3287624.3287635

[82] Xiandong Zhao, Ying Wang, Xuyi Cai, Chuanming Liu, and Lei Zhang. 2020. Linear Symmetric Quantization of Neural Networks for Low-precision Integer Hardware. In *ICLR*.

Received 22 November 2022; revised 21 February 2023; accepted 9 March 2023