# Saturation RRAM Leveraging Bit-level Sparsity Resulting from Term Quantization

Bradley McDanel
*Franklin & Marshall College*
mcdanel@fas.harvard.edu

H. T. Kung
*Harvard University*
kung@harvard.edu

Sai Qian Zhang
*Harvard University*
zhangs@g.harvard.edu

***Abstract*—The proposed saturation RRAM for in-memory computing of a pre-trained Convolutional Neural Network (CNN) inference imposes a limit on the maximum analog value (current) output from each bitline in order to reduce analog-to-digital (A/D) conversion costs. The proposed scheme uses term quantization (TQ) to enable flexible bit annihilation at any position for a value in the context of a group of weights values in RRAM. This enables a drastic reduction in the required ADC resolution while still maintaining CNN model accuracy. Specifically, we show that the A/D conversion errors after TQ have a minimum impact on the classification accuracy of the inference task. For instance, for a 64x64 RRAM, reducing the ADC resolution from 6 bits to 4 bits enables a 1.58x reduction in the total system power, *without impacting classification accuracy*.**

***Index Terms*—analog computing; in-memory computing; resistive RAM (RRAM); noise; analog-to-digital conversion (A/D conversion); analog-to-digital converter (ADC); dot-product computation; convolutional neural network (CNN)**

## I. Introduction

The widespread popularity of Convolutional Neural Networks (CNNs), coupled with their generally high computational requirements, has led to the development of novel hardware architectures with improved computational efficiency. In-memory computation using resistive random-access memory (RRAM) [1] is an approach that offers a simultaneous reduction in power consumption and decrease in processing latency.

However, while RRAM may perform matrix-matrix multiplication extremely efficiently, the digital-to-analog (D/A) and analog-to-digital (A/D) conversions, which interfaces the RRAM with digital components, introduce significant overhead. For instance, in ISAAC [1], the digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) account for over $60\%$ of the total power consumption and $30\%$ of the circuit area of the RRAM-based architecture.

Figure 1 shows a standard RRAM crossbar (see, e.g., ISAAC [1] and CASCADE [2]) performing dot-product computation of length $64$ on bit-sliced weights and data (i.e., activations), where all values are 4-bit fixed-point numbers. The crossbar points on a bitline are used to accumulate partial sums, which we refer to as *accumulation points* or simply *points*. Conventionally, a 6-bit ADC is required for an RRAM with $64$ rows to ensure accurate conversion for the maximum output value of $64$ when all data and weights terms are $1$.

In this work, we propose to reduce the associated costs of ADCs in RRAM-based architectures through the use of a novel quantization technique called term quantization (TQ) [3]. Term

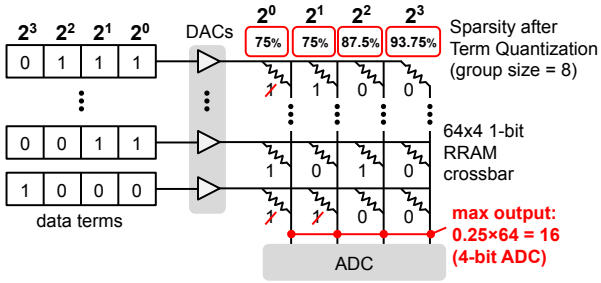


Fig. 1: An RRAM with a 64 rows $\times$ 4 columns crossbar and 1-bit RRAM cells for computing dot products. Both data and weights are bit-sliced, with each weight term (e.g., $2^0$) occupying a separate RRAM column. Term quantization (TQ) [3] sets low-order power-of-two terms to 0 such as the $2^0$ and $2^1$ terms with red slashes to satisfy a group budget. The sparsity introduced by TQ reduces the maximum output signal of the bitlines which enables a low-resolution 4-bit ADC.

quantization limits the number of nonzero power-of-two terms across a group of values. Figure 1 illustrates how TQ can be applied on a group of bit-sliced weights stored in RRAM. In the figure, the $2^0$ and $2^1$ bitlines have several terms that have been truncated to $0$ in order to satisfy the group budget constraint (see Section II-B for details on TQ). This limits the maximum output for a bitline in the crossbar as it depends on the bit-level sparsity of the weights and data multiplied in that bitline. For instance, in this illustration example, the $2^0$ weight column sparsity has been increased due to the term truncation, enabling a lower-resolution 4-bit ADC to be used instead of a standard 6-bit ADC. After apply TQ, we propose to *saturate* (clamp) the analog signal value $x$ output by each bitline for A/D conversion to a preconfigured saturation value $v$ when $x$ exceeds a preconfigured saturation threshold $\tau$. If a signal $x$ is less than $\tau$, then $x$ is sent to a low-resolution ADC for decoding values below $\tau$. Otherwise, the ADC is skipped and $v$ is returned.

Additionally, our saturation approach has the added benefit of not relying on the larger $x$ values subject to higher degrees of noise. Assume that the noise scales with the number of activated accumulation points in an RRAM column. Then, a value $x$ below a clamping threshold $\tau = 8$, such as $12$, will incur smaller noise than a value above $\tau$, such as $53$. Due to

this additional noise, a conventional 6-bit ADC will not be able to accurately decode all 6 bits for larger $x$ values. In Section V we discuss the noise implication of using a 3-bit ADC with saturation over a standard 6-bit ADC.

The main contributions of the paper are:

- The novel idea of *saturation RRAM* in reducing the required ADC resolution and improving noise tolerance via a saturation threshold.
- The use of term quantization to increase the sparsity in bit-sliced weights deployed in RRAM.
- A *noise analysis* (Section V) on the benefits of the proposed use of a low-resolution ADC for saturation RRAM compared to a standard high-resolution ADC.

## II. Background

In this section, we first discuss the development of RRAM-based architectures for CNNs in Section II-A. Then, in Section II-B, we show how different forms of quantization can be applied to groups of weights stored in RRAM.

### A. RRAM Architectures for CNNs

ISAAC [1] described the use of RRAM crossbars to both store CNN weights and perform matrix-matrix multiplication in-memory in an analog fashion. They proposed the use of a bit-sliced format (with 1-bit input and 2-bit cells) discussed earlier to mitigate noise and lower the overhead of D/A and A/D conversions. Using these resolutions with a 128×128 crossbar, a bitline of length 128 could output a maximum value of $4 \times 128 = 512$, which requires a 9-bit ADC.

More recently, CASCADE [2] has suggested that high-resolution RRAM introduces significant noise due to process, voltage and temperature variations [4] that makes accurate A/D conversion both challenging and expensive (note that the ADC power scales exponentially with resolution [1]). Due to this, CASCADE used a smaller 64×64 RRAM crossbar, with 1-bit input and 1-bit weight resolutions, and a 6-bit ADC in their evaluation. We use the same RRAM settings proposed by CASCADE while aiming to reduce the ADC resolution using saturation described in Section III-A.

### B. Quantization for RRAM Weight Values

Figure 2 illustrates three types of quantization applied to a group of four weight values. Uniform quantization [5] shown in Figure 2(a) uses a fixed number of terms to represent values without putting any restrictions on the number of nonzero terms across the values. Logarithmic quantization, in Figure 2(b), is a more aggressive form of quantization that works by rounding each value to the nearest power-of-two term [6]. This allows for more efficient inference in conventional hardware, but leads to a large accuracy degradation compared to conventional quantization. Power-of-two **term quantization** (TQ) relaxes logarithmic quantization by allowing a term budget ($\alpha = 8$ in the figure) of one or more terms for values in a group (e.g., group size $g = 4$) [3]. Unlike the prior work on TQ, in this work, we propose to use TQ to increase sparsity in RRAM bitlines (storing power-of-two terms), which enables a lower resolution ADC to be used.

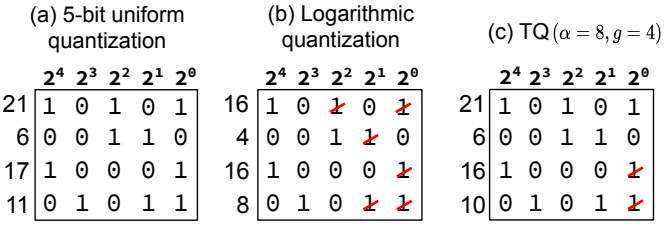


Fig. 2: (a) 5-bit uniform quantization applied on four values. (b) Logarithmic quantization uses only the largest term in each value. (c) Term quantization (TQ) keeps the largest $\alpha = 8$ terms across a group of 4 weights ($g = 4$).

## III. Exploiting TQ Weight Sparsity in RRAM

First, in Section III-A, we propose a saturation function $\mathcal{S}$ which thresholds infrequently occurring large accumulated values to a predefined smaller value. This enables the use of a lower-resolution ADC. In Section III-B, we describe a method for minimizing the saturation-induced error when using a lower-resolution ADC. Finally, in Section III-C, we evaluate the impact to accuracy of the saturation approach on ImageNet [7] across multiple CNNs. Throughout this section, we assume the term quantization with $\alpha = 8$ and $g = 4$ has been applied across the weights in each CNN layer. Multiple groups are placed sequentially when deployed in a RRAM.

### A. Saturation Bitline A/D Conversion

Using the high degree of sparsity introduced by TQ, we propose to reduce the required ADC resolution in RRAM implementations. For a bitline with 64 points, while an accumulated value $x$ requires a 6-bit ADC for accurate decoding, we may use a 4-bit ADC instead. We propose the use of a saturation function $\mathcal{S}$ to enable this lower resolution ADC as follows:

$$\mathcal{S}(x, \tau, v) = \begin{cases} x, & \text{if } x \leq \tau \\ v, & \text{otherwise} \end{cases} \quad (1)$$

where $x$ is the analog input from a bitline, $\tau$ is a saturation threshold, and $v$ is a saturation value. The purpose of $\mathcal{S}$ is to threshold infrequently occurring large accumulated values (e.g., $x$ larger than $\tau = 16$) to a much smaller value (e.g., $v = 16$) so that the ADC resolution can be reduced.

We developed a custom PyTorch implementation of bit-sliced RRAM to analyze how much saturation error is introduced into the system for different settings of $\tau$ (corresponding to different ADC resolutions). Figure 3 shows the saturation-induced error across the different bitlines in an RRAM as $\tau$ is varied from 1 to 20. The horizontal red lines at $\tau = 8$ and $\tau = 16$ denote the maximum value that can be decoded by a 3-bit ADC and a 4-bit ADC, respectively. (Note that the other values of $\tau$ which are not valid ADC resolutions, such as $\tau = 12$, are provided to illustrate the general trend).

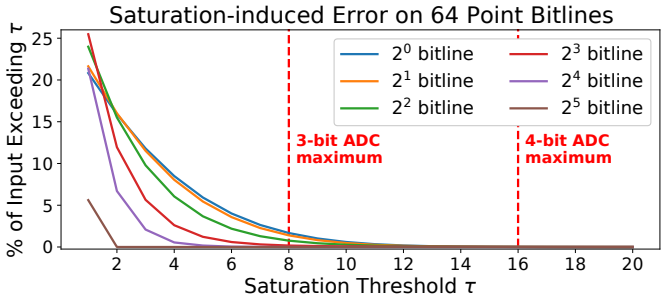


Fig. 3: The percentage of analog input which exceed the saturation threshold $\tau$ (Equation 1) for 6 bitlines as $\tau$ is varied from 1 to 20. The maximum value that can be reliably decoded by a 3-bit and 4-bit ADC are shown in red, corresponding to $\tau = 8$ and $\tau = 16$, respectively. After applying TQ with $\alpha = 8$ and $g = 4$, all bitlines fall below the 4-bit ADC maximum line.

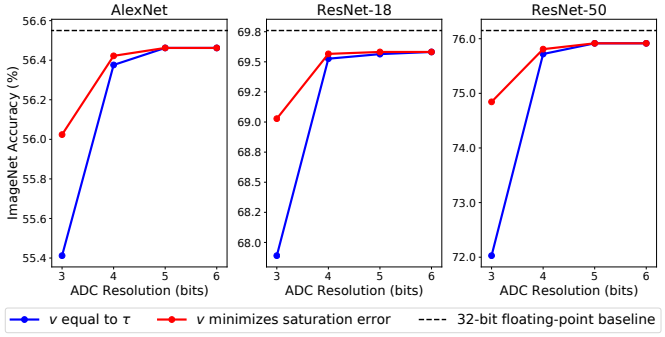


Fig. 4: The classification accuracy of AlexNet, ResNet-18, and ResNet-50 on ImageNet using the proposed saturation RRAM as the ADC resolution is varied from 3 bits to 6 bits.

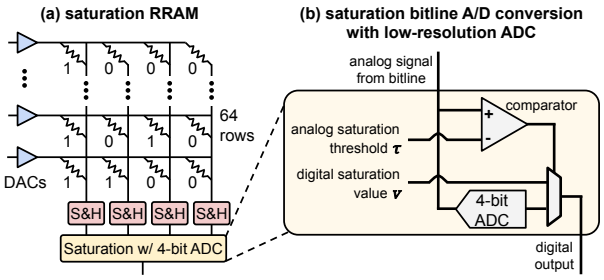


Fig. 5: (a) Saturation RRAM (adapted from ISSAC [1]) which enables the use of a low-resolution (e.g., 4-bit) ADC, instead of the standard 6-bit ADC. (b) The A/D conversion uses a comparator to set bitline signal larger than $\tau$ to $v$. Signals smaller than $\tau$ are passed to a low-resolution (e.g., 4-bit) ADC.

### B. Selection of Saturation Value

Using the saturation value $v$ in Equation 1, we have flexibility in the handling of accumulated values larger than the saturation threshold $\tau$. Here, we present two approaches for the selection of the saturation value $v$. Later, in Section III-C, we provide empirical results on the impact to classification accuracy when using each approach.

*1) $v$ equal to $\tau$:* simply sets $v$ to $\tau$. This approach corresponds to not having a saturation value and instead using the maximum ADC value (e.g., 8 for the 3-bit ADC scenario).

*2) select $v$ which minimizes saturation error:* by sweeping across the possible values for $v$ and selecting the one that introduces the least saturation error. A unique $v_l$ is chosen for each convolution layer $l$, $1 \leq l \leq L$, where $L$ is the total number of layers in the CNN.

### C. Impact of Saturation on CNN Accuracy

We evaluate our saturation approach using pre-trained models for the ImageNet [7] dataset provided by the PyTorch torchvision package for AlexNet [8], ResNet-18 [9], and ResNet-50 [9]. Term quantization ($\alpha = 8$, $g = 4$) is applied to each model to convert the 32-bit floating-point weights and data to 8-bit fixed-point representation with a fixed number of nonzero terms per group. To estimate the saturation-induced error, each convolution layer is then converted to our custom PyTorch implementation, which performs inference using bit-sliced weights and data, and applies the saturation function for the specified ADC resolution (e.g., 4-bit ADC).

Figure 4 shows the ImageNet validation set top-1 classification accuracy under saturation as the ADC resolution is varied from 3 bits to 6 bits using the two different strategies for selecting $v$. The 6-bit ADC resolution is the baseline setting without any saturation error. Interestingly, we observe no difference in classification accuracy between the 5-bit and 6-bit ADC settings for either strategy, suggesting that a 5-bit ADC can replace the standard 6-bit ADC without changing the performance. The 4-bit ADC setting leads to a $0.05\%$ decrease in accuracy when the optimal $v$ is selected per layer compared to a $0.1\%$ to $0.2\%$ decrease when setting $v$ equal to $\tau$. For the 3-bit ADC setting, the difference in performance between the two saturation value strategies is more significant, with a $2.8\%$ difference in accuracy for ResNet-50. The higher percentage of accumulated values that exceed $\tau$ in the 3-bit ADC scenario makes optimal selection of $v$ more important.

## IV. Saturation RRAM System

In this section, we describe a hardware system design for a saturation RRAM which implements the saturation bitline A/D conversion based on $\mathcal{S}$ (Equation 1). We use the RRAM design proposed by ISSAC [1] with 1-bit crossbar points (instead of 2-bit points as in ISSAC) for a $64 \times 64$ RRAM crossbar. Figure 5a provides an overview of the proposed saturation RRAM. The key modification to the system is the use of the saturation function (Equation 1) followed by a low-resolution (e.g., 4-bit) ADC, depicted at the bottom of Figure 5a, which replaces the standard high-resolution 6-bit ADC.

Figure 5b shows the hardware implementation of A/D conversion using saturation with a low-resolution ADC. If the analog signal from a bitline is larger than $\tau$, the low-precision 4-bit ADC is skipped and the predefined saturation value $v$ is returned. Otherwise, it is passed to the low-resolution ADC

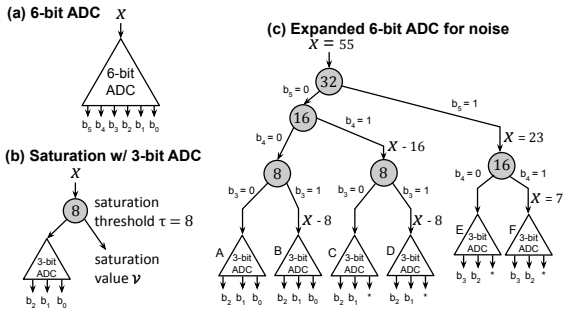


Fig. 6: (a) A 6-bit ADC for an RRAM bitline of 64 rows converting input $x$. (b) The proposed saturation scheme based on a 3-bit baseline ADC. (c) The 6-bit ADC in (a) expanded. Due to noise accumulated by active points, ADCs for large $x$ can only decode higher order bits.

to perform standard A/D conversion in the reduced range (e.g., using a 4-bit resolution instead of the standard 6-bit resolution). As discussed earlier in Section III-A, since large accumulated values output from bitlines are infrequent, this design can be used in place of the higher-resolution ADC without significantly impacting the accuracy of the system.

## V. ADC Noise Analysis

In this section, we show how our proposed saturation scheme with a 3-bit ADC can be viewed as an approximate solution to a 6-bit ADC under noise. We assume that noise on a bitline is proportional to the number of active points.

### A. Saturation Scheme as Approximation under Noise

As depicted in Figure 6a, if there is no noise, a 6-bit ADC can fully decode an analog signal value $x$ accumulated on a bitline of 64 rows into 6 output bits ($b_5$, $b_4$, $b_3$, $b_2$, $b_1$, $b_0$). Figure 6b illustrates our proposed saturation scheme. If $x$ is less than $\tau = 8$, then it is sent to a 3-bit baseline ADC to decode ($b_2$, $b_1$, $b_0$) such that the 6-bit output is (0, 0, 0, $b_2$, $b_1$, $b_0$). Otherwise, $x$ escapes to a saturation value $v$ that is slightly larger than the saturation threshold of 8 (e.g., $v = 10$).

Figure 6c expands the 6-bit ADC into six instances of the 3-bit baseline ADC (A, B, C, D, E, F) by recognizing increased noise associated with large values of $x$. Our proposed saturation scheme in Figure 6b is an approximation to Figure 6c, by escaping the five instances of the 3-bit baseline ADC (i.e., ADC instances B through F). By using the 3-bit baseline ADC once rather than six times, **the proposed saturation scheme achieves a $6\times$ savings in ADC.** In converting an analog signal value $x$ from a bitline, the achievable decoding resolution of $x$ is limited by the amount of noise accumulated on the bitline. That is, a relatively large value of $x$, resulting from a large number of active accumulation points, is subject to a relatively large amount of noise. This means that the ADC should reduce its resolution.

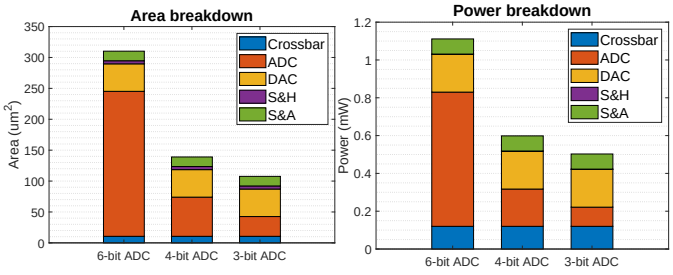


Fig. 7: Area and energy breakdown for a baseline RRAM with a 6-bit ADC compared to saturation RRAMs with 4-bit and 3-bit ADCs. With saturation, the ADC is no longer most expensive system component in terms of area and power.

### B. Further Performance Improvement of Saturation

As we have seen in Section III-C, the proposed saturation scheme with a 3-bit baseline ADC has a modest impact on ImageNet classification accuracy. By comparison, the saturation scheme with a 4-bit baseline ADC, which escapes at a higher threshold $\tau = 16$ rather than $\tau = 8$, has only a minimum impact on accuracy. We could continue improving the performance by reusing the baseline ADC for the additional ADC instances as depicted in Figure 6c. Suppose we are currently using ADC instances A and B. When including an additional ADC, we should use the next one in line, that is C, as it has more impact in improving accuracy as shown in Section III-C.

## VI. Saturation RRAM Evaluation

In this section, we evaluate the area and power of the proposed saturation RRAM using a low-resolution 3-bit or 4-bit ADC shown in Figure 5b against a baseline RRAM with a standard high-resolution 6-bit ADC. We adopt the area and power data in [1], [10] for modeling the crossbar, DAC, Shift & Add (S&A), and Sample & Hold (S&H) components. For modeling the ADC area and power, we use the data from an ADC performance survey [11].

Figure 7 shows the area and power breakdowns for the baseline RRAM and the proposed saturation RRAM with 4-bit and 3-bit ADC, respectively. In the baseline RRAM, the ADC consumes a significant portion of area ($75.6\%$) and power ($54.2\%$). By comparison, the RRAM crossbar, which performs the matrix-matrix multiplication, only accounts for $3.4\%$ of the area and $9.2\%$ of the power. The saturation RRAM with a 3-bit ADC requires $2.65\times$ less area and $1.90\times$ less power than the baseline RRAM. Therefore, by reducing the ADC resolution using the proposed saturation scheme, we can significantly reduce the total area and power consumption of the RRAM.

## VII. Conclusion

The saturation RRAM proposed in this paper, leverages term quantization to reduce the A/D conversion costs. That is, for in-memory CNN computing, there is no need for an RRAM bitline to keep accurate accounting of large values, i.e., they can be simply saturated at a much small value; thereby allowing use of a lower resolution ADC, with minimum impact

on model accuracy. For example, a saturation RRAM of 64 rows can use a threshold of 16, thus allowing the use of a 4-bit ADC instead of 6-bit ADC. Saturation RRAM offers a general direction of leveraging bit-level sparsity of CNNs in the architecture design of RRAM under noise.

## REFERENCES


[1] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, 2016.

[2] T. Chou, W. Tang, J. Botimer, and Z. Zhang, "Cascade: Connecting rrams to extend analog dataflow in an end-to-end in-memory processing paradigm," in *MICRO*, 2019.

[3] H. T. Kung, B. McDanel, and S. Q. Zhang, "Term revealing: Furthering quantization at run time on quantized dnns," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

[4] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *JSSC*, 2018.

[5] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *ICML*, 2016.

[6] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[10] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *Design Automation Conference (DAC)*, IEEE, 2016.

[11] B. Murmann, "Adc performance survey 1997-2020." Available at: https://web.stanford.edu/~murmann/adcsurvey.html.