

# Extending Straight-Through Estimation for Robust Neural Networks on Analog CIM Hardware

Yuannuo Feng<sup>†1,3</sup>, Wenyong Zhou<sup>†2,3</sup>, Yuexi Lyu<sup>‡3</sup>, Yixiang Zhang<sup>3</sup>, Zhengwu Liu<sup>\*2</sup>, Ngai Wong<sup>\*2</sup> and Wang Kang<sup>\*1</sup>

<sup>1</sup>School of Integrated Circuit Science and Engineering, Beihang University, Beijing, China

<sup>2</sup>Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

<sup>3</sup>Zhicun Research Lab, Beijing, China

<sup>†</sup>: Equal Contribution. <sup>‡</sup>: Project Leader. <sup>\*</sup>: Corresponding Author(s).

**Abstract**—Analog Compute-In-Memory (CIM) architectures promise significant energy efficiency gains for neural network inference, but suffer from complex hardware-induced noise that poses major challenges for deployment. While noise-aware training methods have been proposed to address this issue, they typically rely on idealized and differentiable noise models that fail to capture the full complexity of analog CIM hardware variations. Motivated by the Straight-Through Estimator (STE) framework in quantization, we decouple forward noise simulation from backward gradient computation, enabling noise-aware training with more accurate but computationally intractable noise modeling in analog CIM systems. We provide theoretical analysis demonstrating that our approach preserves essential gradient directional information while maintaining computational tractability and optimization stability. Extensive experiments show that our extended STE framework achieves up to 5.3% accuracy improvement on image classification, 0.72 perplexity reduction on text generation, 2.2× speedup in training time, and 37.9% lower peak memory usage compared to standard noise-aware training methods.

**Index Terms**—Analog Compute-In-Memory, Straight-Through Estimator, Hardware-Aware Training

## I. INTRODUCTION

The exponential growth of neural network applications has intensified demand for energy-efficient computing solutions, particularly for edge devices with severe power and computational constraints [1], [2]. Analog Compute-In-Memory (CIM) architectures address these challenges by performing matrix-vector multiplications directly within memory arrays, eliminating energy-intensive data movement and achieving orders of magnitude energy efficiency improvements over traditional von Neumann architectures through analog weight storage and physical law-based computation [3], [4].

However, the analog nature introduces significant deployment challenges, as analog CIM hardware suffers from various noise sources that severely degrade inference accuracy [5], [6]. These include device-level variations from manufacturing and aging, circuit-level noise from voltage drops and current leakage, and peripheral circuit imperfections such as ADC quantization errors and amplifier non-linearities [7], [8]. The cumulative effect of these noise sources can lead to substantial accuracy degradation, limiting practical neural network deployment on analog CIM hardware [9], [10].

To address this challenge, researchers have developed noise-aware training methods that expose neural networks to simulated hardware noise during training, enabling them to develop robustness against deployment conditions [11], [12]. However, existing approaches face a fundamental limitation: they rely on simplified, differentiable noise models that can be easily integrated into standard gradient-based optimization frameworks. While Gaussian noise injection or uniform quantization can be readily handled by automatic differentiation, they fail to capture the full complexity of real analog CIM hardware variations, as many critical hardware effects are inherently non-differentiable, exhibit complex spatial and temporal correlations, or involve computationally expensive physical simulations that make direct gradient computation impractical [7]. This limitation creates a significant gap between the simplified noise models used during training and the complex noise characteristics encountered during deployment, where networks trained with oversimplified noise models often fail to generalize to real hardware conditions, resulting in substantial accuracy drops when deployed on actual analog CIM systems [13].

We extend the Straight-Through Estimator (STE) framework, originally for quantization, to address complex noise in analog CIM hardware [14], [15]. By decoupling forward noise simulation from backward gradient computation, we enable realistic noise models in forward passes and simplified gradients in backpropagation, achieving efficient noise-aware training with high-fidelity hardware models. This paper provides three key contributions:

- We formalize the extension of STE to complex noise environments, providing a general framework that can accommodate various types of hardware noise beyond simple quantization.
- We develop a STE-based gradient approximation strategy and provide theoretical understanding of its effectiveness from the gradient perspective.
- Experiments demonstrate that our extended STE framework achieves up to 5.3% higher accuracy on image classification, 0.72 lower perplexity on text generation, 2.2× faster training, and 37.9% less peak memory usage than standard noise-aware methods.

## II. PRELIMINARIES

The STE was originally developed to enable gradient-based training of neural networks containing non-differentiable functions, particularly quantization operations [16]. In the standard quantization scenario, the forward pass applies a quantization function  $Q(\cdot)$  to the weights or activations:

$$y = Q(x) \quad (1)$$

where  $Q(\cdot)$  is typically a step function that maps continuous values to discrete levels.

The challenge arises during backpropagation, as the gradient of  $Q(\cdot)$  is zero almost everywhere, preventing effective learning. The STE addresses this by using different functions for forward and backward passes. During the forward pass, the actual quantization function is applied, while during the backward pass, the gradient flows through as if the quantization operation were the identity function:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} \cdot \mathbf{1} \quad (2)$$

where  $\mathbf{1}$  represents the identity operation [17].

## III. METHODOLOGY

### A. Problem Formulation

We consider training neural networks that operate robustly under complex deployment noise. Let  $\mathbf{W} \in \mathbb{R}^d$  denote the clean weights and  $\mathcal{N}(\mathbf{W}, \xi)$  represent a complex noise function, where  $\xi$  encapsulates noise parameters. The noisy weights are:

$$\tilde{\mathbf{W}} = \mathcal{N}(\mathbf{W}, \xi) \quad (3)$$

The training objective minimizes the expected loss:

$$\mathbb{E}_\xi[L] = \mathbb{E}_\xi[\ell(f(\tilde{\mathbf{W}}, \mathbf{x}), \mathbf{y})] \quad (4)$$

The challenge arises when computing gradients  $\frac{\partial L}{\partial \mathbf{W}}$  through  $\mathcal{N}(\mathbf{W}, \xi)$ , which may be non-differentiable, have zero gradients almost everywhere, or involve computationally expensive stochastic processes that make standard backpropagation infeasible.

Algorithm 1 demonstrates our STE approach for a linear layer. During the forward pass, inputs and weights are quantized to int8 precision, converted to appropriate formats for the noise model, and processed to generate noisy outputs. The difference between the noisy and clean outputs is computed as a delta term, which is then added to the clean output using gradient detachment to preserve backpropagation through the original linear computation while incorporating the effects of quantization noise.

### B. Theoretical Analysis

To understand how the STE can be extended to complex noise environments, we analyze the approach from the perspective of gradient direction and magnitude. The effectiveness of STE can be understood by decomposing any gradient vector  $\mathbf{g}$  into its components:

$$\mathbf{g} = \|\mathbf{g}\| \cdot \frac{\mathbf{g}}{\|\mathbf{g}\|} \quad (5)$$

---

### Algorithm 1 Noisy Linear Layer with Quantization and Noise

---

```

1: function LINEAR( $x, linear, noise\_level, with\_noise$ )
2:   if not  $with\_noise$  then
3:     return Linear( $x, linear.weight, linear.bias$ )
4:   end if
5:    $y_{clean} = \text{Linear}(x, linear.weight, linear.bias)$ 
6:    $noise\_model = \text{NoiseModelNNTrain}(noise\_level)$ 
7:   with torch.no_grad():
8:      $x_{q8}, scale_x = \text{Quant}_{int8}(x)$ 
9:      $w_{q8}, scale_w = \text{Quant}_{int8}(linear.weight)$ 
10:     $x_{uint8}, w_{split} = \text{SplitInput}(x_{q8}, w_{q8})$ 
11:     $bias_q = \text{QuantBias}(linear.bias, scale_x, scale_w)$ 
12:     $g = \text{ComputeGain}(scale_x, scale_w, y_{clean})$ 
13:     $noise\_model.program(x_{uint8}, w_{split}, bias_q, g)$ 
14:     $y_{noisy} = noise\_model.infer(x_{uint8})$ 
15:     $y_{scaled} = \text{Rescale}(y_{noisy}, g, scale_x, scale_w)$ 
16:     $\delta = y_{scaled} - y_{clean}.detach()$ 
17:   return  $y_{clean} + \delta.detach()$ 
18: end function

```

---

where  $\|\mathbf{g}\|$  represents the gradient norm (step size information) and  $\frac{\mathbf{g}}{\|\mathbf{g}\|}$  represents the gradient direction (update direction).

Consider a neural network layer computation affected by complex hardware noise. The forward computation can be expressed as:

$$\mathbf{y} = \tilde{\mathbf{W}}\mathbf{x} = \mathcal{N}(\mathbf{W}, \xi)\mathbf{x} \quad (6)$$

where  $\mathcal{N}(\cdot, \cdot)$  represents the complex noise function that transforms the clean weights  $\mathbf{W}$  into noisy weights  $\tilde{\mathbf{W}}$ .

The ground truth gradient that properly accounts for the noise characteristics would be:

$$\mathbf{g}^* = \frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathcal{N}(\mathbf{W}, \xi)}{\partial \mathbf{W}} \mathbf{x} \quad (7)$$

However, computing this gradient exactly is often intractable due to the complexity and non-differentiable nature of  $\mathcal{N}(\cdot, \cdot)$ . The STE approach approximates this ground truth gradient using a simplified gradient that assumes  $\frac{\partial \mathcal{N}(\mathbf{W}, \xi)}{\partial \mathbf{W}} \approx \mathbf{I}$ :

$$\tilde{\mathbf{g}} = \frac{\partial L}{\partial \mathbf{y}} \mathbf{x} \quad (8)$$

From a statistical perspective, we can view the relationship between the ground truth gradient and the STE gradient as:

$$\mathbf{g}^* = \tilde{\mathbf{g}} + \boldsymbol{\delta} \quad (9)$$

where  $\boldsymbol{\delta} = \frac{\partial L}{\partial \mathbf{y}} \left( \frac{\partial \mathcal{N}(\mathbf{W}, \xi)}{\partial \mathbf{W}} - \mathbf{I} \right) \mathbf{x}$  represents the bias term introduced by the noise function.

**Direction Analysis:** The STE gradient  $\tilde{\mathbf{g}}$  represents the full-precision gradient direction, which captures the correct optimization trajectory for the clean network. To quantify the directional reliability, consider the cosine similarity between the STE gradient and the ground truth gradient:

$$\cos(\theta) = \frac{\langle \tilde{\mathbf{g}}, \mathbf{g}^* \rangle}{\|\tilde{\mathbf{g}}\| \|\mathbf{g}^*\|} = \frac{\langle \tilde{\mathbf{g}}, \tilde{\mathbf{g}} + \boldsymbol{\delta} \rangle}{\|\tilde{\mathbf{g}}\| \|\tilde{\mathbf{g}} + \boldsymbol{\delta}\|} \quad (10)$$

Expanding this expression:

$$\cos(\theta) = \frac{\|\tilde{g}\|^2 + \langle \tilde{g}, \delta \rangle}{\|\tilde{g}\| \|\tilde{g} + \delta\|} \quad (11)$$

When the noise-induced bias  $\delta$  is uncorrelated with the clean gradient  $\tilde{g}$  (i.e.,  $\mathbb{E}[\langle \tilde{g}, \delta \rangle] \approx 0$ ), the cosine similarity approaches:

$$\mathbb{E}[\cos(\theta)] \approx \frac{\|\tilde{g}\|}{\|\tilde{g} + \delta\|} \approx \frac{\|\tilde{g}\|}{\sqrt{\|\tilde{g}\|^2 + \|\delta\|^2}} \quad (12)$$

This shows that the directional alignment deteriorates as the noise bias magnitude  $\|\delta\|$  increases relative to the clean gradient magnitude  $\|\tilde{g}\|$ . The variance of the ground truth gradient can be expressed as:

$$\text{Var}[g^*] = \text{Var}[\delta] = \mathbb{E}[\|\delta\|^2] - \|\mathbb{E}[\delta]\|^2 \quad (13)$$

Since  $\tilde{g}$  is deterministic given the clean weights, it exhibits zero variance, making it statistically more reliable for consistent optimization direction.

The key insight is that  $\tilde{g}$  provides a projection of the optimization direction onto the clean parameter space. Mathematically, this can be viewed as:

$$\tilde{g} = \mathbb{E}_\xi[g^*] \quad \text{when } \mathbb{E}_\xi[\delta] = 0 \quad (14)$$

When noise is not zero-mean,  $\tilde{g}$  provides a simplified approximation of the optimization direction by ignoring the systematic bias introduced by the noise in  $g^*$ .

**Magnitude Analysis:** While the magnitude of  $\tilde{g}$  may differ from  $g^*$ , this discrepancy is less critical for optimization success. The magnitude relationship can be expressed as:

$$\|g^*\|^2 = \|\tilde{g} + \delta\|^2 = \|\tilde{g}\|^2 + 2\langle \tilde{g}, \delta \rangle + \|\delta\|^2 \quad (15)$$

The relative magnitude difference is:

$$\frac{\|g^*\|^2 - \|\tilde{g}\|^2}{\|\tilde{g}\|^2} = \frac{2\langle \tilde{g}, \delta \rangle + \|\delta\|^2}{\|\tilde{g}\|^2} \quad (16)$$

Modern optimizers such as Adam effectively normalize gradients using adaptive scaling:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (17)$$

where  $\hat{m}_t$  and  $\hat{v}_t$  are bias-corrected momentum estimates. This adaptive mechanism makes the optimization relatively insensitive to the exact gradient magnitude, as the effective step size is determined by the gradient history rather than instantaneous magnitude.

#### IV. EXPERIMENTS

##### A. Experiment Setup

We develop a comprehensive noise simulator modeling realistic analog CIM accelerator conditions with both I/O and tile-level non-idealities detailed in Table I. I/O non-idealities include ADC/DAC quantization noise and device non-linearity effects. Tile-level non-idealities encompass programming noise from weight fabrication variations, cycle-by-cycle read variance, thermal-induced parameter drift,

TABLE I  
HARDWARE NON-IDEALITIES MODELED IN CIM NOISE SIMULATOR.

Category	Noise Source	Type
I/O non-idealities	ADC noise	Quantization noise
	DAC noise	Quantization noise
	Device non-linearity	System non-linearity
Tile non-idealities	Programming noise	Weight fabrication non-ideality
	Cycle-by-cycle read variance	Computational consistency
	Thermal variation	Temperature-dependent drift
	Retention	Memory degradation
	IR-drop	Wire resistance non-ideality

TABLE II  
PERFORMANCE COMPARISON OF MODELS ACROSS DIFFERENT LOGNORMAL NOISE LEVELS.

Dataset	Model	Performance under Noise					
		level = 1.0		level = 2.0		level = 3.0	
		Value	$\Delta$	Value	$\Delta$	Value	$\Delta$
CIFAR-10	VGG-8	90.14%	+1.43%	89.56%	+1.99%	89.38%	+2.75%
	VGG-11	89.90%	+3.20%	89.56%	+4.50%	89.90%	+5.30%
	ResNet-20	90.77%	+2.57%	90.66%	+2.65%	90.44%	+3.02%
	ResNet-18	94.00%	+3.58%	93.59%	+1.98%	93.78%	+2.56%
Shakespeare	BERT	3.10	-0.31	3.11	-0.33	3.16	-0.24
	GPT	4.65	-0.51	4.69	-0.50	4.69	-0.72

TABLE III  
TRAINING PHASE HARDWARE COST COMPARISON OF DIFFERENT GRADIENT MANAGEMENT METHODS.

Method	Time (s)	Memory (MB)		GPU Utilization (%)	
		Peak	Average	Compute	Memory
Baseline	0.136	1501	342	64.9	4.14
Full Gradient	1.172	4019	343	96.9	7.94
<code>torch.no_grad</code>	0.608	2496	342	94.5	8.44
<code>.detach()</code>	0.616	3124	342	94.3	8.40

retention-based memory degradation, and IR-drop effects from wire resistance causing voltage variations across crossbar arrays.

##### B. Experiment Result

Table II presents the performance comparison of various models across different lognormal noise levels, demonstrating that our method significantly improves performance under different noise conditions compared to baseline models. For CIFAR-10 classification tasks, all models show substantial accuracy increases, with improvements ranging from 1.43% to 5.3%. VGG-11 exhibits the most remarkable gains, achieving up to 5.3% accuracy increase at noise level 3.0. For the Shakespeare language modeling task, both BERT and GPT models experience slight perplexity decreases, with reductions ranging from 0.24 to 0.72.

Table III presents a comprehensive comparison of training phase hardware costs across different gradient management

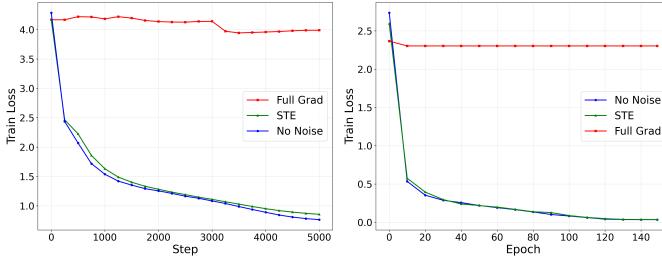


Fig. 1. The training loss reduction curves for GPT-2 on the Shakespeare dataset and ResNet on the CIFAR-10 dataset.

methods. The baseline method, which represents standard training without complex noise considerations, demonstrates the most efficient resource utilization with the lowest time consumption (0.136s), minimal memory usage (1501MB peak, 342MB average). In contrast, the full gradient approach exhibits significantly higher computational overhead, requiring 1.172s per iteration and consuming substantially more memory (4019MB peak, 343MB average). The gradient management techniques `torch.no_grad` and `.detach()` show similar performance characteristics, with execution times of 0.608s and 0.616s respectively, and comparable memory consumption patterns (2496MB and 3124MB peak usage, both with 342MB average). Our proposed methods effectively save memory and accelerate training speed compared to the full gradient approach, with `torch.no_grad` emerging as the superior solution due to its optimal balance between computational efficiency and resource utilization.

The left panel of Fig. 1 shows the training loss curves for GPT-2 models trained on the Shakespeare dataset using three different methods: baseline (No Noise), noise with gradients (Full Grad), and STE. Both the baseline and our STE method exhibit rapid convergence during the initial 1000 training steps, dropping from approximately 4.0 to below 2.0 and reaching final losses around 0.8-1.0. In contrast, the noise gradient method shows minimal improvement throughout training, maintaining consistently high loss around 4.0 for both training and validation sets, suggesting that directly propagating gradients through noisy operations significantly impairs the optimization process. Similarly, the right panel shows ResNet’s performance on CIFAR-10 for image classification, highlighting a significant advantage of our STE method over full gradient training.

## V. CONCLUSION

This work addresses the gap between realistic analog CIM noise modeling and practical training by extending the STE framework to handle complex, non-differentiable hardware variations. By decoupling forward noise simulation from backward gradient computation, our approach enables high-fidelity noise modeling while maintaining computational tractability. Our work improves image classification accuracy by 5.3%, cuts text generation perplexity by 0.72, speeds training 2.2 $\times$ , and reduces peak memory use by 37.9% versus standard noise-aware methods.

## ACKNOWLEDGEMENT

This research was partially conducted by ACCESS – AI Chip Center for Emerging Smart Systems, supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government, and partially supported by the Theme-based Research Scheme (TRS) project T45-701/22-R, the General Research Fund (GRF) Project 17203224 of the Research Grants Council (RGC), Hong Kong SAR, and the National Natural Science Foundation of China Project 62404187.

## REFERENCES

- [1] Z. Guan *et al.*, “A hardware-aware neural architecture search pareto front exploration for in-memory computing,” in *2022 IEEE 16th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*, pp. 1–4, 2022.
- [2] W. Zhou *et al.*, “A time- and energy-efficient cnn with dense connections on memristor-based chips,” in *2023 IEEE 15th International Conference on ASIC (ASIC)*, pp. 1–4, 2023.
- [3] A. Shafiee *et al.*, “ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars,” in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 14–26, 2016.
- [4] P. Chi *et al.*, “PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory,” in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 27–39, 2016.
- [5] W. Mao *et al.*, “Hyimec: Analog-digital hybrid in-memory computing soc for high-quality low-latency speech enhancement,” in *2025 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–2, IEEE, 2025.
- [6] B. Pan *et al.*, “A mini tutorial of processing in memory: From principles, devices to prototypes,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 7, pp. 3044–3050, 2022.
- [7] G. Charan *et al.*, “Accurate Inference with Inaccurate RRAM Devices: Statistical Data, Model Transfer, and On-line Adaptation,” in *ACM/IEEE Design Automation Conference (DAC)*, 2020.
- [8] N. a. Ye, “BayesFT: Bayesian Optimization for Fault Tolerant Neural Network Architecture,” in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 487–492, 2021.
- [9] Y. Chen *et al.*, “Device characteristic-aware quantization for eflash-based in-memory computing soc,” in *2024 IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA)*, pp. 70–71, IEEE, 2024.
- [10] T. Bai *et al.*, “An end-to-end in-memory computing system based on a 40-nm eflash-based imc soc: Circuits, toolchains, and systems co-design framework,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 6, pp. 1729–1740, 2024.
- [11] W. Zhou *et al.*, “Towards Robust RRAM-based Vision Transformer Models with Noise-aware Knowledge Distillation,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–2, 2025.
- [12] W. Zhou *et al.*, “Enhancing robustness of implicit neural representations against weight perturbations,” in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025.
- [13] G. Jung *et al.*, “Cost- and Dataset-free Stuck-at Fault Mitigation for ReRAM-based Deep Learning Accelerators,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.
- [14] Z. Liu *et al.*, “Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4932–4942, 2022.
- [15] Z. Liu *et al.*, “Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm,” (Berlin, Heidelberg), p. 747–763, Springer-Verlag, 2018.
- [16] R. Gong *et al.*, “Differentiable soft quantization: Bridging full-precision and low-bit neural networks,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4851–4860, 2019.
- [17] H. Qin *et al.*, “Forward and backward information retention for accurate binary neural networks,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2247–2256, 2020.