Sustainable Deployment of Deep Neural Networks on Non-Volatile Compute-in-Memory Accelerators

Yifan Qin[†⊛] **Zheyu Yan**[†] **Wujie Wen**[‡] **Xiaobo Sharon Hu**[†] **Yiyu Shi**[†*]
[†]University of Notre Dame, [‡]North Carolina State University {[⊛]yqin3, [*]yshi4}@nd.edu

***Abstract*—Non-volatile memory (NVM) based compute-in-memory (CIM) accelerators have emerged as a sustainable solution to significantly boost energy efficiency and minimize latency for Deep Neural Networks (DNNs) inference due to their in-situ data processing capabilities. However, the performance of NVCIM accelerators degrades because of the stochastic nature and intrinsic variations of NVM devices. Conventional write-verify operations, which enhance inference accuracy through iterative writing and verification during deployment, are costly in terms of energy and time. Inspired by negative feedback theory, we present a novel negative optimization training mechanism to achieve robust DNN deployment for NVCIM. We develop an Oriented Variational Forward (OVF) training method to implement this mechanism. Experiments show that OVF outperforms existing state-of-the-art techniques with up to a 46.71% improvement in inference accuracy while reducing epistemic uncertainty. This mechanism reduces the reliance on write-verify operations and thus contributes to the sustainable and practical deployment of NVCIM accelerators, addressing performance degradation while maintaining the benefits of sustainable computing with NVCIM accelerators.**

***Index Terms*—in-memory computing, sustainable, neural network, accelerators**

I. Introduction

Deep Neural Networks (DNNs) have revolutionized our society, but their acceleration is hindered by the constant need for data movement between memory and processing units, known as the von Neumann bottleneck [1]. Non-volatile memory (NVM) based Computing-In-Memory (CIM) DNN accelerators [2] offer a potential solution by enabling parallel in-situ data processing, surpassing CMOS-based counterparts in energy efficiency and density [1], [3]. These accelerators utilize emerging NVM devices such as ferroelectric field-effect transistors (FeFETs) [4], resistive random-access memories (RRAMs) [5], magnetoresistive random-access memories (MRAMs) [6], and phase-change memories (PCMs) [7], providing a sustainable solution for DNN inference acceleration. Despite their advantages, NVM devices in NVCIM DNN accelerators suffer from inherent non-idealities, such as device variations [5], [8]. These variations cause perturbations in device conductance after programming [9], often resulting in Gaussian-distributed conductance values [5]. Consequently, the model weights are affected, ultimately impacting the inference accuracy of NVCIM DNN accelerators [5], [8], [10].

Ensuring reliable DNN inference on unreliable NVM substrates presents a significant challenge. Among solutions [11]–[13], hardware write-verify has emerged as a widely adopted method for accelerator deployment. However, the time-consuming and energy-intensive iterative write and verify operations hinder sustainable deployment. Therefore, we need more robust network models to achieve more sustainable deployment and acceleration.

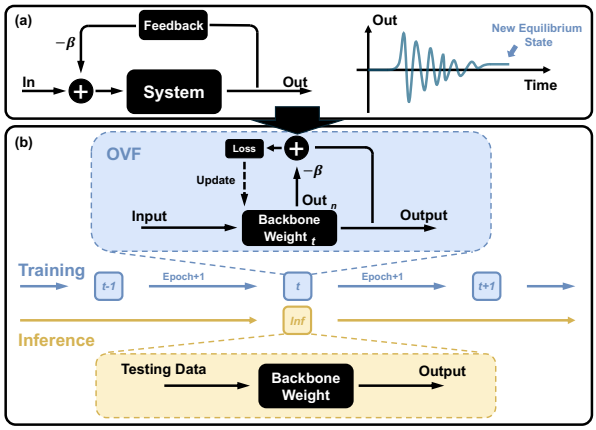

Fig. 1. (a) Schematic diagram of a classic negative feedback system and the process of the system setting to a new equilibrium state. (b) Illustration of negative optimization training mechanism.

Noise-injection training [8] is widely used to enhance model robustness by exposing DNNs to Gaussian noise during training. This improves the model's noise tolerance. However, state-of-the-art (SOTA) noise-injection methods have limitations, such as limited accuracy improvement, increased epistemic uncertainty [14], and challenges in achieving convergence. We attribute this to the mismatch between the non-deterministic nature of noise and the deterministic nature of training. Specifically, **first**, the network is exposed to only a finite number of noise samples during training, limiting its ability to fully understand noise patterns. **Second**, random noise samples provide diverse optimization directions, but some may lead to incorrect states, increasing uncertainty and hindering convergence.

We believe this mismatch can be mitigated by acquiring sufficient variation information during training, rather than relying solely on the final output, as is common in SOTA methods. This hypothesis is rooted in modern control theory, where stability relies on negative feedback. When a system is subjected to noise, noisy outputs help the system resist perturbation through negative feedback, achieving a new equilibrium state. Figure 1(a) illustrates this process. Feedback is generated from a portion of the outputs and modulated by the negative feedback coefficient β. Inspired by this, we introduce a novel **negative optimization training mechanism**, which incorporates negative contributions from outputs, reduces the impact of noise, and helps the neural network achieve a more robust state, as shown in Figure 1(b). At a high level, the entire

2832-6474/24/$31.00 ©2024 IEEE
DOI 10.1109/CODES-ISSS60120.2024.00017

negative optimization training mechanism can be summarized as follows, we use negative optimization training to enhance the robustness of the DNN backbone. After training, all negative contribution components are removed, leaving a robust and unaltered DNN backbone model.

Here we provide an implementation called the **Oriented Variational Forward (OVF)** training method to ground this mechanism. During training, variational inference outputs Out_n are generated with the same backbone weights and oriented increasing amplitude noise, and contribute negatively to the objective in back-propagation with coefficient β, thereby enhancing DNN robustness. The "negative" aspect of OVF reduces the bad effects of target variation while preserving its subject status, and the "feedback" component introduces supplementary noise information that differs from what the backbone output contains, contributing to the objective. OVF constrains network optimization based on the influence of noise itself, with stronger constraints corresponding to greater perturbations. This ensures the network does not deviate far from the optimal direction, facilitating stable convergence to the optimal state during the training process.

Our contributions can be summarized as follows:

- We introduce a negative optimization training mechanism into the DNN training process to enhance stability and improve robustness to device variations. *To the best of our knowledge, this is the first approach of its kind.*
- We propose a novel implementation of this mechanism: Oriented Variational Forward (OVF), which optimizes the network from a comprehensive variational performance perspective.
- Our simulations on NVCIM DNN accelerators with varying device variations demonstrate the effectiveness of OVF in mitigating sensitivity and output fluctuations. OVF boosts confidence and convergence probability while reducing epistemic uncertainty. For example, it achieves up to a 46.71% improvement in DNN average inference performance compared to state-of-the-art methods.

II. Proposed Methodology

In this section, we introduce the negative optimization with the OVF training method.

Our proposed method is inspired by negative feedback theory, a cornerstone of system control. We treat weight variations as perturbations and aim to improve system robustness by suppressing this "noise" through negative optimization. The primary challenge is constructing an effective negative constraint. Simply employing a negatively scaled output of the DNN, as in standard negative feedback systems, is insufficient because it only scales the loss function without altering the training method.

Instead, we require negative constraints that can track changes in the output while remaining distinct from it. The constraint must satisfy two criteria: **First**, it should be generated by components influenced by the same noise pattern present in the backbone. **Second**, the negative constraint should strongly connect with the backbone weights, accurately reflecting weight perturbations.

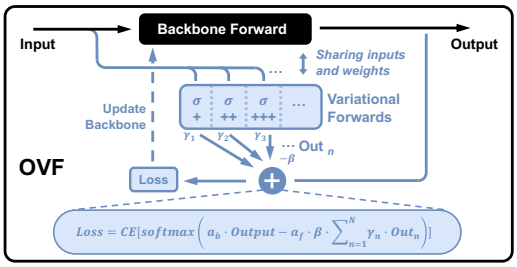

Fig. 2. Negative optimization implementations: oriented variational forward training.

As shown in Figure 2, OVF generates constraints using less representative outputs Out_n from oriented variational forwards, which involve device variations larger than those in backbone inference. By employing negative constraints, OVF prevents the backbone from deviating from the optimal optimization direction. Specifically, during each training iteration, we sample a variation instance $\Delta \mathbf{w}_i$ from a Gaussian distribution $\mathcal{D}ist = \mathcal{N}(0, \sigma^2)$, which is the same as the inference device variations in accelerators. This variation is added to the backbone weights in the feed-forward process, generating the backbone output $\mathcal{O}_{\text{backbone}}$. Unlike typical training, which directly performs back-propagation and weight update, OVF performs multiple oriented variational forwards using the same variation-free backbone weights with noises sampled from $\mathcal{N}(0, \sigma^2)$ but with larger σ, collecting N constraint outputs Out_n. The total output $\mathcal{O}_{\text{total}}$ is given by

$$\mathcal{O}_{\text{total}} = a_b \cdot \mathcal{O}_{\text{backbone}} - a_f \cdot \beta \cdot \sum_{n=1}^{N} \gamma_n \cdot Out_n \qquad (1)$$

where β is the negative constraint coefficient, γ_n are the decay factors, and a_b and a_f are factors influencing the contribution of the two-part outputs. The back-propagation process commences only after all outputs have been obtained.

When selecting hyperparameters, it is important to note that as the standard deviation σ of the Gaussian distribution increases, noise instances introduce a higher degree of entropy and uncertainty to the model and its outputs. This leads to more significant deviations from the target. Consequently, Out_n generated with a larger σ is assigned a larger decay factor γ_n in Eq. 1, thereby imposing a stronger constraint on the backbone model. To achieve this, we set γ_n as 10^{n-N}. These variational forwards generate constraints from the same variation-free backbone weight and employ the same Gaussian noise pattern as the backbone variational forward with different parameters, thus satisfying the criteria outlined previously.

III. Experiments

In this section, we introduce the weight variation model and the setup of experiments, then show the effectiveness of the OVF method with experimental results.

A. Model and Setup

Without loss of generality, we primarily consider device variations stemming from the programming process, where the programmed conductance value in NVM devices deviates from the desired value. Set a DNN weight with M bits, the desired weight value $\bar{\mathcal{W}}_d$ after quantization can be expressed as

$$\bar{\mathcal{W}}_d = \frac{\max|\mathcal{W}|}{2^M - 1} \sum_{i=0}^{M-1} m_i \times 2^i \tag{2}$$

where $\mathcal{W}$ represents floating-point weights, $\max|\mathcal{W}|$ denotes the maximum absolute value among weights, and $m_i \in \{0, 1\}$ signifies the value of the i^{th} bit of the desired weight value. For a NVM device representing K bits of data, each weight can be stored in M/K devices[1], and the mapping process is given by $\bar{g}_j = \sum_{i=0}^{K-1} m_{j\times K+i} \times 2^i$, where $\bar{g}_j$ is the desired conductance of the j^{th} device. It is worth noting that negative weights can be mapped in the same manner to a separate crossbar array. Taking device variation into account, the actual device conductance after programming is denoted as $g_j = \bar{g}_j + \Delta g$, where Δg represents the deviation from the desired conductance value $\bar{g}_j$ and follows a Gaussian distribution. Consequently, the actual weight $\mathcal{W}_p$ represented by programmed NVM devices is given by

$$\mathcal{W}_p = \bar{\mathcal{W}}_d + \frac{\max|\mathcal{W}|}{2^M - 1} \sum_{j=0}^{M/K-1} \Delta g \times 2^{j\times K} \tag{3}$$

In our study, we set $K = 2$, while the value of M was determined by the specific model configuration. For this research, we selected $M = 8$, indicating 8-bit precision for a single DNN weight and 2-bit precision for a single device conductance. To model device variation, we employed a Gaussian distribution with $\Delta g \sim \mathcal{N}(0, \sigma_d^2)$, where σ_d represents the relative standard deviation of conductance corresponding to the maximal conductance of a single device. We set a constraint on σ_d, limiting it to $\sigma_d \leq 0.4$. This range is considered reasonable in prior research [4]–[7] and can be achieved through optimizations at the device level, including write-verify techniques.

We carried out experiments using the PyTorch environment on NVIDIA GPUs. Unless otherwise specified, the reported results represent the average of at least five independent runs. We used the average accuracy of noise-injection inference as the performance metric and performed a Monte Carlo simulation with 200 runs to ensure high precision. Our experiments show that the results have a 95% confidence interval of ± 0.01, in line with the central limit theorem. We compared OVF with two baselines: 1) vanilla training (W/O Noise) and 2) Gaussian noise-injection training (W/ Noise). We did not evaluate OVF against other orthogonal methods, such as NAS-based DNN topology design or Bayesian Neural Networks, as it can be used in combination with them.

Through our experiments with comprehensive datasets and neural network backbones, we have found that the appropriate value of the negative constraint coefficient β consistently falls within the set $\{1e-1, 1e-2, 1e-3, 1e-4\}$. Consequently, a four-step search suffices to determine the setting. For hyperparameter values, we set $start = 0$ and $end = 2 \times \sigma_d$, evaluating OVF efficacy across various σ_d values. We also set the contribution factors $a_b = a_f = 1/(N+1)$, where N represents the number of variational forwards. Other training hyperparameters, such as learning rate, batch size, and learning rate schedulers, follow best practices for training a noise-free model.

[1]For simplicity, we assume that M is a multiple of K.

B. Accuracy Improvement

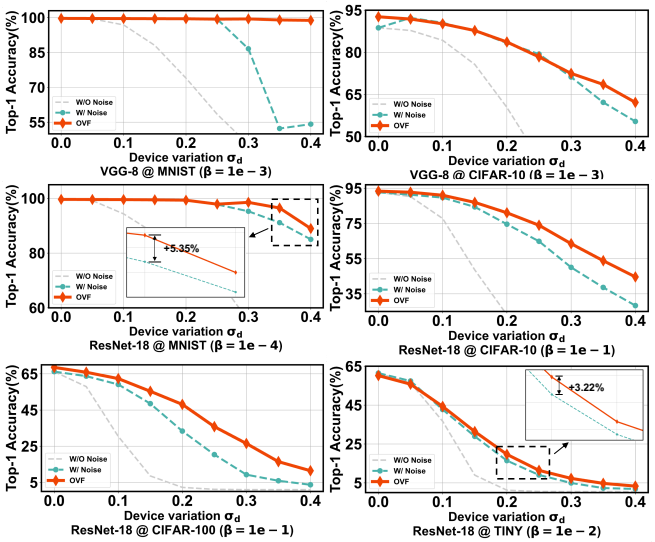

Fig. 3. Effectiveness of OVF: Average noisy inference accuracy on VGG-8 and ResNet-18 backbone models for different datasets across σ_d values.

For our experiments, we employed the VGG-8 backbone and the ResNet-18 backbone on the MNIST, CIFAR-10, CIFAR-100, and Tiny ImageNet datasets. In OVF, we empirically set the variational forwards $N = 3$ for both VGG-8 and ResNet-18, with each increase $\Delta\sigma_d$ fixed at 0.05. Furthermore, the negative constraint coefficient β for each model on a specific dataset was determined through a four-step search process, as stated in section III-A.

Figure 3 illustrates the Top-1 inference accuracy of models trained with different methods under varying levels of device variations σ_d, following the noise model discussed in Section III-A. OVF clearly surpasses all baselines in most device value deviation values and performs similarly to baselines in rare cases where the device variation is too small to have a significant impact. Compared to the Gaussian noise-injection training baseline, OVF enhances the Top-1 accuracy by up to 46.71%, 6.78%, 5.35%, 16.30%, 17.21%, and 3.22% in VGG-8 for MNIST and CIFAR-10, ResNet-18 for MNIST, CIFAR-10, CIFAR-100, and Tiny ImageNet, respectively. OVF constrains the network based on the overall forward performance and is particularly suitable for networks that have not reached the limit of their representational ability, such as VGG-8 on MNIST.

The effectiveness of the OVF method highlights the generality and practicality of the negative optimization training mechanism in enhancing DNN robustness against device variation, thereby contributing to the sustainable deployment of NVCIM accelerators.

C. Uncertainty and Convergence

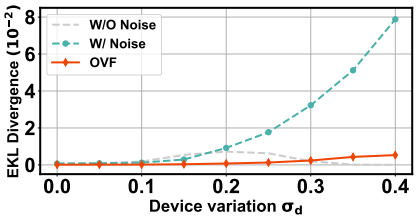

Fig. 4. Average EKL divergence for correct predictions with different methods.

Device variation amplifies epistemic uncertainty, resulting in increased output uncertainty. To quantify the impact of device variation on uncertainty, we employ the Expected Kullback–Leibler (EKL) divergence [14] (lower values indicate better performance). Accuracy is excluded from the analysis for fair comparison purposes. Specifically, among all correct predictions in noisy inference, we calculate the Kullback–Leibler divergence between each softmax output and its corresponding label. The results, shown in Figure 4, represent the average EKL divergence for each correct prediction. Compared to the W/O Noise baseline, the W/ Noise baseline improves accuracy but at the cost of increased uncertainty. In contrast, our OVF method not only achieves even higher accuracy than the noise-injection training baseline but also maintains low uncertainty and high confidence in the output. In cases where device variation is too substantial for effective predictions using the vanilla training baseline, its EKL divergence appears slightly lower than that of OVF, as it generates completely random and meaningless predictions.

For certain devices and aging-related issues, the device variation can be significant. The low uncertainty achieved by OVF also contributes to model convergence. For example, in 10 separate runs of experiments with VGG-8 on MNIST using $\sigma_d = 0.35$, the number of non-converging models[2] is 6 and 0 for noise-injection training and OVF, respectively. In addition to the increase in accuracy, this may also partially explain the substantial improvement of OVF in VGG-8 for MNIST, as shown in Figure 3.

IV. Conclusion

In conclusion, the proposed Oriented Variational Forward (OVF) method significantly enhances the robustness of deep neural networks (DNNs) against device variations and contributes to the sustainable deployment of NVCIM accelerators. By maintaining high accuracy while keeping uncertainty low, OVF reduces the reliance on write-verify operations, improving deployment time and energy efficiency, and achieving better inference accuracy on chips with the same devices. This method demonstrates the generality and practicality of the negative optimization training mechanism, providing valuable insights into the development of robust AI accelerators that can adapt to the non-ideal characteristics of NVM devices, thereby promoting the sustainable development of AI hardware.

[2]where accuracy decreases by more than 5% compared to the average accuracy across multiple independent runs

Acknowledgment

This research was supported by ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR.

References

[1] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.

[2] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[3] W. Zhang, P. Yao, B. Gao, Q. Liu, D. Wu, Q. Zhang, Y. Li, Q. Qin, J. Li, Z. Zhu *et al.*, "Edge learning using a fully integrated neuro-inspired memristor chip," *Science*, vol. 381, no. 6663, pp. 1205–1211, 2023.

[4] D. Reis, M. Niemier, and X. S. Hu, "Computing in memory with fefets," in *Proceedings of the international symposium on low power electronics and design*, 2018, pp. 1–6.

[5] Y. Qin, R. Kuang, X. Huang, Y. Li, J. Chen, and X. Miao, "Design of high robustness bnn inference accelerator based on binary memristors," *IEEE Transactions on Electron Devices*, vol. 67, no. 8, pp. 3435–3441, 2020.

[6] S. Angizi, Z. He, A. Awad, and D. Fan, "Mrima: An mram-based in-memory accelerator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1123–1136, 2019.

[7] X. Sun, W. Khwa, Y. Chen, C. Lee, H. Lee, S. Yu, R. Naous, J. Wu, T. Chen, X. Bao *et al.*, "Pcm-based analog compute-in-memory: Impact of device non-idealities on inference accuracy," *IEEE Transactions on Electron Devices*, vol. 68, no. 11, pp. 5585–5591, 2021.

[8] Z. Yan, Y. Qin, W. Wen, X. S. Hu, and Y. Shi, "Improving realistic worst-case performance of nvcim dnn accelerators through training with right-censored gaussian noise," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[9] M. Rizzi, A. Spessot, P. Fantini, and D. Ielmini, "Role of mechanical stress in the resistance drift of ge2sb2te5 films and phase change memories," *Applied Physics Letters*, vol. 99, no. 22, 2011.

[10] Z. Yan, D.-C. Juan, X. S. Hu, and Y. Shi, "Uncertainty modeling of emerging device based computing-in-memory neural accelerators with application to neural architecture search," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, 2021, pp. 859–864.

[11] R. Degraeve, A. Fantini, N. Raghavan, L. Goux, S. Clima, B. Govoreanu, A. Belmonte, D. Linten, and M. Jurczak, "Causes and consequences of the stochastic aspect of filamentary rram," *Microelectronic Engineering*, vol. 147, pp. 171–175, 2015.

[12] W. Shim, J.-s. Seo, and S. Yu, "Two-step write–verify scheme and impact of the read noise in multilevel rram-based inference engine," *Semiconductor Science and Technology*, vol. 35, no. 11, p. 115026, 2020.

[13] A. Eldebiky, G. L. Zhang, G. Böcherer, B. Li, and U. Schlichtmann, "Correctnet: Robustness enhancement of analog in-memory computing for neural networks by error suppression and compensation," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.

[14] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1513–1589, 2023.