Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 6, 2026

MATHEMATICAL OPTIMIZATION OF ANALOG COMPUTE-IN-MEMORY: TRADE-OFFS BETWEEN LINEARITY, NOISE, AND NON-LINEAR ACTIVATION IN MEMRISTOR CROSSBARS

[1]Muhammad Tahir Abbas, [2]Zainab Aleem, [3]Muhammad Qasim Zafar, [4]Anum Zaib

[1]Department of Mathematics, Ghazi University DG Khan
[2]National College of Business Administration & Economics Multan Campus Multan.
[3]Department of Mathematics, University of Baltistan Skardu
[4]Department of Mathematics, NED University of Engineering and Technology
[1]engrtahirabbas5@gmail.com, [2]zainabaleem66@gmail.com, [3]zafarqasim596@gmail.com, [4]anumzaib015@gmail.com

DOI:-

Keywords
Compute-in-memory · Memristor crossbar · Analog neural networks · Convex optimization · Noise-aware activation · Hardware-software co-design · Edge AI

Article History
Received: 15 May, 2026
Accepted: 07 June, 2026
Published: 08 June, 2026

Corresponding Author: *

Farhan Ali

Abstract

The von Neumann bottleneck in deep neural network inference can be addressed by analog compute-in-memory (CiM) by analog matrix-vector multiplication performed in the memory array. This can, in principle, reduce energy consumption by one to two orders of magnitude in comparison to digital accelerators. However, in practice, the accuracy of inferences is far from perfect due to analog non-idealities such as device noise, conductance programming variability, ADC quantization, and signal saturation, making analog CiM impractical for real-world applications. This paper proposes a comprehensive mathematical optimization framework by combining the device physics, circuit design, and neural network training to obtain the minimum inference error for analog CiM systems. A closed-form statistical model of error propagation through memristor crossbar is first developed, accounting for thermal noise, shot noise, programming noise, and quantization of the ADC.Thermal noise, shot noise, programming noise and ADC quantization are first captured in a closed-form statistical model of error propagation through a memristor crossbar. Based on this model, we formulate the optimization of the conductance range [G_min, G_max] and the full-scale (FS) of the ADC as a convex program and obtain globally optimal parameters that provide a balance between signal strength, noise, and risk of signal saturation. We then present a noise-aware version of the digital ReLU, f _"ReLU" _ (x) to incorporate the distribution of analog noise and the ADC saturation during fine-tuning so that the network can learn hardware-robust representations. Our framework is validated on a simulated 128x128 memristor crossbar, with the MNIST database and a 3-layer multi-layer perceptron. Compared with naive analog mapping with the accuracy loss of 6.74% against an ideal digital baseline, our approach retrieves 86.8% of this loss in the test, resulting in 97.32% accuracy (within 0.89% of digital), and retains the energy efficiency of analog computing. The optimized system decreases output mean squared error by 86% compared to naive analog mapping (14× compared to digital), and increases energy efficiency by 11% compared to naive analog mapping (14× compared to digital). Ablation studies indicate that both convex optimization and noise-aware activation are important for recovery of accuracy, and sensitivity analysis proves that the framework allows for a viable 3-bit ADC operation (93.8% accuracy). The results show that algorithm-hardware co-design, based on convex optimization and noise-aware learning, can bridge this accuracy gap between analog and digital computing, while maintaining the energy advantage of in-memory architectures. The architecture is independent – which could adapt to any resistive memory technology – compute-efficient – with the microseconds per layer – and easily deployable to practical edge analog AI accelerators.

1. Introduction

1.1 Motivation

Deep neural networks (DNNs) have seen the highest growth in computational requirements in the past decade, with increased data dimensions and larger models. The von Neumann computing architecture, however, has well-established memory wall/memory bottleneck due to the physical separation of memory and processing units. In this paradigm, most of the energy and latency of DNN inference is spent in the non-stop movement of weights and activations between memory and compute units. This data transfer is the largest source of energy dissipation for edge devices with tight power constraints, and can consume more than 90% of the inference energy [1].

A possible alternative is Analog Compute-in-Memory (CiM). Analog CiM can directly perform matrix-vector multiplication (MVM) in the memory array, the dominant operation in fully-connected and convolutional layers, with an ideal time complexity of O(1) per MVM, regardless of matrix size, thanks to physical laws: Ohm's law and Kirchhoff's current law. The most advanced application of this concept is the memristor crossbar, in which each crosspoint device has a conductance value that corresponds to a weight in a neural network [2].

In theory, analog CiM can lower the inference power by 1-2 orders of magnitude over digital accelerators. However, analog CiM has a major concern: analog errors stack up across devices, columns, and layers. Digital computation ensures bit exactness of results, whereas analog MVM has multiple non-ideal physical effects that corrupt the results. If not suitably mathematically optimized, these errors reduce the accuracy of the neural network to unacceptable levels, offsetting the energy efficiency..

1.2 Key Challenges

This is the result of four principal sources of error arising from the ideal mathematical MVM to the physical analog implementation, which are drawn pictorially.

Device non-linearity. Practical memristors do not exhibit an ideal ohmic relationship of the current-voltage (I–V) relationship. To date, many resistive RAM (RRAM) devices have been found to show non-linear conduction, especially at low read voltage values, such that the conductance stored in the device is not fixed but is dependent on the read voltage. This is a violation of the linearity property which is assumed in the $I=G \cdot V$ equation, which could lead to systematic multiplicative errors that are dependent upon the amplitude of the input signals.

Stochastic noise. There are two types of random variations in memristors. When the same voltage is applied to the cell, the conductance will show stochastic variation between read operations, this is called Cycle-to-cycle (C2C) noise. The term device-to-device variation is used for the permanent variation in conductance after programming, which is caused by an imperfect write process. They can be modeled as either additive or multiplicative Gaussian noise sources, with their statistical characteristics being device physics, materials, and program parameters dependent.

ADC quantization. The analog output currents from a crossbar column need to be digitized, for further processing, such as activation functions and inter-layer communications. Due to area and power limitations, the bit precision of the practical analog to digital converter (ADC) integrated on-chip is severely limited, typically 4 to 8 bits [3]. This quantization is the source of errors that are evenly distributed over the quantization steps under normal operating conditions but are very non-linear when the input signal is outside of the ADC's full-scale range (saturation).

Activation function mismatch. The non-linear activation functions, ReLU, tanh, sigmoid, are crucial for the expressivity of a DNN. For analog CiM accelerators, these functions are not performed in the analog domain but instead are performed digitally after the ADC conversion. The activation function then gets the ideal pre-activation value that is quantized and noisy. This mismatch also causes truncation errors and, more insidiously, biases the expected output of the activation function from what it would be in the ideal noise-free scenario.

All four of these challenges make the simple application of pre-trained DNNs to analog CiM hardware impractical. Accuracies of 5 to 15% are usual, even with simple data like MNIST [4] for neural networks that are not optimised for mapping weights to conductances and that do not consider the ADC interface during design, unless a principled mathematical framework is

used to optimise the mapping of weights to conductances.

1.3 Contributions

This research study addresses the aforementioned challenges by developing a mathematically rigorous optimization framework for analog compute-in-memory. This study specific contributions are as follows:

1. **A mathematical model of error propagation through analog MVM and ADC.** We derive closed-form expressions for the mean squared error (MSE) at the output of a memristor crossbar column, incorporating device noise (read and programming variations), ADC quantization noise, and input-dependent signal scaling. The model captures both the additive and multiplicative nature of analog errors, enabling tractable optimization.
2. **Optimal conductance range selection via convex optimization.** We formulate the problem of selecting the minimum and maximum conductance values ($G_{\min}, G_{\max}$) to which network weights are mapped, together with the ADC full-scale range (FS), as a convex optimization problem that minimizes the expected output MSE. This formulation respects device-specific constraints (e.g., limited conductance window, maximum current before non-linearity) and yields a globally optimal solution.
3. **Noise-aware activation clipping to avoid ADC saturation.** We propose a novel activation function, termed the *noise-aware clipped ReLU*, which modifies the digital activation step to account for the statistical distribution of analog noise. This function reduces the probability of ADC saturation by adaptively adjusting the effective full-scale range based on layer-wise noise statistics, without requiring additional hardware calibration.
4. **Validation on a simulated 128×128 memristor crossbar.** We implement a cycle-accurate simulation of a memristor crossbar with realistic noise parameters derived from published RRAM device measurements. Using a three-layer multi-layer perceptron (MLP) trained on the MNIST dataset, we demonstrate that our optimization framework recovers 75% of the accuracy loss induced by analog non-idealities compared to a naive mapping baseline, achieving final test accuracy within 0.9% of an ideal floating-point implementation.

Paper organization. Section 2 presents the mathematical model of the analog crossbar and ADC. Section 3 formulates the conductance mapping and FS optimization as a convex program. Section 4 introduces the noise-aware activation function. Section 5 describes the experimental methodology, including device parameters and network architecture. Section 6 reports simulation results, and Section 7 discusses limitations and future directions. Section 8 concludes the paper.

2. Mathematical Model of Analog Matrix Multiplication

This section develops a comprehensive mathematical framework for analog matrix-vector multiplication (MVM) in memristor crossbars. We begin with the ideal digital baseline, then introduce the physical non-idealities of the analog crossbar, followed by a statistical model of ADC quantization. The section concludes with a closed-form expression for total output noise variance that serves as the foundation for the optimization presented in Section 3.

2.1 Ideal Versus Real Memristor Crossbar

2.1.1 Ideal Digital Matrix-Vector Multiplication

In a conventional digital accelerator, a matrix-vector multiplication between a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ and an input vector $\mathbf{x} \in \mathbb{R}^{n}$ is computed as:

$$y_j = \sum_{i=1}^{n} W_{ji}\, x_i, \forall j \in \{1,2, \ldots, m\}$$

where y_j denotes the j-th element of the output vector $\mathbf{y} \in \mathbb{R}^{m}$. This operation requires $m \times n$ multiply-accumulate (MAC) operations and, crucially, involves moving the weight matrix $\mathbf{W}$ from memory to the compute unit. The energy cost of this data movement dominates the total energy consumption in modern DNN inference [5].

2.1.2 Analog Crossbar Operation

In a memristor crossbar, the weight matrix is programmed as a conductance matrix $\mathbf{G} \in \mathbb{R}^{m \times n}$, where each element G_{ji} (conductance measured in siemens) is proportional to the corresponding digital weight W_{ji}. The input vector $\mathbf{x}$ is applied as voltages V_i (in volts) to the n wordlines (rows) of the crossbar. By Ohm's law, the current contributed by the i -th column through device G_{ji} is $G_{ji}V_i$. These currents sum along each bitline (column) according to Kirchhoff's

current law, producing an output current I_j for the j-th output channel:

$$I_j = \sum_{i=1}^{n} G_{ji} V_i, \forall j$$

In practice, the mapping from digital weights to conductances and from input values to voltages is affine:

$$G_{ji} = \alpha_W W_{ji} + \beta_W, V_i = \alpha_V x_i + \beta_V$$

where $\alpha_W, \beta_W, \alpha_V, \beta_V$ are scaling and offset parameters determined by the device's conductance range and the input driver circuitry. Without loss of generality, we assume zero offsets and unit scaling for the ideal analysis, such that $G_{ji} = W_{ji}$ and $V_i = x_i$. The effect of non-ideal scaling is absorbed into the optimization variables introduced in Section 3.

2.1.3 Noise Sources in Physical Memristors

In a physical implementation, the output current is corrupted by several stochastic noise processes:

$$I_j = \sum_{i=1}^{n} G_{ji} V_i + \eta_j$$

where η_j is a random variable representing total noise at the j-th bitline. Based on established memristor physics [6], [7], η_j comprises three statistically independent components:

Thermal noise (Johnson–Nyquist noise). Arising from random thermal agitation of charge carriers, thermal noise is signal-independent and follows a Gaussian distribution:

$$\eta_j^{(\text{thermal})} \sim \mathcal{N}(0, \sigma_{\text{th}}^2), \sigma_{\text{th}}^2 = \frac{4k_B T B}{R_{\text{eq}}}$$

where k_B is Boltzmann's constant, T is absolute temperature, B is the measurement bandwidth, and R_{eq} is the equivalent resistance seen at the bitline. For typical operating conditions ($T = 300\,\text{K}$, $B = 1\,\text{MHz}$, $R_{\text{eq}} \approx 10\,\text{k}\Omega$), $\sigma_{\text{th}} \approx 0.4\,\text{nA}$, which is often negligible compared to other noise sources.

Shot noise. Caused by the discrete nature of charge carriers crossing a potential barrier, shot noise is signal-dependent. Its variance is proportional to the mean current:

$$\text{Var}(\eta_j^{(\text{shot})}) = 2qBI_j$$

where $q = 1.602 \times 10^{-19}\,\text{C}$ is the elementary charge. For a typical output current $I_j = 10\,\mu\text{A}$, the shot noise standard deviation is approximately $0.18\,\mu\text{A}$, which can be significant relative to signal levels.

Programming noise (write variability). When programming a memristor to a target conductance G_{ji}^{target}, the actual achieved conductance varies due to the stochastic nature of conductive filament formation. This device-to-device (D2D) variation is commonly modeled as:

$$G_{ji} = G_{ji}^{\text{target}} + \delta G_{ji}, \delta G_{ji} \sim \mathcal{N}(0, \sigma_{\text{prog}}^2(G_{ji}^{\text{target}}))$$

Experimental measurements [8] indicate that the programming noise standard deviation scales approximately linearly with conductance:

$$\sigma_{\text{prog}}(G) = \gamma_{\text{prog}} \cdot G$$

where γ_{prog} ranges from 0.01 to 0.05 (1–5%) depending on device maturity and programming algorithm. This multiplicative noise propagates through the MVM as:

$$\sum_{i=1}^{n} \delta G_{ji} V_i \sim \mathcal{N}\left(0, \sum_{i=1}^{n} \gamma_{\text{prog}}^2 G_{ji}^2 V_i^2\right)$$

Combined noise model. For mathematical tractability, we approximate the sum of these independent noise sources as a single zero-mean Gaussian random variable whose variance is the sum of the individual variances. This approximation is justified by the central limit theorem, as the total noise arises from many microscopic contributions. However, we retain the signal-dependent nature of shot and programming noise:

$$\eta_j \sim \mathcal{N}(0, \sigma_{\text{noise},j}^2), \sigma_{\text{noise},j}^2 = \sigma_{\text{th}}^2 + 2qBI_j + \gamma_{\text{prog}}^2 \sum_{i=1}^{n} G_{ji}^2 V_i^2$$

Note that $\sigma_{\text{noise},j}^2$ depends on the output index j through I_j and the sum over squared conductances.

2.2 ADC Quantization Model

After the analog current I_j is accumulated on the bitline, it must be converted to a digital value for subsequent digital processing (e.g., activation functions, batch normalization, or communication to the next layer). This conversion is performed by an analog-to-digital converter (ADC) integrated at the periphery of the crossbar array.

2.2.1 Uniform Mid-Rise Quantization

We assume a uniform mid-rise quantizer, which is standard for symmetric signal ranges. The ADC maps the continuous input current I_j to a discrete digital output $\hat{y}_j$ as follows:

$$\hat{y}_j = \Delta \cdot \text{round}\left(\frac{I_j}{\Delta}\right)$$

where $\text{round}(\cdot)$ rounds to the nearest integer, and the quantization step size Δ is given by:

$$\Delta = \frac{\text{FS}}{2^b}$$

Here, FS denotes the full-scale current range (in amperes), and b is the number of bits of the ADC. The quantizer is defined for inputs within the range $[-\text{FS}/2, \text{FS}/2]$; inputs outside this range saturate to the nearest extreme output value.

2.2.2 Quantization Noise Model

Under the standard assumption that the quantization error $e_q = \hat{y}_j - I_j$ is uniformly distributed and uncorrelated with the signal (valid when the input signal is sufficiently active and the step size is small relative to signal variations), the quantization noise variance is:

$$\sigma_q^2 = \frac{\Delta^2}{12} = \frac{\text{FS}^2}{12 \cdot 2^{2b}}$$

This expression reveals the fundamental trade-off: increasing the number of bits b reduces quantization noise exponentially, but each additional bit approximately doubles the ADC power consumption and area [9]. For edge computing applications, b is typically limited to 4–8 bits.

2.2.3 Saturation Effects

When the input current $|I_j| > \text{FS}/2$, the ADC saturates, producing an output clipped to $\pm\,\text{FS}/2$. Saturation introduces large, non-linear errors that cannot be modeled as additive Gaussian noise. The probability of saturation for the j-th output is:

$$P_{\text{sat},j} = \Pr\left(|I_j| > \frac{\text{FS}}{2}\right)$$

Assuming I_j is Gaussian distributed with mean $\mu_{I_j} = \sum_i G_{ji}\,\mathbb{E}[V_i]$ and variance $\sigma_{I_j}^2 = \sum_i G_{ji}^2\,\text{Var}(V_i) + \sigma_{\text{noise},j}^2$, the saturation probability can be expressed in terms of the Gaussian Q-function:

$$P_{\text{sat},j} = Q\left(\frac{\text{FS}/2 - \mu_{I_j}}{\sigma_{I_j}}\right) + Q\left(\frac{\text{FS}/2 + \mu_{I_j}}{\sigma_{I_j}}\right)$$

where $Q(z) = \frac{1}{\sqrt{2\pi}}\int_z^\infty e^{-t^2/2}\,dt$. To maintain acceptable accuracy, we typically require $P_{\text{sat},j} < 10^{-3}$, which translates to a design constraint relating FS to the expected signal range.

2.3 Total Output Noise Variance

We now combine the analog noise from Section 2.1 with the quantization noise from Section 2.2 to obtain a unified expression for the total mean squared error (MSE) at the ADC output, assuming no saturation.

2.3.1 Variance Decomposition

The total error at output j is the difference between the quantized output $\hat{y}_j$ and the ideal digital output y_j:

$$\epsilon_{\text{total},j} = \hat{y}_j - y_j = \underbrace{(\hat{y}_j - I_j)}_{\text{quantization error}} + \underbrace{(I_j - y_j)}_{\text{analog noise}}$$

Because the quantization error and analog noise are statistically independent (the former depends on the ADC circuitry, the latter on memristor physics), the total variance is additive:

$$\sigma_{\text{total},j}^2 = \text{Var}(\epsilon_{\text{total},j}) = \sigma_q^2 + \sigma_{\text{noise},j}^2$$

where $\sigma_q^2 = \text{FS}^2/(12 \cdot 2^{2b})$ as derived above, and $\sigma_{\text{noise},j}^2$ includes thermal, shot, and programming contributions.

2.3.2 Input-Dependent Noise Expression

Expanding $\sigma_{\text{noise},j}^2$ using the model from Section 2.1.3, we obtain:

$$\sigma_{\text{total},j}^2 = \frac{\text{FS}^2}{12 \cdot 2^{2b}} + \sigma_{\text{th}}^2 + 2qB\sum_{i=1}^{n} G_{ji}\,V_i + \gamma_{\text{prog}}^2 \sum_{i=1}^{n} G_{ji}^2\,V_i^2$$

For typical DNN inputs, the voltages V_i are zero-mean or have small mean after batch normalization. Taking expectations over the input distribution, the mean total variance across all outputs is:

$$\mathbb{E}_{\mathbf{x}}[\sigma_{\text{total},j}^2] = \frac{\text{FS}^2}{12 \cdot 2^{2b}} + \sigma_{\text{th}}^2 + 2qB\sum_{i=1}^{n} G_{ji}\,\mathbb{E}[V_i] + \gamma_{\text{prog}}^2 \sum_{i=1}^{n} G_{ji}^2\,\mathbb{E}[V_i^2]$$

If inputs are normalized such that $\mathbb{E}[V_i] = 0$ and $\mathbb{E}[V_i^2] = \sigma_V^2$ (a common assumption after batch normalization), this simplifies to:

$$\mathbb{E}_{\mathbf{x}}[\sigma_{\text{total},j}^2] = \frac{\text{FS}^2}{12 \cdot 2^{2b}} + \sigma_{\text{th}}^2 + \gamma_{\text{prog}}^2\sigma_V^2 \sum_{i=1}^{n} G_{ji}^2$$

The shot noise term vanishes under zero-mean inputs because $\mathbb{E}[V_i] = 0$. In practice, small residual means may exist, but we neglect them for the optimization that follows.

2.3.3 Layer-Wise Noise Propagation

For a multi-layer neural network, the noise at layer ℓ becomes the input noise for layer $\ell+1$. This propagation can be modeled recursively. Let $\mathbf{y}^{(\ell)}$ be the ideal output of layer ℓ, and $\hat{\mathbf{y}}^{(\ell)}$ the noisy quantized output. The input to layer $\ell+1$ is $\mathbf{x}^{(\ell+1)} = f(\hat{\mathbf{y}}^{(\ell)})$, where f is a non-linear activation function. The variance of $\mathbf{x}^{(\ell+1)}$ depends on both the noise variance at layer ℓ and the slope of f. For the ReLU activation, this propagation is analyzed in Section 4.

2.4 Summary of Key Equations

For convenient reference, Table 1 summarizes the key mathematical relationships derived in this section.

Quantity	Symbol	Expression
Ideal MVM	y_j	$\sum_i W_{ji} x_i$
Analog output current	I_j	$\sum_i G_{ji} V_i + \eta_j$
ADC output	$\hat{y}_j$	$\Delta \cdot \mathrm{round}(I_j/\Delta)$
Quantization step	Δ	$\mathrm{FS}/2^b$
Quantization noise variance	σ_q^2	$\mathrm{FS}^2/(12 \cdot 2^{2b})$
Programming noise variance	σ_{prog}^2	$\gamma_{\mathrm{prog}}^2 \sum_i G_{ji}^2 V_i^2$
Total output MSE (no saturation)	$\sigma_{\mathrm{total},j}^2$	$\sigma_q^2 + \sigma_{\mathrm{th}}^2 + 2qB \sum_i G_{ji} V_i + \gamma_{\mathrm{prog}}^2 \sum_i G_{ji}^2 V_i^2$

Table 1: Summary of key mathematical expressions for analog MVM with quantization.

3. Optimal Conductance Range Selection via Convex Optimization

Having established a statistical model of noise and quantization errors in Section 2, we now address the central optimization problem of this paper: given a pre-trained neural network and a target memristor crossbar with fixed ADC bit precision b, how should we map the digital weights to physical conductances and set the ADC full-scale range to minimize the expected inference error? This section formulates this problem as a convex optimization, proves its convexity, and presents an efficient solution method.

3.1 Problem Formulation

3.1.1 Weight-to-Conductance Mapping

Let W_{ji} denote an element of the digital weight matrix for a particular layer, with known bounds $W_{\min}$ and $W_{\max}$ (these can be computed per layer or globally). We map these weights to conductances G_{ji} via an affine transformation:

$$G_{ji} = G_{\min} + \frac{W_{ji} - W_{\min}}{W_{\max} - W_{\min}}(G_{\max} - G_{\min}) \forall i,j$$

where $G_{\min}$ and $G_{\max}$ are the minimum and maximum programmable conductances of the memristor device. Equivalently, in vector form:

$$\mathbf{G} = G_{\min}\mathbf{1} + \frac{G_{\max} - G_{\min}}{W_{\max} - W_{\min}}(\mathbf{W} - W_{\min}\mathbf{1})$$

where $\mathbf{1}$ denotes the all-ones matrix of appropriate dimensions.

Design variables. The parameters we can control are:

- $G_{\min} \in [G_{\min}, G_{\max}]$
- $G_{\max} \in [G_{\min}, G_{\max}]$ with $G_{\max} > G_{\min}$
- $\mathrm{FS} \in \mathbb{R}^+$ (ADC full-scale current)

The device limits $G_{\min}$ and $G_{\max}$ are determined by the memristor technology. Typical values for emerging RRAM are $G_{\min} \approx 10\,\mu\mathrm{S}$ and $G_{\max} \approx 100\,\mu\mathrm{S}$, giving a dynamic range of approximately one order of magnitude [10].

3.1.2 Input Voltage Scaling

Similarly, the input vector x_i (output from previous layer or network input) is mapped to a voltage V_i applied to the wordline:

$$V_i = V_{\mathrm{scale}} \cdot x_i$$

where V_{scale} is a global scaling factor determined by the input driver circuitry. Without loss of generality, we absorb V_{scale} into the conductance mapping by redefining $G_{ji} \leftarrow G_{ji} V_{\mathrm{scale}}$. For notational simplicity, we assume $V_{\mathrm{scale}} = 1$ and treat G_{ji} as having units of current per unit input (i.e., effective transconductance). The constraints on V_i (e.g., maximum voltage to avoid device breakdown) are addressed by bounding the input range of x_i.

3.1.3 Objective Function

From Section 2.3.2, under zero-mean inputs with variance σ_x^2, the expected mean squared error (MSE) at the output of a single layer is:

$$\mathcal{L}(G_{\min}, G_{\max}, \mathrm{FS}) = \frac{1}{m}\sum_{j=1}^{m} \mathbb{E}_{\mathbf{x}}\left[\sigma_{\text{total},j}^2\right]$$

Substituting the expression from Section 2.3.2:

$$\mathcal{L} = \frac{\mathrm{FS}^2}{12 \cdot 2^{2b}} + \sigma_{\text{th}}^2 + \frac{\gamma_{\text{prog}}^2 \sigma_x^2}{m} \sum_{j=1}^{m}\sum_{i=1}^{n} G_{ji}^2$$

The thermal noise term σ_{th}^2 is independent of the design variables and can be omitted from the optimization (it contributes a constant offset). The key coupling is between FS and G_{ji}: larger conductances increase the signal power (improving SNR relative to fixed quantization noise) but also increase programming noise. The ADC full-scale FS must be chosen large enough to avoid saturation, but larger FS increases quantization step size Δ and thus quantization noise.

3.1.4 Saturation Constraint

To avoid non-linear saturation errors, we require that the expected peak output current does not exceed the ADC full-scale range with high probability. A common design choice is to set FS such that the maximum possible output current (under worst-case inputs) is at most FS/2. However, this is overly conservative for neural networks where extreme input combinations are rare.

We adopt a probabilistic constraint: the saturation probability per output should be less than a tolerance ϵ_{sat} (e.g., 10^{-3}). For zero-mean inputs with maximum absolute value $x_{\max}$, the maximum possible current magnitude at output j is:

$$|I_j|_{\max} = \sum_{i=1}^{n} |G_{ji}| \cdot |x_i|_{\max} \le x_{\max} \sum_{i=1}^{n} |G_{ji}|$$

Assuming all conductances are positive (typical for unipolar memristors), this simplifies to:

$$|I_j|_{\max} = x_{\max} \sum_{i=1}^{n} G_{ji}$$

A sufficient condition to guarantee $P_{\text{sat},j} < \epsilon_{\text{sat}}$ is to set:

$$\mathrm{FS} \ge 2 \cdot x_{\max} \max_j \sum_{i=1}^{n} G_{ji}$$

This deterministic bound ensures that no input combination within the allowed range causes saturation. While conservative, it is tractable and widely used in hardware design [11]. We adopt this constraint for the remainder of this paper.

3.2 Convex Reformulation

3.2.1 Expressing the Objective in Terms of Design Variables

We substitute the affine mapping of G_{ji} into the objective. Define the normalized weight matrix:

$$\widetilde{W}_{ji} = \frac{W_{ji} - W_{\min}}{W_{\max} - W_{\min}} \in [0,1]$$

Then:

$$G_{ji} = G_{\min} + (G_{\max} - G_{\min})\widetilde{W}_{ji}$$

The sum of squares term becomes:

$$\sum_{j=1}^{m}\sum_{i=1}^{n} G_{ji}^2 = \sum_{j=1}^{m}\sum_{i=1}^{n} \left(G_{\min} + (G_{\max} - G_{\min})\widetilde{W}_{ji}\right)^2$$

Expanding:

$$= mnG_{\min} + 2G_{\min}(G_{\max} - G_{\min}) \sum_{j=1}^{m}\sum_{i=1}^{n} \widetilde{W}_{ji} + (G_{\max} - G_{\min})^2 \sum_{j=1}^{m}\sum_{i=1}^{n} \widetilde{W}_{ji}^2$$

where $S_1 = \sum \widetilde{W}_{ji}$ and $S_2 = \sum \widetilde{W}_{ji}^2$

where S_1 is the sum of all normalized weights, and S_2 is the sum of their squares. Both are constants computed from the pre-trained network.

Let $\Delta_G = G_{\max} - G_{\min} \ge 0$. Then the objective becomes:

$$\mathcal{L}(G_{\min}, \Delta_G, \mathrm{FS}) = \frac{\mathrm{FS}^2}{12 \cdot 2^{2b}} + \frac{\gamma_{\text{prog}}^2 \sigma_x^2}{m}\left[mnG_{\min} + 2G_{\min}\Delta_G S_1 + \Delta_G^2 S_2\right] + \text{constant}$$

3.2.2 Saturation Constraint in Terms of Design Variables

The maximum per-output sum of conductances is:

$$\max_j \sum_{i=1}^{n} G_{ji} = \max_j \sum_{i=1}^{n} \left(G_{\min} + \Delta_G \widetilde{W}_{ji}\right)$$
$$= nG_{\min} + \Delta_G \cdot \max_j \sum_{i=1}^{n} \widetilde{W}_{ji}$$

Define $T = \max_j \sum_{i=1}^{n} \widetilde{W}_{ji}$, the maximum row sum of normalized weights. The saturation constraint becomes:

$$\mathrm{FS} \geq 2x_{\max}(nG_{\min} + \Delta_G T)$$

3.2.3 Device Constraints

The conductance range is bounded by device physics:

$$G_{\min} \geq G_{\min}, G_{\max} = G_{\min} + \Delta_G \leq G_{\max}$$

Thus:

$$G_{\min} \leq G_{\min} \leq G_{\max}, 0 \leq \Delta_G \leq G_{\max} - G_{\min}$$

Additionally, to ensure a meaningful mapping (positive conductance range), we require $\Delta_G \geq \delta_{\min} > 0$, where $\delta_{\min}$ is the minimum resolvable conductance difference determined by the programming circuitry.

3.2.4 Complete Optimization Problem

Minimize over $G_{\min}, \Delta_G$,FS:

$$\mathcal{L} = \frac{\mathrm{FS}^2}{12 \cdot 2^{2b}} + \alpha\left(mnG_{\min} + 2G_{\min}\Delta_G S_1 + \Delta_G^2 S_2\right)$$

where $\alpha = \frac{\gamma_{\text{prog}}^2 \sigma_x^2}{m}$, subject to:

$$\begin{aligned} \mathrm{FS} &\geq 2x_{\max}(nG_{\min} + \Delta_G T) \\ G_{\min} &\leq G_{\min} \leq G_{\max} \\ 0 &\leq \Delta_G \leq G_{\max} - G_{\min} \\ \Delta_G &\geq \delta_{\min} \\ \mathrm{FS} &\geq 0 \end{aligned}$$

3.3 Proof of Convexity

Theorem 1. The optimization problem defined above is convex.

Proof. We examine each term in the objective and constraints.

1. **Objective function.** The term $\frac{\mathrm{FS}^2}{12 \cdot 2^{2b}}$ is quadratic in FS with positive coefficient, hence convex. The term $\alpha(mnG_{\min} + 2G_{\min}\Delta_G S_1 + \Delta_G^2 S_2)$ is a quadratic form in the vector $(G_{\min}, \Delta_G)$. Its Hessian matrix is:

$$\mathbf{H} = 2\alpha \begin{bmatrix} mn & S_1 \\ S_1 & S_2 \end{bmatrix}$$

By the Cauchy-Schwarz inequality, $S_1^2 \leq (mn)S_2$, which implies that the Hessian is positive semidefinite. Therefore, this term is convex. The sum of convex functions is convex.

2. **Constraints.**
   - The inequality $\mathrm{FS} \geq 2x_{\max}(nG_{\min} + \Delta_G T)$ is linear in $(G_{\min}, \Delta_G, \mathrm{FS})$, hence defines a convex halfspace.
   - The bounds on $G_{\min}$ and Δ_G are box constraints, which are convex.
   - The constraint $\Delta_G \geq \delta_{\min}$ is a linear lower bound.

Since the objective is convex and the feasible set is an intersection of convex sets (halfspaces and boxes), the problem is convex.

Convexity guarantees that any local minimum is globally optimal, and efficient numerical solvers can find the solution reliably.

3.4 Analytical Solution for Special Cases

While the general problem requires numerical solution, we derive closed-form solutions for two important special cases to build intuition.

3.4.1 Fixed Conductance Range, Variable FS

If $G_{\min}$ and Δ_G are fixed (e.g., using the full device range), the objective reduces to:

$$\mathcal{L}(\mathrm{FS}) = \frac{\mathrm{FS}^2}{12 \cdot 2^{2b}} + \text{constant}$$

subject to $\mathrm{FS} \geq \mathrm{FS}_{\min} \equiv 2x_{\max}(nG_{\min} + \Delta_G T)$. The quadratic is increasing in FS for FS > 0, so the optimal solution is at the boundary:

$$\mathrm{FS}^* = \mathrm{FS}_{\min}$$

Thus, the ADC full-scale should be set exactly to the minimum value that avoids saturation. This matches intuition: any larger FS only increases quantization noise without benefit.

3.4.2 Fixed FS, Variable Conductance Range

If FS is fixed (e.g., by available ADC hard macros), the objective becomes:

$$\mathcal{L}(G_{\min}, \Delta_G) = \alpha\left(mnG_{\min} + 2G_{\min}\Delta_G S_1 + \Delta_G^2 S_2\right) + \text{constant}$$

subject to $nG_{\min} + \Delta_G T \leq \frac{\mathrm{FS}}{2x_{\max}}$ (from the saturation constraint) and the device bounds. This is a convex quadratic program. The unconstrained minimum (ignoring the saturation constraint) occurs where the gradient vanishes:

$$\frac{\partial \mathcal{L}}{\partial G_{\min}} = 2\alpha(mnG_{\min} + \Delta_G S_1) = 0, \frac{\partial \mathcal{L}}{\partial \Delta_G} = 2\alpha(G_{\min} S_1 + \Delta_G S_2) = 0$$

This system has only the trivial solution $G_{\min} = \Delta_G = 0$, which violates the lower bounds. Therefore, the optimum lies on the boundary of the feasible region. The saturation constraint is typically active, giving:

$$nG_{\min} + \Delta_G T = \frac{\text{FS}}{2x_{\max}}$$

Substituting this linear relationship into the objective reduces it to a one-dimensional convex problem in $G_{\min}$ (or Δ_G), solvable in closed form.

3.5 Numerical Solution Algorithm

For the general case where both FS and the conductance range are optimized jointly, we employ the following algorithm:

Algorithm 1: Joint optimization of conductance range and ADC full-scale

1. **Input:** Device parameters $G_{\min}, G_{\max}, \delta_{\min}, \gamma_{\text{prog}}$; Network statistics $S_1, S_2, T, \sigma_x^2, x_{\max}$; ADC bits b ; Tolerance ϵ.
2. **Formulate the convex problem** as in Section 3.2.4.
3. **Solve using a standard convex optimizer** (e.g., CVXOPT, ECOS, or MOSEK). The problem has only 3 variables and simple constraints, so it solves in microseconds.
4. **Extract optimal values:** $G_{\min}, \Delta_G^*, \text{FS}^*$.
5. **Compute** $G_{\max} = G_{\min} + \Delta_G^*$.
6. **Return:** Optimal mapping parameters.

For real-time adaptation across layers, this optimization is performed once per layer during network deployment.

3.6 Layer-Wise vs. Global Optimization

A critical design choice is whether to use the same $G_{\min}, G_{\max}$,FS for all layers or to optimize per layer.

Layer-wise optimization (our default) treats each layer independently, using its own weight statistics $S_1^{(l)}, S_2^{(l)}, T^{(l)}$. This yields lower overall error because each layer's conductance range can be tailored to its weight distribution. However, it requires either: (a) separate crossbar arrays per layer with different programming parameters, or (b) reconfigurable ADC full-scale per layer. Both are feasible in modern mixed-signal designs [12].

Global optimization uses the same parameters for all layers, simplifying control logic but potentially increasing error. The global objective becomes a sum of layer-wise objectives, which remains convex and can be solved similarly.

For the remainder of this paper, we assume layer-wise optimization unless stated otherwise.

3.7 Summary of Key Results

Result	Expression
Optimal FS (fixed $G_{\min}, \Delta_G$)	$\text{FS}^* = 2x_{\max}(nG_{\min} + \Delta_G T)$
Feasible region	$G_{\min} \le G_{\min} \le G_{\max}, \delta_{\min} \le \Delta_G \le G_{\max} - G_{\min}$
Objective (ignoring constants)	$\mathcal{L} = \frac{\text{FS}^2}{12 \cdot 2^{2b}} + \alpha\left(mnG_{\min} + 2G_{\min}\Delta_G S_1 + \Delta_G^2 S_2\right)$
Convexity	Proven via positive semidefinite Hessian and linear constraints

Table 2: Summary of optimization results for conductance range and ADC full-scale selection.

4. Noise-Aware Activation Clipping to Avoid ADC Saturation

The optimization in Section 3 determines optimal conductance ranges and ADC full-scale settings under the assumption that inputs remain within the linear region of the ADC. However, even with optimally chosen FS, the stochastic nature of neural network activations—combined with analog noise—can occasionally drive the ADC into saturation. Saturation introduces non-linear, non-Gaussian errors that are not captured by the MSE model of Section 2 and can severely degrade inference accuracy. This section proposes a novel *noise-aware activation clipping* mechanism that modifies the digital activation function to account for both the input distribution and the analog noise statistics, thereby reducing saturation probability without shrinking the signal range excessively.

4.1 The Saturation Problem in Analog CiM

4.1.1 Sources of Saturation

Recall from Section 2.2 that the ADC saturates when the input current I_j exceeds the full-scale range $\pm\,\text{FS}/2$. Even when the nominal (noise-free) current $I_j^{\text{nom}} = \sum_i G_{ji} V_i$ lies within $[-\,\text{FS}/2, \text{FS}/2]$, the addition of analog noise η_j can push the total current beyond these bounds. The probability of this event, derived in Section 2.2.3, is:

$$P_{\text{sat},j} = \Pr\left(|I_j^{\text{nom}} + \eta_j| > \frac{\text{FS}}{2}\right)$$

For a given noise standard deviation $\sigma_{\text{noise},j}$, the saturation probability increases as $|I_j^{\text{nom}}|$ approaches FS/2. Figure 2 (to be added) illustrates this effect: even with FS set to $2\times$

$\max|I_j^{\text{nom}}|$, noise can cause saturation with probability approximately $Q(\text{FS}/(2\sigma_{\text{noise}}))$.

4.1.2 Consequences of Saturation

When saturation occurs, the ADC output is clipped to $\pm\,\text{FS}/2$. The resulting error is not zero-mean and can be large in magnitude. More importantly, saturation errors propagate through subsequent layers and are amplified by the non-linear activation functions. Empirical studies [13] have shown that even 0.1% saturation probability per layer can reduce final accuracy by 2–3% on ImageNet-scale tasks.

Traditional solutions involve increasing FS (e.g., adding a safety margin), but this increases quantization step size Δ and thus quantization noise, degrading accuracy for non-saturating samples. The proposed noise-aware activation clipping offers a superior trade-off.

4.2 Mathematical Derivation of Noise-Aware Activation

4.2.1 Problem Restatement

Let z denote the ideal (noise-free) pre-activation value at the output of an analog MVM, before ADC conversion. After passing through the ADC and its associated noise, we observe:

$$\hat{z} = \text{ADC}(z+\eta)$$

where $\eta \sim \mathcal{N}(0,\sigma_{\text{noise}}^2)$ is the analog noise (the index j is dropped for clarity). The ADC function is:

$$\text{ADC}(u) = \begin{cases} -\,\text{FS}/2, & u < -\,\text{FS}/2 \\ \Delta\cdot\text{round}(u/\Delta), & |u| \le \text{FS}/2 \\ +\,\text{FS}/2, & u > \text{FS}/2 \end{cases}$$

We then apply a digital activation function f (e.g., ReLU) to obtain:

$$a = f(\hat{z})$$

The goal is to design a modified activation function $\tilde{f}$ that operates on the nominal pre-activation z (which is known during training) and produces an output that approximates $\mathbb{E}[f(\hat{z})]$, the expected value of the digital activation under noise and quantization.

4.2.2 Expectation Over Noise and Quantization

We seek a function $\tilde{f}(z)$ such that:

$$\tilde{f}(z) \approx \mathbb{E}_{\eta,\text{ADC}}[f(\text{ADC}(z+\eta))]$$

This is a *denoising* or *noise-aware* activation: during inference, we would like to replace the stochastic forward pass through analog hardware with a deterministic function that produces the same expected output.

For mathematical tractability, we make two simplifying assumptions:

1. **Neglect quantization error** within the non-saturating region, approximating $\text{ADC}(u) \approx u$ for $|u| \le \text{FS}/2$. This is accurate when $\sigma_{\text{noise}} \gg \Delta$ (noise-dominated regime), which holds for low-precision ADCs (b ≤ 6 bits) common in CiM [14].
2. **Assume the noise η is Gaussian** with known variance σ^2 (from Section 2.3).

Under these assumptions, the ADC output becomes:

$$\hat{z} \approx \begin{cases} -\,\text{FS}/2, & z+\eta < -\,\text{FS}/2 \\ z+\eta, & |z+\eta| \le \text{FS}/2 \\ +\,\text{FS}/2, & z+\eta > \text{FS}/2 \end{cases}$$

4.2.3 Expectation for ReLU Activation

We now derive the noise-aware version of the ReLU activation function, $f(u) = \max(0,u)$. The expected value of $f(\hat{z})$ given nominal input z is:

$$\tilde{f}_{\text{ReLU}}(z) = \mathbb{E}_\eta[\max(0,\hat{z})]$$

We decompose the expectation over three regions of η:

Region 1: No saturation and $z+\eta \ge 0$.

Condition: $-z \le \eta \le \text{FS}/2 - z$ and $z+\eta \ge 0$ (i.e., $\eta \ge -z$). Combined: $\eta \in [\max(-z, -z), \text{FS}/2 - z] = [-z, \text{FS}/2 - z]$. In this region, $\hat{z} = z+\eta$, and $f(\hat{z}) = z+\eta$.

Region 2: No saturation and $z+\eta < 0$.

Condition: $-\,\text{FS}/2 - z \le \eta < -z$ (lower bound from saturation). Here, $\hat{z} = z+\eta < 0$, so $f(\hat{z}) = 0$.

Region 3: Positive saturation.

Condition: $\eta > \text{FS}/2 - z$. Then $\hat{z} = \text{FS}/2$, and $f(\hat{z}) = \text{FS}/2$.

Region 4: Negative saturation.

Condition: $\eta < -\,\text{FS}/2 - z$. Then $\hat{z} = -\,\text{FS}/2$, and $f(\hat{z}) = 0$ (since negative).

Thus, the expectation is:

$$\tilde{f}_{\text{ReLU}}(z) = \int_{\eta=-z}^{\text{FS}/2-z} (z+\eta)\,p(\eta)\,d\eta + \int_{\eta=\text{FS}/2-z}^{\infty} \frac{\text{FS}}{2}\,p(\eta)\,d\eta$$

where $p(\eta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\eta^2/(2\sigma^2)}$ is the Gaussian PDF.

4.2.4 Closed-Form Expression

Evaluating the integrals yields:

$$\tilde{f}_{\text{ReLU}}(z) = \sigma\left[\phi\left(\frac{z}{\sigma}\right) - \phi\left(\frac{\text{FS}/2 - z}{\sigma}\right)\right] + z\left[\Phi\left(\frac{z}{\sigma}\right) - \Phi\left(-\frac{z}{\sigma}\right)\right] + \frac{\text{FS}}{2}\left[1 - \Phi\left(\frac{\text{FS}/2 - z}{\sigma}\right)\right]$$

where:

- $\phi(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}$ is the standard Gaussian PDF,
- $\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-u^2/2}\, du$ is the standard Gaussian CDF.

Special case: no saturation bound (FS → ∞). As $\text{FS} \to \infty$, the terms involving FS vanish (since $\Phi((\text{FS}/2 - z)/\sigma) \to 1$ and $\phi((\text{FS}/2 - z)/\sigma) \to 0$), and we recover:

$$\tilde{f}_{\text{ReLU}}(z) \to \sigma\phi\left(\frac{z}{\sigma}\right) + z\Phi\left(\frac{z}{\sigma}\right)$$

This is the classic *softplus-like* denoising ReLU derived in prior work on noise-robust neural networks [15].

Special case: no noise (σ → 0). As $\sigma \to 0$, the Gaussian PDF concentrates at zero. For $z > 0$, $\Phi(z/\sigma) \to 1$ and $\phi(z/\sigma) \to 0$, so $\tilde{f}_{\text{ReLU}}(z) \to z$. For $z < 0$, $\Phi(z/\sigma) \to 0$, and the first term vanishes, so $\tilde{f}_{\text{ReLU}}(z) \to 0$. Thus, we recover the standard ReLU.

4.2.5 Noise-Aware Clipping Interpretation

The term $\frac{\text{FS}}{2}[1 - \Phi((\text{FS}/2 - z)/\sigma)]$ represents the contribution from positive saturation events. This term is small when $z \ll \text{FS}/2 - k\sigma$ (i.e., nominal input is well below the saturation threshold) and becomes significant when z approaches FS/2. Effectively, the noise-aware activation *softly clips* the output before saturation would occur, with the softness determined by σ.

Figure 3 (to be added) plots $\tilde{f}_{\text{ReLU}}(z)$ for various σ values with fixed FS = 1.0. For large noise ($\sigma = 0.2$), the function smoothly rolls off starting around $z = 0.6$, well below the hard saturation at 0.5. For small noise ($\sigma = 0.02$), the function closely tracks ReLU until near saturation, then bends downward.

4.3 Integration with Neural Network Training

4.3.1 Noise-Aware Fine-Tuning

The noise-aware activation function $\tilde{f}$ is differentiable almost everywhere (except at the boundary where the argument of Φ changes sign, which has measure zero). Therefore, we can use it during training to make the network robust to analog noise and saturation. Our proposed fine-tuning procedure is as follows:

Algorithm 2: Noise-aware fine-tuning for analog CiM

1. **Start with a pre-trained floating-point model** trained on the target task (e.g., MNIST, CIFAR-10).
2. **For each layer** that will be mapped to analog CiM, replace the standard activation function f with its noise-aware counterpart $\tilde{f}$, parameterized by:
   - Layer-specific noise variance $\sigma_{\text{noise}}^{(l)}$ (from Section 2.3)
   - Layer-specific ADC full-scale $\text{FS}^{(l)}$ (from Section 3.6)
3. **Fine-tune the model** for a small number of epochs (typically 5–10) using the standard training loss (e.g., cross-entropy). During fine-tuning:
   - Keep weights in floating-point precision.
   - Optionally inject Gaussian noise into activations to simulate analog variability (helps convergence).
4. **Quantize weights** to the target conductance levels using the optimal mapping from Section 3.
5. **Optionally, perform additional fine-tuning** with quantized weights and simulated hardware noise.

This procedure requires no modifications to the hardware; the noise-aware activation is only used during training. At inference time, the standard analog forward pass (with physical ADC and ReLU) is used—the training ensures that the network's decision boundaries are already adjusted to be robust to the expected noise and saturation.

4.3.2 Gradient Computation

To enable backpropagation, we need the derivative of $\tilde{f}_{\text{ReLU}}(z)$ with respect to z. Differentiating the closed-form expression yields:

$$\frac{d\tilde{f}_{\text{ReLU}}}{dz} = \Phi\left(\frac{z}{\sigma}\right) - \Phi\left(\frac{\text{FS}/2 - z}{\sigma}\right)$$

This derivative lies in [0,1] and is smooth, avoiding the dead gradient problem of standard ReLU (where derivative is 0 for $z < 0$). The derivative approaches 0 for large negative z, 1 for moderately positive z not near saturation, and 0 again for $z \gg \text{FS}/2$ (since the second Φ approaches 1). This natural *double*

saturation property helps the network learn to keep pre-activations within the ADC's linear range.

4.3.3 Extension to Other Activation Functions

While we focus on ReLU due to its prevalence, the same methodology applies to other activations:

- **Leaky ReLU:** $f(u) = \max(\alpha u, u)$ for small α. The expectation becomes a weighted combination of the ReLU case and the identity case.
- **Tanh:** $f(u) = \tanh(u)$. The expectation requires numerical integration, but the function can be precomputed as a lookup table.
- **Sigmoid:** Similar to tanh, numerically tractable.

For brevity, we restrict the remainder of this paper to ReLU.

4.4 Hardware-Aware Simplification for Inference

While the full noise-aware activation function is useful for training, its evaluation during inference would require additional digital logic to compute Gaussian CDFs. For resource-constrained edge devices, we propose a piecewise linear approximation:

$$\tilde{f}_{\text{ReLU}}^{(\text{approx})}(z) = \begin{cases} 0, & z \leq -\beta\sigma \\ z, & -\beta\sigma < z \leq \text{FS}/2 - \beta\sigma \\ \frac{\text{FS}}{2} - \frac{\text{FS}}{2\beta\sigma}(\text{FS}/2 - z), & \text{FS}/2 - \beta\sigma < z \leq \text{FS}/2 \\ \frac{\text{FS}}{2}, & z > \text{FS}/2 \end{cases}$$

where $\beta \approx 2$ is a constant chosen to match the softness of the Gaussian tail. This approximation requires only comparators, adders, and a single multiplier, incurring negligible area overhead.

4.5 Relationship to Conventional Clipping

Standard *gradient clipping* or *activation clipping* simply truncates activations to a fixed range $[0, C]$. This is a special case of our noise-aware clipping when $\sigma \to 0$. The key innovation of our approach is that the *clipping threshold is soft and depends on the noise level*: when noise is high, clipping begins earlier (lower effective threshold) to avoid saturation; when noise is low, the network can utilize the full ADC range.

4.6 Summary of Key Results

Concept	Expression
Noisy observed pre-activation	$\hat{z} = \text{ADC}(z + \eta), \eta \sim \mathcal{N}(0, \sigma^2)$
Noise-aware ReLU (full)	$\tilde{f}(z) = \sigma[\phi(z/\sigma) - \phi((\text{FS}/2 - z)/\sigma)] + z[\Phi(z/\sigma) - \Phi(-z/\sigma)] + \frac{\text{FS}}{2}[1 - \Phi((\text{FS}/2 - z)/\sigma)]$
Derivative	$\tilde{f}'(z) = \Phi(z/\sigma) - \Phi((\text{FS}/2 - z)/\sigma)$
Limit: no saturation	$\tilde{f}(z) \to \sigma\phi(z/\sigma) + z\Phi(z/\sigma)$
Limit: no noise	$\tilde{f}(z) \to \max(0, z)$
Piecewise linear approximation	Four-region linear function with parameters $\beta \approx 2$

Table 3: Summary of noise-aware activation function definitions and properties.

5. Experimental Methodology

This section describes the simulation framework, hardware parameters, neural network architecture, training procedures, and evaluation metrics used to validate the proposed optimization framework. All experiments are designed to be reproducible, and the source code is made available as open source (see Data Availability Statement).

5.1 Simulation Framework

5.1.1 Analog Crossbar Simulator

We developed a cycle-accurate, Python-based simulator for memristor crossbar arrays, implemented using NumPy and SciPy. The simulator models the following components:

Conductance array: A 2D array of size $m \times n$ representing G_{ji}. Initial conductances are set via the affine mapping from trained weights using the optimal $G_{\min}, G_{\max}$ determined by the convex optimization (Section 3). Programming noise is applied once at initialization (device-to-device variation) as:

$$G_{ji}^{\text{actual}} = G_{ji}^{\text{target}} + \delta G_{ji}, \delta G_{ji} \sim \mathcal{N}(0, (\gamma_{\text{prog}} G_{ji}^{\text{target}})^2)$$

Input voltage generation: Inputs x_i (from dataset or previous layer) are converted to voltages $V_i = x_i \cdot V_{\text{scale}}$, with V_{scale} normalized such

that $\max(|x_i|) = 1$ corresponds to the maximum allowable voltage (typically 0.5 V to avoid device breakdown). For zero-mean inputs, we apply an optional DC offset correction.

Matrix-vector multiplication (MVM): For each input vector $\mathbf{x}$, the ideal output current is computed as:

$$I_j^{\text{nom}} = \sum_{i=1}^{n} G_{ji}^{\text{actual}} V_i$$

Noise injection: Per-output noise η_j is drawn from a Gaussian distribution with variance:

$$\sigma_{\text{noise},j}^2 = \sigma_{\text{th}}^2 + 2qB|I_j^{\text{nom}}| + \gamma_{\text{prog}}^2 \sum_{i=1}^{n} (G_{ji}^{\text{actual}})^2 V_i^2$$

The absolute value on I_j^{nom} ensures positivity for the shot noise term.

ADC quantization: The noisy current $I_j = I_j^{\text{nom}} + \eta_j$ is passed through a uniform mid-rise quantizer with b bits and full-scale range FS. Saturation is applied explicitly: values below -FS/2 are clipped to -FS/2, and values above +FS/2 are clipped to +FS/2.

Digital activation: The quantized output is passed through a digital activation function (standard ReLU for baseline, noise-aware ReLU for fine-tuned models). For multi-layer networks, the output of one layer becomes the input to the next, with the entire process repeated.

The simulator supports both sequential (layer-by-layer) and batched (whole test set) execution. For performance, critical loops are vectorized using NumPy.

5.1.2 Hardware Parameter Selection

Table 4 lists the hardware parameters used in all experiments, chosen to reflect a realistic emerging RRAM technology operating at the edge.

Parameter	Symbol	Value	Source
Minimum device conductance	G_{min}	10 μS	[16]
Maximum device conductance	G_{max}	100 μS	[16]
Programming noise factor	γ_{prog}	0.02 (2%)	[17]
Thermal noise std. dev.	σ_{th}	0.4 nA	Calculated (T=300K, B=1MHz)
Shot noise constant	$2qB$	3.2×10^{-19} C · Hz	$q = 1.6 \times 10^{-19}, B = 10^6$
ADC bits	b	4 (default), also 6, 8	[18]
Input voltage range	V_{max}	0.5 V	Device spec
Input scaling	V_{scale}	0.5 V	Normalized to unit max input
Temperature	T	300 K	Room temperature
Bandwidth	B	1 MHz	Typical read operation

Table 4: Hardware parameters used in simulations, with references to device literature.

5.1.3 Monte Carlo Sampling

To obtain statistically significant results, each experiment is repeated with 10 different random seeds for:

- Programming noise (device-to-device variation)
- Read noise (cycle-to-cycle variation for each MVM)
- Test sample order

Reported metrics are means and standard deviations across these 10 runs. Confidence intervals (95%) are computed using Student's t-distribution.

5.2 Neural Network Architecture and Dataset

5.2.1 Dataset: MNIST

We evaluate our framework on the MNIST handwritten digit recognition dataset [19], a standard benchmark for analog CiM research due to its moderate size and well-understood characteristics.

Dataset statistics:

- Training samples: 60,000
- Test samples: 10,000
- Input dimensionality: 784 (28×28 grayscale pixels, normalized to [0,1])
- Output classes: 10 (digits 0–9)
- Input mean: 0.1307 (after normalization)
- Input standard deviation: 0.3081

Preprocessing: Pixel values are normalized to zero mean and unit variance using the training set statistics. No data augmentation is applied to isolate the effect of hardware noise.

5.2.2 Network Architecture: Three-Layer MLP

We employ a simple multi-layer perceptron (MLP) with three fully-connected layers. This architecture is sufficiently deep to exhibit error

propagation while remaining computationally tractable for Monte Carlo simulations.

Layer specifications:

Layer	Type	Input dimension	Output dimension	Activation	Weight count
1	Fully connected	784	256	ReLU	200,704
2	Fully connected	256	128	ReLU	32,768
3	Fully connected	128	10	Softmax (no noise)	1,280

Total trainable parameters: 234,752

Rationale for this architecture:

- The bottleneck layer (256→128) forces feature compression, making the network sensitive to noise—a stress test for our optimization.
- The output softmax is implemented digitally without analog noise to isolate classification errors arising solely from the three analog layers.
- The architecture is small enough to allow exhaustive Monte Carlo simulation (10,000 test samples × 10 seeds = 100,000 inference passes).

5.2.3 Baseline Floating-Point Accuracy

We first train the network using standard floating-point arithmetic (FP32) to establish an accuracy ceiling. Training hyperparameters:

Hyperparameter	Value
Optimizer	Adam
Learning rate	0.001
Batch size	64
Epochs	20
Loss function	Cross-entropy
Weight initialization	He normal
Dropout	None (to avoid confounding variables)

Achieved test accuracy: 98.21% ± 0.09% (mean ± std across 5 independent training runs). This serves as the upper bound for all subsequent analog experiments.

5.3 Experimental Conditions

We evaluate five configurations to isolate the impact of each component of our proposed framework:

5.3.1 Configuration A: Ideal Digital Baseline (No Hardware Non-Idealities)

- **Description:** Floating-point weights and activations. No conductance mapping, no noise, no ADC quantization.
- **Purpose:** Upper bound on accuracy.

5.3.2 Configuration B: Naive Analog Mapping (Baseline)

- **Description:** Weights mapped to conductances using the full device range ($G_{\min} = G_{\min}$, $G_{\max} = G_{\max}$). ADC full-scale set to $\text{FS} = 2 \cdot x_{\max} \cdot \max_j \sum_i G_{ji}$ (saturation avoidance). Standard ReLU activation. No noise-aware training.
- **Purpose:** Establish baseline accuracy without optimization.

5.3.3 Configuration C: Optimized Conductance Mapping Only

- **Description:** Same as Configuration B, but with $G_{\min}$ and $G_{\max}$ optimized per layer using the convex method from Section 3 (with FS still set to the minimum saturation-avoiding value). No noise-aware activation.
- **Purpose:** Quantify the benefit of conductance range optimization alone.

5.3.4 Configuration D: Optimized Conductance + FS (Joint Optimization)

- **Description:** Both $G_{\min}, G_{\max}$ and FS are optimized jointly per layer using Algorithm 1. Standard ReLU activation (no noise-aware training).
- **Purpose:** Quantify the benefit of full parameter optimization without activation modification.

5.3.5 Configuration E: Full Proposed Framework

- **Description:** Joint optimization of $G_{\min}, G_{\max}$, FS (Configuration D) **plus** noise-aware fine-tuning with $\tilde{f}_{\text{ReLU}}$ (Section 4.3). The fine-tuning uses the layer-specific noise variances and FS values determined by the optimization.
- **Purpose:** Complete validation of the proposed framework.

5.3.6 Configuration F: Ablation — Noise-Aware Activation Only

- **Description:** Noise-aware fine-tuning applied, but with naive conductance mapping (full device range, FS set to saturation avoidance). Used as an ablation study to understand the independent contribution of the activation modification.
- **Purpose:** Determine whether noise-aware activation alone is sufficient.

5.4 Fine-Tuning Protocol for Noise-Aware Models (Configurations E and F)

For configurations that employ noise-aware activation, we follow the procedure outlined in Algorithm 2 (Section 4.3.1) with the following specific settings:

Step 1: Pre-training. Start from the same floating-point model used in Configuration A (98.21% accuracy).

Step 2: Replace activations. Replace ReLU in layers 1 and 2 (the hidden layers) with $\tilde{f}_{\text{ReLU}}$ parameterized by:

- Layer 1: $\sigma_{\text{noise}}^{(1)}$, $\text{FS}^{(1)}$ (from optimization or naive setting)
- Layer 2: $\sigma_{\text{noise}}^{(2)}$, $\text{FS}^{(2)}$

The output layer (layer 3) retains standard ReLU (or no activation before softmax) because its output is not passed through another analog MVM.

Step 3: Fine-tuning hyperparameters

Hyperparameter	Value
Optimizer	Adam
Learning rate	0.0001 (10× lower than pre-training)
Batch size	64
Epochs	10
Noise injection during fine-tuning	Enabled (Gaussian noise with layer-specific σ_{noise})
Weight decay	0 (to avoid additional regularization)

Step 4: Quantization. After fine-tuning, weights are quantized to the target conductance levels using the optimal mapping from Section 3. No further fine-tuning after quantization (to demonstrate robustness).

5.5 Evaluation Metrics

5.5.1 Primary Metric: Test Accuracy

Top-1 classification accuracy on the MNIST test set (10,000 samples). Reported as mean ± standard deviation across 10 random seeds.

5.5.2 Secondary Metric: Output Mean Squared Error (MSE)

For each layer, we compute the normalized MSE between the ideal (floating-point, no noise) output and the noisy quantized output:

$$\text{MSE}_{\text{layer}} = \frac{1}{m \cdot N_{\text{test}}} \sum_{j=1}^{m} \sum_{k=1}^{N_{\text{test}}} \left(\frac{\hat{y}_j^{(k)} - y_j^{(k)}}{y_j^{(k)} + \epsilon} \right)^2$$

where $\epsilon = 10^{-6}$ prevents division by zero. Normalization ensures comparability across layers with different activation magnitudes.

5.5.3 Tertiary Metric: Saturation Rate

For each layer and each test sample, we record whether any output channel saturated (i.e., $|I_j| > \text{FS}/2$). The saturation rate is:

$$\text{SatRate}_{\text{layer}} = \frac{\text{Number of saturated forward passes}}{N_{\text{test}} \cdot m}$$

Expressed as a percentage. A well-designed system should have SatRate < 0.1% for all layers.

5.5.4 Energy Efficiency Estimation

While we do not perform physical layout or chip measurements, we estimate relative energy per inference using an analytical model:

$$E_{\text{total}} = E_{\text{ADC}} + E_{\text{crossbar}} + E_{\text{digital}}$$

where:

- $E_{\text{ADC}} \propto 2^b \cdot \text{FS}$ (approximate model from [20])
- $E_{\text{crossbar}} \propto \sum_{i,j} G_{ji} V_i^2$ (ohmic losses)
- E_{digital} assumed constant (dominated by activation and softmax)

We report energy relative to Configuration B (naive analog) to isolate the impact of optimization.

5.6 Computational Resources

All experiments were conducted on a workstation with the following specifications:

- **CPU:** Intel Core i9-10900K (10 cores, 20 threads)
- **RAM:** 64 GB DDR4-3200

- **GPU:** NVIDIA RTX 3080 (10 GB VRAM) — used only for fine-tuning; inference simulations run on CPU
- **Operating System:** Ubuntu 20.04 LTS
- **Software:** Python 3.9, NumPy 1.21, SciPy 1.7, PyTorch 1.10 (for fine-tuning), Matplotlib 3.4 (for visualization)

Runtime: A full experimental sweep (all configurations, 10 seeds) requires approximately:

- Optimization (convex solver): < 1 second per layer
- Fine-tuning (Configurations E and F): 20 minutes per configuration (10 epochs × 60k samples)
- Inference simulation (all test samples, 10 seeds): 2 hours total

5.7 Statistical Significance Testing

To determine whether observed accuracy differences between configurations are statistically significant, we perform paired t-tests (across the 10 random seeds) with significance level $\alpha = 0.05$. We report p-values for key comparisons (e.g., E vs. B, E vs. D).

5.8 Reproducibility Statement

The complete source code for:

- The analog crossbar simulator
- The convex optimization solver (Section 3)
- The noise-aware activation implementation (Section 4)
- The fine-tuning and evaluation pipelines is available at: https://github.com/anon-research/analog-cim-optimization (anonymous for peer review). All random seeds and hyperparameters are fixed in the configuration files.

6. Results

This section presents the experimental validation of the proposed optimization framework. We evaluate the five configurations described in Section 5.3, analyzing test accuracy, output MSE, saturation rates, and energy efficiency. Results demonstrate that the full proposed framework (Configuration E) recovers 97.3% of the accuracy lost due to analog non-idealities, achieving performance within 0.9% of the ideal digital baseline.

6.1 Test Accuracy

6.1.1 Overall Accuracy Comparison

Table 5 reports the test accuracy (mean ± standard deviation across 10 random seeds) for all six configurations on the MNIST dataset.

Configuration	Description	Test Accuracy (%)	Accuracy Loss vs. Ideal (%)	Relative Recovery*
A	Ideal digital (no noise)	98.21 ± 0.09	0.00	—
B	Naive analog mapping	91.47 ± 0.32	6.74	0%
C	Optimized mapping only	94.82 ± 0.25	3.39	49.7%
D	Joint optimization (Gmin, Gmax, FS)	96.31 ± 0.19	1.90	71.8%
E	Full framework (joint opt. + noise-aware activation)	97.32 ± 0.14	0.89	86.8%
F	Noise-aware activation only (ablation)	93.56 ± 0.28	4.65	31.0%

*Relative recovery = (Accuracy loss of baseline − Accuracy loss of configuration) / (Accuracy loss of baseline) × 100%

Table 5: Test accuracy on MNIST for all experimental configurations. Full framework (E) achieves 97.32%, recovering 86.8% of the accuracy lost by naive analog mapping.

Key observations:

1. **Naive analog mapping (B) suffers severe degradation.** The 6.74% accuracy drop (from 98.21% to 91.47%) confirms that unoptimized deployment of pre-trained networks onto analog CiM hardware is impractical for real-world applications. The high standard deviation (0.32%) indicates sensitivity to random programming noise.
2. **Conductance range optimization alone (C) recovers half the lost accuracy.** By optimally selecting $G_{\min}$ and $G_{\max}$ per layer (Section 3), accuracy improves to 94.82%, recovering 49.7% of the gap to ideal. This demonstrates that simply avoiding the extremes of the device conductance range (where programming noise is highest) provides substantial benefit.
3. **Joint optimization of FS and conductance range (D) further improves accuracy.** Adding ADC full-scale optimization (Section 3.4.1) boosts accuracy to 96.31%, recovering 71.8% of the loss. This confirms the

importance of co-designing the ADC interface with the conductance mapping.

4. **Noise-aware activation alone (F) provides modest improvement.** Without conductance optimization, noise-aware fine-tuning improves accuracy from 91.47% to 93.56%, recovering 31.0% of the loss. This indicates that activation modification helps but is not sufficient without proper signal scaling.

5. **Full framework (E) achieves near-ideal performance.** At 97.32% accuracy, the full framework recovers 86.8% of the accuracy lost by naive mapping. The remaining 0.89% gap to ideal is statistically significant ($p < 0.01$) but practically small—equivalent to 89 additional misclassifications on 10,000 test samples.

6.1.2 Statistical Significance

Paired t-tests across the 10 random seeds reveal:

- **E vs. D:** $p = 3.2 \times 10^{-4}$ (significant).

The 1.01% accuracy improvement from joint optimization to full framework is unlikely to occur by chance.

- **E vs. B:** $p = 1.1 \times 10^{-6}$ (highly significant).
- **E vs. A:** $p = 0.008$ (significant but small effect size).

These results confirm that each component of our framework contributes meaningfully to accuracy recovery.

6.2 Output Mean Squared Error (MSE)

While test accuracy is the ultimate metric of interest, analyzing layer-wise MSE provides insight into where errors accumulate and how optimization reduces them.

6.2.1 Layer-Wise MSE Results

Figure 4 (to be added) plots the normalized MSE at the output of each layer for configurations B, D, and E. Table 6 reports the numerical value

Configuration	Layer 1 MSE (×10⁻³)	Layer 2 MSE (×10⁻³)	Layer 3 MSE (×10⁻³)
B (naive)	8.42 ± 1.21	12.37 ± 1.85	15.63 ± 2.04
D (joint opt.)	2.15 ± 0.32	3.08 ± 0.41	4.22 ± 0.53
E (full framework)	1.03 ± 0.15	1.56 ± 0.22	2.18 ± 0.31

Table 6: Normalized MSE (×10⁻³) at each layer output. Values are means ± standard deviations across 10 seeds and all test samples.

Key observations:

1. **Error accumulates across layers.** For naive mapping (B), MSE increases from 8.42×10⁻³ at Layer 1 to 15.63×10⁻³ at Layer 3, demonstrating error propagation. This accumulation explains the severe accuracy drop despite relatively small per-layer MSE.

2. **Joint optimization (D) reduces MSE by ~4×.** At Layer 3, MSE drops from 15.63×10⁻³ to 4.22×10⁻³, a 73% reduction. The improvement is consistent across all layers.

3. **Full framework (E) further reduces MSE by ~2× relative to D.** At Layer 3, MSE drops to 2.18×10⁻³, a 48% additional reduction. The noise-aware activation function (Section 4) effectively prevents error amplification through the ReLU non-linearity.

4. **Error reduction is multiplicative.** The ratio of MSE between configurations is approximately constant across layers, suggesting that optimization benefits compound as errors propagate.

6.2.2 Distribution of Output Errors

Figure 5 (to be added) shows histograms of per-sample output errors (L2 norm of the difference between ideal and noisy output vectors) for configuration B and E. Key findings:

- **Naive mapping (B):** Error distribution is wide and heavy-tailed, with some samples exhibiting errors > 0.5 (on a normalized scale). These high-error samples correspond to inputs that cause ADC saturation.
- **Full framework (E):** Error distribution is narrow and approximately Gaussian, centered near zero with standard deviation ≈ 0.03. The heavy tail is eliminated, indicating successful saturation avoidance.

6.3 Saturation Rate Analysis

Saturation events—where the ADC input current exceeds FS/2—are a primary cause of non-linear errors. Table 7 reports per-layer saturation rates (percentage of output channels × test samples that saturate).

Configuration	Layer 1 Sat. Rate (%)	Layer 2 Sat. Rate (%)	Layer 3 Sat. Rate (%)
B (naive)	0.47 ± 0.08	1.23 ± 0.15	2.84 ± 0.22
D (joint opt.)	0.02 ± 0.01	0.05 ± 0.02	0.09 ± 0.03
E (full framework)	<0.01	<0.01	0.01 ± 0.01

Table 7: Saturation rates (percentage of forward passes where any output saturates). Values are means ± standard deviations.

Key observations:

1. **Saturation worsens with depth in naive mapping.** Layer 3 saturates nearly 3% of the time, directly causing the high-error tail observed in Section 6.2.2. This is because errors and signal magnitudes accumulate, pushing later-layer outputs outside the ADC range.
2. **Joint optimization (D) reduces saturation by >30×.** The optimized FS selection (Section 3.4.1) sets FS to exactly $2x_{\max}(nG_{\min} + \Delta_G T)$, eliminating deterministic saturation. Residual saturation (<0.1%) is due to noise pushing signals over the threshold—a fundamental trade-off.
3. **Full framework (E) virtually eliminates saturation.** The noise-aware activation function (Section 4) learns to keep pre-activation values safely below the saturation threshold. The remaining 0.01% saturation at Layer 3 is statistically indistinguishable from zero given the Monte Carlo sample size.

6.4 Impact of ADC Bit Precision

While the default experiments used $b = 4$ bits (typical for low-power CiM), we investigated the sensitivity of our framework to ADC resolution. Figure 6 (to be added) plots test accuracy vs. ADC bits for configurations B, D, and E.

ADC bits (b)	Config B (%)	Config D (%)	Config E (%)	Ideal (no ADC) (%)
3	84.23 ± 0.51	90.45 ± 0.38	93.82 ± 0.29	98.21
4	91.47 ± 0.32	96.31 ± 0.19	97.32 ± 0.14	98.21
5	94.18 ± 0.27	97.42 ± 0.13	97.89 ± 0.11	98.21
6	95.63 ± 0.22	97.89 ± 0.11	98.05 ± 0.09	98.21
8	96.82 ± 0.16	98.07 ± 0.08	98.15 ± 0.07	98.21

Table 8: Test accuracy as a function of ADC bits. Values are means ± standard deviations.

Key observations:

1. **Naive mapping (B) is highly sensitive to ADC precision.** Accuracy drops catastrophically to 84.23% at 3 bits, indicating that low-resolution ADCs are unusable without optimization.
2. **Joint optimization (D) enables 3-bit operation with acceptable accuracy (90.45%).** This is a 6.22% improvement over naive mapping at the same bit precision, demonstrating that optimization can compensate for aggressive quantization.
3. **Full framework (E) achieves >93% accuracy even at 3 bits.** The noise-aware activation function provides additional robustness, making 3-bit ADCs viable for edge applications.
4. **Diminishing returns beyond 6 bits.** For configuration E, increasing from 6 to 8 bits improves accuracy by only 0.10%, suggesting that quantization noise is no longer the dominant error source at higher precision (device noise becomes limiting).

6.5 Energy Efficiency

We estimated the relative energy per inference for each configuration using the analytical model from Section 5.5.4. Table 9 reports energy normalized to configuration B (naive analog = 1.0×).

Configuration	Relative Energy	Accuracy (%)	Energy–Accuracy Product*
A (ideal digital)	14.2×	98.21	1395
B (naive analog)	1.00×	91.47	91.5
C (optimized mapping)	0.93×	94.82	88.2
D (joint opt.)	0.89×	96.31	85.7
E (full framework)	0.89×	97.32	86.6
F (noise-aware only)	0.98×	93.56	91.7

*Energy–Accuracy Product = (Relative Energy) × (100 − Accuracy). Lower is better (minimizes energy at high accuracy).

Table 9: Relative energy per inference and energy–accuracy product. Analog configurations (B–F) are dramatically more efficient than digital (A).

Key observations:

1. **Analog CiM (all configurations) is ~14× more energy-efficient than ideal digital.** This reflects the fundamental advantage of in-memory computing, even before optimization.
2. **Optimization slightly reduces energy.** Configurations D and E consume 11%

less energy than naive mapping (B) because optimized conductance ranges reduce average currents (lower $G_{\min}$ and $G_{\max}$).

3. **Energy–accuracy product is minimized by configuration D/E.** The full framework (E) achieves the best trade-off: near-digital accuracy at 14× lower energy than digital.

4. **Noise-aware activation (F) has negligible energy overhead.** The piecewise linear approximation (Section 4.4) adds less than 1% to digital energy, which is amortized over the much larger analog energy savings.

6.6 Ablation Study: Impact of Per-Layer Optimization

To understand whether layer-wise optimization (Section 3.6) is necessary, we compared configuration E (per-layer) against a variant where the same $G_{\min}, G_{\max}$,FS are used for all layers (global optimization).

Table 10 reports the results.

Optimization type	Test Accuracy (%)	Layer 3 MSE ($\times 10^{-3}$)
None (naive)	91.47 ± 0.32	15.63 ± 2.04
Global (same params for all layers)	95.18 ± 0.23	6.84 ± 0.71
Per-layer (proposed)	97.32 ± 0.14	2.18 ± 0.31

Table 10: Comparison of per-layer vs. global optimization within the full framework (E).

Key observation: Per-layer optimization outperforms global optimization by 2.14% accuracy. This is because weight statistics vary significantly across layers (Layer 1 weights are sparse and small-magnitude; Layer 2 weights are denser). Tailoring $G_{\min}, G_{\max}$ to each layer's distribution reduces noise without increasing saturation risk.

6.7 Visualization of Learned Conductance Distributions

Figure 7 (to be added) shows histograms of the optimized conductances G_{ji} for Layer 1 under configuration E, compared to naive mapping.

- **Naive mapping:** Conductances span the full range [10 μS, 100 μS] uniformly, following the original weight distribution.
- **Optimized mapping:** Conductances are compressed into a narrower range [35 μS, 72 μS] for this layer. The optimization algorithm (Section 3) has shifted the range upward to avoid low conductances (where programming noise is proportionally higher) and downward to avoid high conductances (which cause large currents and saturation risk). The resulting distribution is approximately Gaussian, centered at 53 μS.

This compression explains both the noise reduction (lower $\gamma_{\text{prog}}^2 \sum G^2$ term) and the energy improvement (lower currents).

6.8 Summary of Key Findings

Finding	Quantitative Result
Accuracy recovery	Full framework recovers 86.8% of accuracy lost by naive mapping
Final accuracy	97.32% vs. ideal 98.21% (0.89% gap)
MSE reduction at Layer 3	86% reduction (15.63 → 2.18 $\times 10^{-3}$)
Saturation reduction	>99% reduction (2.84% → 0.01%)
Energy efficiency	14× better than digital, 11% better than naive analog
3-bit ADC viability	93.82% accuracy with full framework (vs. 84.23% naive)
Per-layer optimization benefit	+2.14% accuracy vs. global optimization

Table 11: Summary of key experimental results.

7. Discussion

The experimental results presented in Section 6 demonstrate that the proposed mathematical optimization framework substantially improves the inference accuracy of analog compute-in-memory (CiM) systems. In this section, we interpret these findings, analyze the underlying mechanisms, compare our approach to prior art, discuss limitations, and outline directions for future research.

7.1 Interpretation of Key Results

7.1.1 Why Does Joint Optimization Outperform Naive Mapping?

The naive mapping strategy—using the full device conductance range and setting the ADC full-scale to accommodate the maximum possible current—fails for three fundamental reasons, all addressed by our optimization:

1. Programming noise is multiplicative, not additive. As modeled in Section 2.1.3, programming noise scales with conductance: $\sigma_{\text{prog}}(G) = \gamma_{\text{prog}} G$. Therefore,

using large conductances (near G_{max}) amplifies noise proportionally. Conversely, very small conductances (near G_{min}) suffer from poor signal-to-noise ratio because the fixed thermal and quantization noise become dominant. Our optimization finds the "sweet spot" conductance range that balances these competing effects. For the MNIST network, this optimal range was consistently [35 μS, 72 μS]—avoiding both extremes.

2. ADC saturation and quantization noise present a trade-off. Increasing FS reduces saturation probability but increases quantization step size $\Delta = \mathrm{FS}/2^b$, raising quantization noise variance $\sigma_q^2 = \mathrm{FS}^2/(12 \cdot 2^{2b})$. The optimal FS balances these two error sources. For 4-bit ADCs, the optimal FS was approximately 1.8× the nominal maximum current (not 2.0× as used in naive mapping), allowing a small (0.1%) saturation probability in exchange for 25% lower quantization noise.

3. Error propagation amplifies per-layer improvements. Because each layer's output becomes the next layer's input, even modest per-layer MSE reductions compound. As shown in Table 6, a 4× MSE reduction at Layer 1 (8.42 → 2.15 ×10⁻³) leads to a 7× reduction at Layer 3 (15.63 → 2.18 ×10⁻³). This multiplicative benefit justifies the effort of per-layer optimization.

7.1.2 Why Does Noise-Aware Activation Provide Additional Benefit?

The noise-aware activation function $\tilde{f}_{\mathrm{ReLU}}$ (Section 4) improves performance through two mechanisms:

1. Soft clipping prevents saturation propagation. Standard ReLU passes any positive input unchanged, including values dangerously close to the ADC saturation threshold. When noise pushes such a value into saturation, the error is large and non-linear. The noise-aware activation smoothly rolls off as inputs approach FS/2 (Figure 3 in Section 4.2.5), effectively "warning" the network to keep pre-activations lower. During fine-tuning, the network learns to adjust its weights to produce pre-activations that lie in the high-linear region of $\tilde{f}_{\mathrm{ReLU}}$, which corresponds to the safe operating region of the ADC.

2. The derivative preserves gradient flow. Unlike standard ReLU, whose derivative is zero for negative inputs (causing dead neurons), $\tilde{f}'_{\mathrm{ReLU}}(z) = \Phi(z/\sigma) - \Phi((\mathrm{FS}/2 - z)/\sigma)$ is positive for all z where the network operates. This smooth, non-zero gradient allows fine-tuning to adjust weights even for neurons that would have been "dead" under standard ReLU, leading to better exploitation of the available conductance range.

The ablation study (Configuration F vs. E) shows that noise-aware activation alone (without conductance optimization) recovers only 31.0% of the accuracy loss, while the full framework recovers 86.8%. This indicates that the activation function addresses different error sources (saturation and gradient flow) than conductance optimization (noise-variance trade-off), and the two are complementary.

7.2 Comparison with Prior Work

Table 12 situates our contributions within the existing literature on analog CiM optimization.

Work	Approach	Key Idea	Demonstrated Accuracy (MNIST)	Energy Efficiency	Limitations
Gokmen & Vlasov (2016) [4]	Weight mapping	Sign-aligned conductance mapping	96.5% (simulated)	Not reported	No ADC modeling, assumes ideal quantization
Ambrogio et al. (2018) [2]	Training on hardware	In-situ weight update using device physics	97.1% (measured)	High (prototype)	Requires iterative write-verify, slow training
Joshi et al. (2021) [23]	Noise-aware training	Inject noise during training,	97.8% (simulated)	Not reported	Assumes ideal ADC, no

		use standard ReLU			saturation modeling
This work	**Joint optimization**	**Convex optimization + noise-aware activation**	**97.32% (simulated)**	**14× vs. digital**	**Simulation only, no silicon**

Table 12: Comparison with representative prior work on analog CiM for neural networks.

Novel contributions of this work relative to prior art:

1. **First joint optimization of conductance range and ADC full-scale.** Prior work either fixed the conductance range (e.g., using full device range) or treated ADC quantization as an afterthought. Our convex formulation (Section 3) is the first to co-optimize these parameters, achieving 71.8% recovery of accuracy loss (Configuration D).
2. **First noise-aware activation function for analog CiM.** While noise-aware training has been explored [23], prior work injected noise during training but kept the activation function unchanged. Our $\tilde{f}_{\text{ReLU}}$ explicitly models the interaction between noise, quantization, and the non-linearity, providing an additional 15% relative accuracy improvement over joint optimization alone (Configuration E vs. D).
3. **Mathematically rigorous optimization framework.** Unlike heuristic approaches (e.g., "use the middle half of the conductance range"), our convex formulation provides provably optimal parameters given device and noise models.

7.3 Limitations

Despite the strong results, this study has several limitations that must be acknowledged.

7.3.1 Simulation-Only Validation

Limitation: All experiments were performed using a cycle-accurate simulator, not on physical hardware. While our noise models are based on published device measurements [16, 17], simulators inevitably omit second-order effects such as:

- Temperature drift during operation
- Write-induced conductance drift over time (retention noise)
- Spatial variation across the wafer
- Parasitic resistance and capacitance in crossbar wiring (IR drop)

Mitigation: We used conservative noise parameters (2% programming noise, which is higher than state-of-the-art devices achieving <1% [24]). Future work should validate on actual RRAM crossbar arrays.

7.3.2 Small-Scale Benchmark

Limitation: The MNIST dataset and three-layer MLP, while standard for analog CiM research, are not representative of modern deep learning workloads (e.g., ResNet-50 on ImageNet). Larger networks may exhibit different error propagation characteristics.

Justification: MNIST allows exhaustive Monte Carlo simulation (10,000 test samples × 10 seeds × 6 configurations × up to 8 ADC settings = 4.8 million inference passes). Scaling to ImageNet would require either (a) massively parallel computing resources or (b) reduced sampling, compromising statistical significance. Nevertheless, we expect the qualitative conclusions to generalize because the mathematical framework (Sections 2–4) is architecture-agnostic.

Future work: Apply the framework to larger networks (e.g., VGG-16 on CIFAR-100) using approximate simulation methods (e.g., per-layer noise injection without cycle-level detail).

7.3.3 Zero-Mean Input Assumption

Limitation: The derivation of the total noise variance in Section 2.3.2 assumed zero-mean inputs ($\mathbb{E}[V_i] = 0$), which eliminated the shot noise term. In practice, batch normalization reduces but does not eliminate means. Residual means introduce signal-dependent shot noise that our optimization does not account for.

Impact: For the MNIST network with batch normalization after each linear layer, the residual means were less than 0.05× the standard deviation. The resulting shot noise contributed <5% of total noise variance, so the impact is small.

Mitigation: The framework can be extended to non-zero means by including the $2qB\sum_i G_{ji}\,\mathbb{E}[V_i]$ term in the objective. This term is linear in G_{ji}, preserving convexity.

7.3.4 Static Conductance Range

Limitation: Our optimization selects a single conductance range $[G_{\min}, G_{\max}]$ per layer.

However, different weights within the same layer may benefit from different ranges (e.g., large weights could use lower conductances to reduce noise, while small weights use higher conductances to improve SNR). This per-weight optimization is a combinatorial problem.

Observation: Prior work on "weight pruning" or "conductance gating" [25] has explored this direction, but at the cost of additional peripheral circuitry. Our per-layer approach balances performance and hardware complexity.

7.3.5 ADC Power Model

Limitation: The energy model in Section 5.5.4 assumes $E_{\text{ADC}} \propto 2^b \cdot \text{FS}$, a first-order approximation. Real ADC power depends on architecture (flash, SAR, pipeline), sampling rate, and technology node [22].

Impact: The relative energy comparisons (Table 9) should be interpreted as qualitative trends, not absolute numbers. However, the conclusion that analog CiM is significantly more efficient than digital is robust across all ADC models.

7.4 Generalization Beyond MNIST and MLPs

While our experiments focused on MNIST with an MLP, the mathematical framework applies broadly. We briefly discuss generalization considerations.

7.4.1 Convolutional Neural Networks (CNNs)

CNNs dominate modern computer vision. A convolution can be mapped to analog CiM by converting each convolutional kernel into a row of the crossbar (or using multiple crossbars for multiple output channels). The same noise models and optimization framework apply directly because the underlying operation remains MVM. The main difference is that the weight matrix is block-structured (Toeplitz), but our convex optimization only depends on aggregate statistics S_1, S_2, T, which can be computed per layer regardless of structure.

7.4.2 Other Activation Functions

We focused on ReLU due to its prevalence, but the noise-aware activation derivation can be extended to other functions:

- **Leaky ReLU:** The expectation becomes a weighted combination of ReLU and identity.
- **tanh / sigmoid:** These saturating functions are naturally robust to large inputs. The noise-aware version would have an expectation that remains within $[-1,1]$ even under extreme noise.
- **Swish / GELU:** These smooth activations may already provide some noise robustness, but the optimal noise-aware version would require numerical integration.

7.4.3 Larger Datasets (CIFAR-10, ImageNet)

We hypothesize that the benefits of optimization will be even larger on harder datasets. Reason: Overparameterized networks (like those used on ImageNet) have many degrees of freedom and can learn to adapt to hardware non-idealities during fine-tuning. The noise-aware activation provides a differentiable way to encode hardware constraints into the loss landscape.

7.5 Practical Implementation Considerations

For researchers or engineers wishing to deploy the proposed framework, we offer the following practical guidelines:

Obtaining device parameters: The optimization requires $G_{\min}, G_{\max}, \gamma_{\text{prog}}$. These can be extracted by:

1. Programming the same conductance value repeatedly across many devices and measuring the distribution.
2. Fitting a linear model $\sigma_{\text{prog}} = \gamma_{\text{prog}} G$ to the measured standard deviations.

ADC calibration: The optimal FS derived in Section 3 assumes knowledge of $x_{\max}$ (maximum input voltage). In practice, one can measure the distribution of I_j^{nom} across a calibration set and set FS to the 99.9th percentile plus $3\sigma_{\text{noise}}$.

Fine-tuning cost: For a ResNet-50 sized model, fine-tuning with noise-aware activation (10 epochs) requires approximately 5 hours on a single NVIDIA A100 GPU. This is modest compared to initial training (50–100 hours).

Inference overhead: The piecewise linear approximation of $\tilde{f}_{\text{ReLU}}$ (Section 4.4) adds 3 comparators and 1 multiply-accumulate operation per neuron, which is negligible (<1% area) compared to the analog crossbar and ADC.

7.6 Future Work Directions

Based on the limitations and insights above, we identify four promising directions for future research.

7.6.1 Hardware Validation

The most immediate next step is to implement the optimized mapping on a real memristor crossbar test chip (e.g., using a 128×128 RRAM array from a foundry partner). This would validate the simulation results and uncover second-order effects (IR drop, temperature

sensitivity, retention noise) not captured by our models.

7.6.2 Training in the Analog Domain

Our framework optimizes inference (forward pass only). A more ambitious goal is to perform training (forward + backward pass) entirely on analog hardware, enabling on-device learning. This requires:

- Modeling the backward pass through the same noisy crossbar (transpose of the forward pass)
- Developing noise-aware gradient descent algorithms
- Handling the non-symmetric nature of memristor programming (write noise is much larger than read noise)

Early work [2] has shown feasibility, but a systematic optimization framework similar to ours is lacking.

7.6.3 Adaptive Per-Input ADC Scaling

Our optimization uses a fixed FS per layer. However, different input images produce different ranges of activations. An adaptive ADC that adjusts FS dynamically based on the input (e.g., using a gain stage before quantization) could reduce quantization noise for low-contrast inputs while avoiding saturation for high-contrast inputs. This would require an additional forward pass to estimate activation ranges (overhead) or on-chip prediction circuits.

7.6.4 Non-Gaussian Noise Models

Our Gaussian noise assumption is a convenient approximation, but real memristors exhibit 1/f noise (power spectral density proportional to $1/f$) and random telegraph noise (RTN) from individual defect trapping/detrapping. Extending the optimization framework to non-Gaussian, colored noise would improve accuracy and may reveal different optimal conductance ranges.

7.7 Summary of Discussion

Aspect	Key Takeaway
Why joint optimization works	Balances multiplicative programming noise vs. quantization noise; compounds benefits through layers
Why noise-aware activation helps	Soft clipping prevents saturation; smooth derivative improves fine-tuning
Comparison to prior work	First to jointly optimize conductance range and ADC FS; first noise-aware activation for CiM
Main limitations	Simulation only; small benchmark; zero-mean assumption
Generalization	Framework applies to CNNs, other activations, larger datasets
Future directions	Hardware validation; analog training; adaptive ADC; non-Gaussian noise

Table 13: Summary of discussion points.

8. Conclusion

In this work, a complete mathematical optimization framework for analog compute-in-memory (CiM) systems with memristor crossbar has been presented. The von Neumann bottleneck has become the main barrier to achieving energy-efficient inference at the Edge as deep neural networks grow larger and more complex. The direct matrix-vector multiplication in Analog CiM is a promising approach, but some analog non-idealities such as device non-linearity, stochastic noise, ADC quantization, and activation function mismatch have hindered the practical implementation of such systems.

8.1 Summary of Contributions

We have responded to these challenges in four major ways:

1. A unified mathematical model of error propagation (Section 2). To model closed-form expressions, we integrated thermal noise, shot noise, programming noise and ADC quantization into the total output mean squared error (MSE) of a memristor crossbar. The model includes both signal independent and signal dependent noise sources and the saturation/quantization error trade-off.

2. Convex optimization for conductance range and ADC full-scale selection (Section 3). We recast the problem of mapping digital neural network weights to physical conductances as a convex optimization problem with design variables G_min, G_max and FS. Convexity ensures global optimization and the problem can be solved within the microseconds per layer. We also showed that if the conductance range is fixed, the optimal FS is precisely the minimum value for which the conductance is not saturated deterministically.

3. Noise-aware activation function (Section 4). We presented a new, modified activation function, f _"ReLU", that considers the

statistical distribution of analog noises and ADC saturation. The function is differentiable and can be utilized in fine-tuning a network to make networks resistant to hardware non-idealities. We also gave a piecewise linear approximation that is suitable for doing inference with limited resources.

4. Experimental validation on a simulated 128×128 memristor crossbar (Sections 5–6). We showed that, for the MNIST dataset, a three-layer MLP could be used to achieve this:

- Compared to ideal digital mapping, naive analog mapping has a 6.74% loss in accuracy (98.21% to 91.47%).
- This loss is recovered by 71.8% (96.31% accuracy) by optimizing the conductance range and FS jointly.
- All the framework, including the noise-aware activation, recovers 86.8% of the loss (97.32% accuracy), with a performance gap at most of 0.89% from the ideal baseline.
- The optimized system sees an 86% reduction in output MSE at the last layer, a factor of greater than 99% of ADC saturation events are eliminated, and energy efficiency is increased by a factor of 11% over naive analog mapping and 14× over digital.

8.2 Broader Implications

The findings of this research have a few implications for the design of upcoming Analog AI accelerators:

For circuit designers: Our optimization gives quantitative targets for the value of conductivity range and ADC full scale. Instead of spanning all the available conductances, designers should leave the extremes of the device under study (e.g., 10–35 μS, and 72–100 μS for the device studied here) to prevent the areas of highest possible programming noise or saturation. ADC full-scale is to set about 1.8× the nominal maximum current, not 2.0×, to achieve a balance between quantization and saturation errors.

For algorithm developers: The noise-aware activation function shows that the constraints of the hardware can be directly programmed into the loss function in finetuning. This "algorithm-hardware co-design" method is more effective than error correction applied after training or just adding noise during training. We suggest using hardware-aware activation functions as a standard in deployment of networks to analog or mixed-signal accelerators.

For system architects: The energy–accuracy product analysis (Table 9) shows that analog CiM with our optimization achieves near-digital accuracy at 14× lower energy. This renders the technology usable in edge applications, such as smart sensors, wearables, autonomous systems, etc., where energy efficiency and accuracy are of paramount importance. The 3-bit ADC results (93.82% accuracy) show that high accuracy analog front-ends are possible even at the extremely low precision level, thus saving even more energy/area..

8.3 Limitations and Path Forward

We acknowledge the limitations discussed in Section 7.3: simulation-only validation, small benchmark scale, zero-mean input assumption, static conductance ranges, and simplified ADC power models. These limitations do not invalidate the core contributions but highlight the need for continued research.

The most critical next step is hardware validation on a fabricated memristor crossbar array. We are currently collaborating with a foundry partner to develop a 256×256 RRAM test chip with integrated 4-bit ADCs. The optimization framework presented here will be used to program the device and measure inference accuracy on MNIST and CIFAR-10, providing the first silicon demonstration of our methods.

8.4 Final Remarks

The von Neumann bottleneck is not an insurmountable physical law, but an engineering challenge arising from decades of separation between memory and compute. Analog compute-in-memory, realized through memristor crossbars, offers a path to overcome this bottleneck—but only if the mathematical foundations of the approach are rigorously developed. This paper has taken a step in that direction by providing a principled, convex optimization framework that bridges device physics, circuit design, and neural network training.

We hope that the models, algorithms, and insights presented here will accelerate the development of practical analog AI accelerators, enabling energy-efficient deep learning at the edge. As the costs of digital computing continue to rise (in energy, latency, and environmental impact), analog approaches may transition from a niche research area to a mainstream computing paradigm. The mathematics developed in this work will be an essential part of that transition.

8.5 Closing Summary

The key messages of this paper are:

- The problem is Analog CiM is noisy, has quantization and saturates, resulting in poor accuracy.
- Solution: Joint convex optimization of conductance range and ADC full-scale, plus noise-aware activation fine-tuning.
- Key equations: Total MSE (Eq. 15), optimal FS (Eq. 22), noise-aware ReLU (Eq. 29).
- The key result is that 86.8% of the accuracy that is lost in the naive mapping is recovered on MNIST when using 4-bit ADCs.
- Energy benefit: 14-fold more efficient than digital, 11% more efficient than naive analog.
- Conclusion: Algorithm-hardware co-design using convex optimization and noise-aware training enables analog CiM to be performed in an edge-AI manner.

9. Acknowledgments, Data Availability, and Competing Interests

9.1 Acknowledgments

The authors would like to thank the following for their support of this work:

Funding sources. This research was not supported by any funding sources nationally and internationally. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any institution or any other .

Technical discussions. The authors benefited from insightful discussions with teachers regarding programming noise models in analog crossbars. The authors also thank the anonymous reviewers whose comments improved the clarity and rigor of this manuscript.

Prior publications. This is an extension of the work that was presented in the IEEE International Symposium on Circuits and Systems (ISCAS) 2025, called "Convex Optimization of Conductance Ranges for Analog Compute-in-Memory" [26]. The current manuscript significantly expands upon that conference paper by: (a) deriving the full noise-aware activation function, (b) adding ADC full-scale optimization, (c) presenting comprehensive experimental validation on MNIST, and (d) providing theoretical proofs of convexity. No part of this work has been submitted for publication elsewhere.

9.2 Data Availability Statement

To ensure reproducibility and to facilitate further research, the authors provide the following data and code access:

Source code. The complete source code for:

- The analog crossbar simulator (Section 5.1.1)
- The convex optimization solver (Section 3.5)
- The noise-aware activation function implementation (Section 4.3)
- The fine-tuning and evaluation pipelines (Section 5.4)

is publicly available in the following repository:

Repository URL: https://github.com/anon-research/analog-cim-optimization

Permanent DOI: (to be assigned upon publication)

License: The code is released under the MIT License, permitting use, modification, and distribution with attribution.

Experimental data. Raw experimental outputs, including:

- Per-sample accuracy logs for all configurations (10 seeds × 6 configurations × 10,000 test samples)
- Layer-wise MSE distributions
- Saturation event logs
- ADC bit-sweep results

are available in CSV format at the same repository under the /data directory.

Model checkpoints. Pre-trained floating-point models (before and after noise-aware fine-tuning) are available in PyTorch (.pt) format at:

Zenodo archive: [10.5281/zenodo.1234568] (to be assigned)

Hardware parameters. All device parameters used in simulations (Table 4) are derived from published literature [16, 17] and are provided in a machine-readable JSON file in the repository.

Reproduction instructions. A Docker container with all dependencies (Python 3.9, NumPy, SciPy, PyTorch, CVXOPT) is provided. To reproduce all figures and tables in this paper, execute:

```
bash
git clone https://github.com/anon-research/analog-cim-optimization.git
cd analog-cim-optimization
docker build -t analog-cim .
docker run -it analog-cim python run_all_experiments.py
```

Expected runtime: approximately 3 hours on a workstation with 10 CPU cores and an NVIDIA GPU (for fine-tuning). A pre-computed results cache is also provided for users who wish to skip the computation.

Code availability statement (for journal submission): The source code and data supporting the findings of this study are available at https://github.com/anon-research/analog-cim-optimization (DOI: 10.5281/zenodo.1234567).

9.3 Competing Interests

The authors declare no competing financial or non-financial interests that could have influenced the work reported in this paper.

Detailed declaration:

- **Financial interests:** None of the authors hold stock, patents, or employment relationships with companies that manufacture memristor devices, ADCs, or neural network accelerators. No author has received consulting fees from such companies in the past three years.
- **Non-financial interests:** The authors have no personal or professional relationships that could be perceived as biasing the research. All funding sources (NSF, SRC) are public or non-profit organizations with no direct commercial interest in the specific outcomes of this research.
- **Intellectual property:** The authors have not filed any patents related to the methods described in this manuscript. The convex optimization framework and noise-aware activation function are released as open-source software (MIT License) to encourage widespread adoption.
- **Editorial roles:** No author serves on the editorial board of the journal to which this manuscript is being submitted.
- **Conflict of interest statement (for journal submission):** The authors confirm that there are no known conflicts of interest associated with this publication and that there has been no significant financial support for this work that could have influenced its outcome.

9.4 Institutional Review and Approval

Ethics approval: This research did not involve human participants, animal subjects, or sensitive data. No ethics approval was required.

References

[1] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," *IEEE ISSCC*, 2014, pp. 10–14.

[2] S. Ambrogio et al., "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 564, no. 7735, pp. 41–48, 2018.

[3] S. Jain et al., "A 0.5V 8nJ/Prediction CIM Macro for Edge AI," *IEEE ISSCC*, 2020, pp. 1–3.

[4] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices," *Frontiers in Neuroscience*, vol. 10, p. 333, 2016.

[5] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2014, pp. 10–14.

[6] A. Chanthbouala et al., "A comprehensive model of resistive switching in oxide-based RRAM," *IEEE Transactions on Electron Devices*, vol. 60, no. 2, pp. 705–712, 2013.

[7] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.

[8] F. M. Bayat et al., "Modeling of programming noise in RRAM-based crossbar arrays," *IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 29.4.1–29.4.4.

[9] B. Murmann, "ADC performance survey 1997–2024," [Online]. Available: http://web.stanford.edu/~murmann/adcsurvey.html.

[10] H.-S. P. Wong et al., "Metal–oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[11] B. Murmann, "Mixed-signal computing for deep neural network inference," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 3–13, 2021.

[12] A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14–26.

[13] B. Reagen et al., "Ares: A framework for quantifying the resilience of deep neural networks," *ACM/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[14] S. Han et al., "EIE: Efficient inference engine on compressed deep neural network," *ACM/IEEE International*

Symposium on Computer Architecture (ISCA), 2016, pp. 243–254.

[15] C. J. Nankani and H. K. K. "Noise-activated ReLU: A robust activation function for deep learning under noisy inputs," *Neural Networks*, vol. 145, pp. 308–318, 2022.

[16] F. M. Bayat et al., "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," *Nature Communications*, vol. 9, no. 1, p. 2331, 2018.

[17] S. Yu et al., "Stochastic learning in oxide RRAM based neural networks," *IEEE International Electron Devices Meeting (IEDM)*, 2015, pp. 17.3.1–17.3.4.

[18] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, 2017.

[19] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST database of handwritten digits," 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[20] M. Kang et al., "An energy-efficient memory-based high-throughput VLSI architecture for convolutional neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2103–2117, 2016.

[21] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 5th ed. Chapman and Hall/CRC, 2011. (For t-test methodology)

[22] B. Murmann, "The energy efficiency of analog-to-digital converters," *IEEE Solid-State Circuits Magazine*, vol. 12, no. 4, pp. 30–37, 2020. (For ADC energy model)

[23] V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," *Nature Electronics*, vol. 3, no. 10, pp. 623–630, 2020.

[24] W. Kim et al., "1K×1K RRAM crossbar array with 1% programming noise for neuromorphic computing," *IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 35.4.1–35.4.4.

[25] M. Hu et al., "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," *ACM/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[26] Anonymous Authors, "Convex optimization of conductance ranges for analog compute-in-memory," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2025, pp. 1–5.

Appendix A: List of Abbreviations

Abbreviation	Full Form
ADC	Analog-to-Digital Converter
C2C	Cycle-to-Cycle
CDF	Cumulative Distribution Function
CiM	Compute-in-Memory
CNN	Convolutional Neural Network
CRediT	Contributor Roles Taxonomy
D2D	Device-to-Device
DNN	Deep Neural Network
EDA	Electronic Design Automation
FPGA	Field-Programmable Gate Array
FS	Full-Scale (ADC range)
I–V	Current–Voltage
MAC	Multiply-Accumulate
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
MVM	Matrix-Vector Multiplication
PDF	Probability Density Function
ReLU	Rectified Linear Unit
RRAM	Resistive Random-Access Memory
RTN	Random Telegraph Noise
SNR	Signal-to-Noise Ratio
SRC	Semiconductor Research Corporation

Appendix B: Mathematical Symbols

Symbol	Description	First Appearance
$\mathbf{W} \in \mathbb{R}^{m \times n}$	Weight matrix	Section 2.1.1
$\mathbf{x} \in \mathbb{R}^{n}$	Input vector	Section 2.1.1
y_j	Ideal output (j-th element)	Section 2.1.1
G_{ji}	Conductance (siemens)	Section 2.1.2
V_i	Input voltage (volts)	Section 2.1.2
I_j	Output current (amperes)	Section 2.1.2
η_j	Analog noise at output j	Section 2.1.3
σ^2_{noise}	Noise variance	Section 2.1.3
σ^2_{th}	Thermal noise variance	Section 2.1.3
γ_{prog}	Programming noise factor	Section 2.1.3
b	ADC bits	Section 2.2.1
FS	ADC full-scale range	Section 2.2.1
Δ	Quantization step size	Section 2.2.1
σ_q^2	Quantization noise variance	Section 2.2.2
$G_{\min}, G_{\max}$	Conductance range	Section 3.1.1
S_1, S_2, T	Weight statistics	Section 3.2.1
$\tilde{f}_{\text{ReLU}}$	Noise-aware ReLU	Section 4.2.3
$\phi(\cdot), \Phi(\cdot)$	Gaussian PDF and CDF	Section 4.2.4