



Article

A Design Methodology for Fault-Tolerant Neuromorphic Computing Using Bayesian Neural Network

Di Gao ¹, Xiaoru Xie ² and Dongxu Wei ^{3,*}

¹ The School of Intelligent Manufacturing, Hangzhou Polytechnic, Hangzhou 311402, China; gaodi1995@outlook.com

² The School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China

³ The College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China

* Correspondence: dspracy@gmail.com

Abstract: Memristor crossbar arrays are a promising platform for neuromorphic computing. In practical scenarios, the synapse weights represented by the memristors for the underlying system are subject to process variations, in which the programmed weight when read out for inference is no longer deterministic but a stochastic distribution. It is therefore highly desired to learn the weight distribution accounting for process variations, to ensure the same inference performance in memristor crossbar arrays as the design value. In this paper, we introduce a design methodology for fault-tolerant neuromorphic computing using a Bayesian neural network, which combines the variational Bayesian inference technique with a fault-aware variational posterior distribution. The proposed framework based on Bayesian inference incorporates the impacts of memristor deviations into algorithmic training, where the weight distributions of neural networks are optimized to accommodate uncertainties and minimize inference degradation. The experimental results confirm the capability of the proposed methodology to tolerate both process variations and noise, while achieving more robust computing in memristor crossbar arrays.



Citation: Gao, D.; Xie, X.; Wei, D. A Design Methodology for Fault-Tolerant Neuromorphic Computing Using Bayesian Neural Network. *Micromachines* **2023**, *14*, 1840. <https://doi.org/10.3390/mi14101840>

Academic Editors: Shubhro Chakrabartty, Alaaddin Al-Shidaifat and Sachin Kumar

Received: 31 August 2023

Revised: 22 September 2023

Accepted: 25 September 2023

Published: 27 September 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: neuromorphic computing; memristor crossbar array; process variation; Bayesian neural network; variational inference

1. Introduction

Neuromorphic computing represents an exciting and promising approach to artificial intelligence (AI) that models its elements on the human brain and nervous system. Instead of relying on a central processing unit (CPU) like traditional von Neumann architectures, neuromorphic systems distribute computations across interconnected nodes, inspired by the remarkable capabilities of the human brain. This parallelism enables efficient and real-time processing of data, making it well suited to tasks such as image and speech recognition [1,2]. Recent research in the field of non-volatile memory technologies has enabled memory devices to tune their conductance and store multi-level states [3,4] such as resistive RAM (ReRAM), etc. On this basis, the utilization of memristor crossbar arrays to speed up matrix–vector multiplication (MVM) operations holds great promise for building energy-efficient and high-performance neural network architectures.

However, the operations of a memristor crossbar array are indeed susceptible to various non-ideal effects due to its analog computing nature. These non-ideal effects can arise from several aspects, such as driver/sensing resistance, analog-to-digital converter (ADC) and digital-to-analog converter (DACs) non-linearity, process variations, and other physical phenomena. The above-mentioned faults can be further categorized to the device or circuit level [5,6].

This paper mainly focuses on device-level fault sources, including process variations [7,8], analog fluctuations, and noise during read and write operations [3,9,10]. The stochastic behavior exhibited by memristor devices can introduce deviations in its conductance, which

then inevitably affect the reliability of the computations performed on the whole memristor crossbar array. Researchers are actively exploring multiple approaches to enhance the overall performance of memristor-based analog computing systems, such as redundancy and error correction [11], adaptive algorithms [12], statistical analysis and modeling [13], and even their combination.

Process variations are inherent in nano-technology manufacturing processes, and can be decomposed into global and local variations. Global components are shared by all the memristors on a die. Local variation is related to the proximity effect and results in variation correlation of the devices across the array. Their combined impact makes neural network parameters heavily deviate from their expected values, thus making the hardware inference performance also significantly deviate from the software validation accuracy. As shown in Figure 1, when mapping a neural network model onto one or more memristor crossbar arrays, the weights of the model are typically represented by the conductance values of the memristors. However, the programmed weights stored in the memristors may deviate from the expected values, i.e., the actual synapse weight may follow a distribution instead of a point value; thus, the inference accuracy cannot reach the same level as in training.

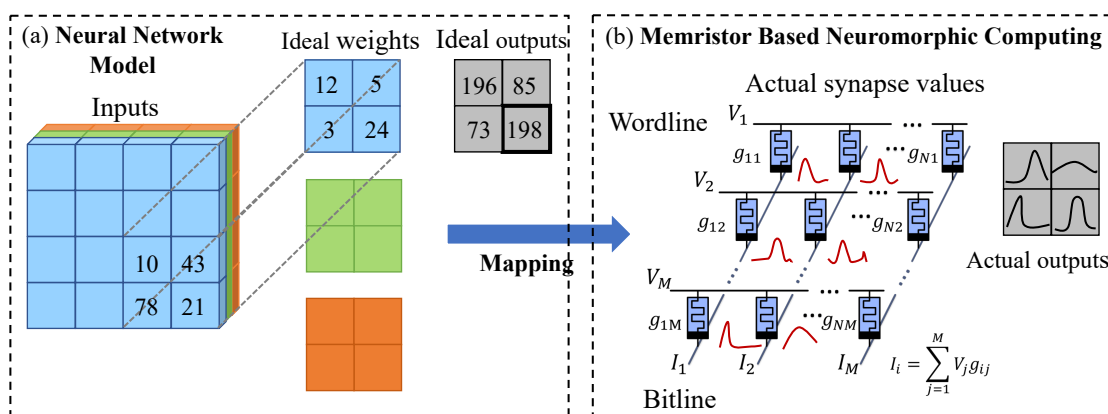


Figure 1. Performance deviation between (a) a well-trained neural network model and (b) its deployment on a memristor-based neuromorphic computing system.

From a system designer's perspective, an as good as possible inference performance is always desired to maximize the benefit of the underlying system. Note that improving the fault tolerance of memristor crossbar arrays is an ongoing research area. Advancements in device fabrication, circuit design, and algorithmic techniques are necessary to develop effective solutions that mitigate the impact of conductance variations and enhance the fault tolerance of memristor-based neuromorphic computing systems.

In this work, we propose a novel design methodology for fault-tolerant neuromorphic computing to address the weight disturbance problem without involving either redundancy design or adaptive retraining/remapping. In particular, by leveraging the probabilistic nature of Bayesian inference, we employ a Bayesian neural network (BNN) architecture to adjust the synapse weights of memristors, accounting for process variations. Specifically, we first model the synapse weights as probability distributions considering the effects of both process variations and noise, i.e., the programmed memristor conductances follow a log-normal distribution [12]. On this basis, we introduce a priori from the same distribution family to estimate the optimal posterior distributions over network parameters that can accommodate uncertain and stochastic behavior in memristor conductance.

2. Background and Related Work

A memristor is a two-terminal electronic device for which the resistance changes with the voltage applied across it, allowing for non-volatile storage and information representation [14–17]. The memristor mentioned in this paper uses resistive random-

access memory (ReRAM), but the proposed methodology is also compatible with other memristor technologies.

Figure 2 shows an overview of a memristor-based neuromorphic computing system as well as the matrix–vector multiplication (MVM) operation conducted on memristor crossbars. In particular, a memristor crossbar array of $M \times N$ is programmed to represent the neural network weights by tuning the conductances [1,18]. When applying an input voltage vector $V = [V_1, V_2, \dots, V_M]$ to the wordlines, the array can output a current vector $I = [I_1, I_2, \dots, I_N]$ according to Kirchhoff's laws. Benefiting from the structure of crossbar arrays, multiple dot products can be computed simultaneously in a single operation [19]. This parallelism enables efficient and high-throughput MVM operations. Additionally, the non-volatile nature of ReRAM allows for the retention of weight values even when the power is turned off, further eliminating the need for frequent weight reloading while reducing the overhead associated with data transfer.

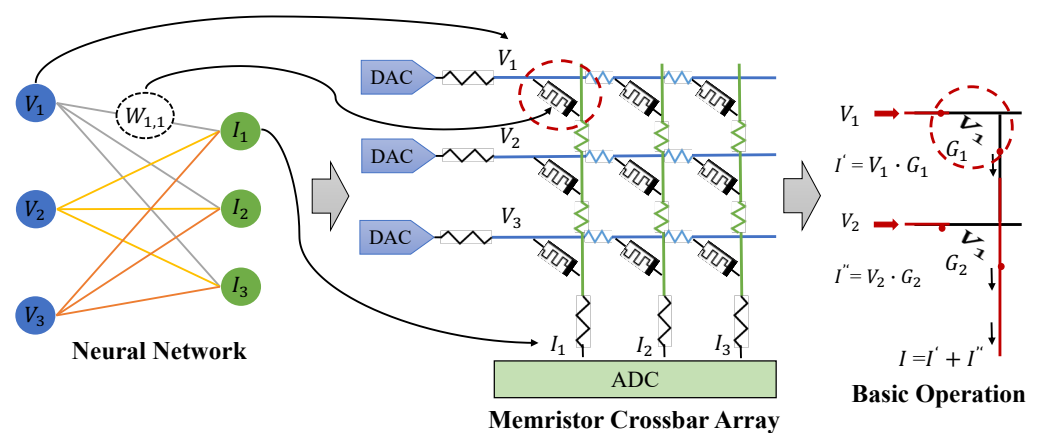


Figure 2. The matrix–vector multiplication (MVM) operation conducted on memristor crossbar arrays.

According to prior work [5,6], the error sources in memristor crossbar arrays include the following: (i) non-ideal peripheral circuitry (DACs and ADCs); (ii) circuit non-idealities (wire resistance, sneak paths, source and sink resistances); and (iii) device deviations due to process variations and thermal, drift, and limited endurance. Considering the analog nature of memristor crossbar arrays, the computational correctness of the underlying MVM operations is easily compromised due to deviations or errors. Note that the inherent presence of process variations and noises makes the weights represented by the synaptic elements within the crossbar array always stochastic. Thus, in the following, we will focus on the effects of process variations and investigate the corresponding fault-tolerant technologies.

Generally speaking, process variations are inherent in the nano-technology manufacturing process [7,8]. Existing work on memristor variation characterization found that it follows a log-normal distribution after the collective motion of many atoms in changing the state of a memristor [12]. To mitigate the impacts of devices with high variations, many adaptive algorithms such as retraining and remapping have been applied in prior work [12,14]. However, both the retraining and remapping cannot depart from the support of precise variation measurements at a post-silicon stage, which consumes huge testing costs and hardware overheads. Prior work [20,21] also employed a Bayesian optimization method to improve the robustness of analog DNNs, while presenting a thorough theoretical study about the interactions between fault tolerance and the neural network architecture choice (i.e., dropout, normalization, etc.) as well as its influence on the robustness. In contrast, Bayesian optimization in the above work [20,21] was used to automatically search for the optimal neural network architecture which would be robust to weight drifting, while the objective of our proposed BNN method is to learn the true posterior distribution presented by the memristor conductance states.

3. Preliminary Work on Bayesian Neural Networks

In this section, we give a brief introduction to Bayesian neural networks. Bayesian neural networks (BNNs) combine the principles of neural networks with the Bayesian inference technique, offering a probabilistic framework for modeling and learning from data. Unlike traditional neural networks that use fixed weights, BNNs assign probability distributions to the weights, thus allowing for uncertainty estimation in predictions and more robust decision making.

Specifically, the training process of a traditional NN model is viewed as performing a maximum likelihood estimation (MLE) [22,23]. The goal of MLE is to find a set of network parameters that maximizes the likelihood of observing the given data:

$$\mathbf{w}_{\text{MLE}} = \operatorname{argmin}_{\mathbf{w}} -\log P(\mathcal{D}|\mathbf{w}) \quad (1)$$

in which $P(\mathcal{D}|\mathbf{w})$ is the likelihood, representing the probability of observing the training data $\mathcal{D}$ given the model's parameters $\mathbf{w}$.

In contrast to MLE, which aims to find a set of point estimates of the optimal parameters, Bayesian inference directly computes the probability distribution of the weights that is also known as posterior:

$$P(\mathbf{w}|\mathcal{D}) = P(\mathbf{w})P(\mathcal{D}|\mathbf{w})/P(\mathcal{D}) \quad (2)$$

in which the prior distribution $P(\mathcal{D}|\mathbf{w})$ captures the initial beliefs about the weights before observing any data, reflecting the assumptions or knowledge about the weights' likely values.

Bayesian neural networks combine the prior distribution with the likelihood of the data given the weights to compute the posterior distribution $P(\mathbf{w}|\mathcal{D})$. The posterior distribution provides a more comprehensive representation of uncertainty compared to a single point estimate obtained through MLE. Instead of only finding the optimal point estimate of the weights, the posterior distribution represents the entire range of plausible values for the weights, along with their associated probabilities. In general, various techniques can be employed to compute the posterior distribution, such as Markov Chain Monte Carlo (MCMC) methods [24], variational inference [25,26], or other approximate inference techniques [27]. During the prediction phase, BNNs can use the posterior distribution to generate a distribution of predictions, thereby making more robust decisions in applications where uncertainty is critical.

To make a prediction given an input sample x^* , a traditional NN with deterministic weights $\mathbf{w}$ predicts the output as $P(y^*|x^*, \mathbf{w})$, while a BNN considers the weights as variational and makes the prediction by computing the expected value over the posterior distribution $P(\mathbf{w}|\mathcal{D})$: $P(y^*|x^*) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(y^*|x^*, \mathbf{w})]$. To address the intractable computation of the posterior distribution, previous research [24,26] suggested finding a variational approximation to the posterior on the weights, also known as variational inference.

Finding the variational distribution that is closest to the true posterior distribution corresponds to minimizing the Kullback–Leibler (KL) divergence between two probability distributions. In the context of BNNs, the KL divergence is used to quantify the discrepancy between the posterior distribution and the variational distribution. Mathematically, the learning process of BNNs can be formulated as minimizing the following objective function given the variational parameters θ :

$$\begin{aligned} \theta^* &= \operatorname{argmin}_{\theta} \text{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] \\ &= \operatorname{argmin}_{\theta} \int \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} d\mathbf{w} \\ &= \operatorname{argmin}_{\theta} \text{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})] + \log P(\mathcal{D}) \end{aligned} \quad (3)$$

Combined with the maximum likelihood estimation observing the data, the objective function of a Bayesian neural network model can be expressed as:

$$\mathcal{F}(\mathcal{D}, \theta) = \text{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})] = -\text{ELBO} \quad (4)$$

The above objective function is a sum of a data-dependent part (the former) referred to as the likelihood cost, and a prior-dependent part (the latter) referred to as the complexity cost. Taking the negative sign of the resulting objective function is known as the variational evidence lower bound (evidence lower bound, ELBO) [23]. The learning process is a trade-off between fitting the data $\mathcal{D}$ and satisfying the prior $P(\mathbf{w})$, subtly combining the flexibility and learning capabilities of neural networks with the probabilistic framework of Bayesian inference.

4. Proposed Methodology

4.1. Overview

To overcome the challenges of memristor deviations, we investigate and propose a design methodology for fault-tolerant neuromorphic computing using a Bayesian neural network. The proposed methodology mainly focuses on the faults incurred by process variations and noise at the device level, assuming other components and operations are error-free. Process variation refers to the inherent differences that occur during the fabrication of electronic components. Combined with the random fluctuations in the conductance values that occur during read and write operations, the weights involved in the forward propagation inevitably deviate from the offline trained values. These deviations can impact the performance and reliability of electronic systems, especially when precise and consistent conductance values are crucial for operation.

The intrinsic deviations in memristors can result in an imprecise deployment of a well-trained model on the corresponding synapse devices in one or more crossbar arrays. Correspondingly, the memristor conductances are no longer suitable to being treated as deterministic quantities. Therefore, we treat the weights as variational to make the weight distributions during the training process consistent with the programmed weights at memristor crossbars, thus compensating for the deviations through an adaptive training algorithm instead of post-calibration. This approach allows the model to capture and propagate the uncertainties in the weights and thus provides a more realistic representation of the model's behavior in memristor crossbar arrays. More specifically, variational inference is employed to approximate the posterior weight distributions, providing computationally efficient techniques to incorporate memristor deviations into the training process.

To establish a unified Bayesian neural network framework for a fault-tolerant neuromorphic computing system, as discussed in Section 3, there are two crucial components that need to be modified:

- Log-normal variational posterior: We introduce a log-normal variational posterior to quantify and characterize the inevitable deviations in synapse weights represented by memristor crossbars, attempting to accommodate process variations and noise through algorithmic training.
- Variational Bayesian inference: We employ variational inference to find the network parameters of the log-normal variational posterior that minimize the KL divergence between the variational posterior and the true Bayesian posterior of weights.

For the underlying system that deploys neural network models on imprecise and random conductance states of memristors, we need a unified framework to incorporate the impact of device variations into algorithmic training. By employing a variation-aware prior distribution over the weight space that follows a log-normal distribution, the BNN enforces the posterior to better fit the true distribution of conductance states through iterative training, capable of adaptively tolerating to weight drifting. This can be achieved by minimizing the KL divergence between the two, i.e., $\text{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})]$ in Equation (4). As expected, the proposed methodology unifies data observation with the fault tolerance capability, and is capable of finding an

optimal weight posterior approximation in one training session. Besides the introduction of the log-normal variational posterior, the underlying learnt uncertainty in the weights of BNN models can be also used to improve the fault tolerance to random noise.

4.2. Log-normal Variational Posterior

Generally speaking, deviations with different severities due to process variations are stochastically distributed in all the memristors in a crossbar array. Obviously, it is not enough to compensate their effects through retraining and/or remapping, which come at the huge hardware resources costs. It is therefore essential to develop a fault-aware learning algorithm to improve the robustness of NN models deployed on the hardware.

As discussed in Section 3, the Gaussian variational posterior is a typical case in Bayesian neural networks [24]. However, it is known that memristor device variation follows a log-normal distribution [11,12]. Specifically, the conductance of an on-state memristor can be expressed as $\mathbf{g} \rightarrow e^\lambda \cdot \mathbf{g}_0$, where $\lambda \sim \mathcal{N}(0, \sigma^2)$. $\mathbf{g}_0$ is equivalent to the ideal value obtained from a well-trained NN model without considering any faults. σ is the scaling factor, denoting the severity of process variations. Having liberated the algorithm from the confines of Gaussian priors and posteriors, we employ a simple log-normal prior combined with a variation-aware log-normal posterior.

Suppose that the variational posterior is a log-normal distribution, i.e., $\sim \mathcal{LN}(\mu, \sigma^2)$. Then, a sample of the weights $\mathbf{w}$ can be obtained by sampling a unit log-normal distribution, shifting it by a mean μ and scaling by a standard deviation σ . In the context of memristor-based neuromorphic computing, the weights represented by memristor conductances deviate from the expected/ideal values $\mathbf{w}_0$: $\mathbf{w} \rightarrow e^{\mathcal{N}(0, \sigma^2)} \cdot \mathbf{w}_0$. Thus, the synapse weight can be further expressed as a product of $\mathbf{w}_0$ and a sample of a log-normal distribution with the mean as zero and the standard deviations as σ .

Moreover, the pointwise standard deviation is parameterized as $\sigma = \log(1 + \exp(\rho))$ and is always non-negative. In the following, we represent the expected weights $\mathbf{w}_0$ as μ , which are not the mean of the above log-normal distribution. Thus, a log-normal sample of the weights can be transformed from a sample of random noise ϵ and the variational posterior parameters $\theta = (\mu, \rho)$ to:

$$\mathbf{w} = \mu \cdot e^{\log(1 + \exp(\rho)) \cdot \epsilon} \quad (5)$$

in which ϵ is sampled from a standard Gaussian distribution.

In this paper, we adopt the 1T1M-based memristor crossbar structure [28], where the relationship between weights and memristor conductances is $\mathbf{g} = \alpha \cdot \mathbf{w} + \beta$, where $\mathbf{w}$ is the expected weight vector learned from Equation (5) and $\mathbf{g}$ is the vector for the memristor conductances of the crossbar corresponding to $\mathbf{w}$. After the variational posterior approximation is accomplished, the expected weight vector $\mathbf{w}$ can be mapped and programmed on the memristor crossbar arrays.

4.3. Variational Bayesian Inference

The reality is that the programming of a memristor is an inherently random process, so the devices are not well suited to being treated as deterministic quantities. In Bayesian neural networks, model parameters are not single high-precision values but probability distributions. Then, we can deploy variational posterior distributions that are trained with both variation-aware prior knowledge and data observations of the corresponding devices in crossbar arrays. In this way, a weight represented by the programmed conductance state falls within the variational posterior of tolerated error. When performing inference, the weights can be considered as being sampled from the posterior distributions, enabling a natural pairing of the algorithm and device variations.

Similar to Section 3, we here use $P(\mathbf{w})$ as the prior to the BNN and $q(\mathbf{w}|\theta)$ as the estimated weight posterior. Reviewing the available literature [23], only expressions for the KL divergence between two distributions from the same distribution family can be found. For the objective function in Equation (4), the complexity cost, i.e., the KL divergence term,

needs to be derived with the prior and the variational posterior that are from the log-normal distribution family.

Without loss of generality, let the prior distribution $P \sim \mathcal{LN}(\mu_p, \sigma_p^2)$ and the posterior $Q \sim \mathcal{LN}(\mu_q, \sigma_q^2)$ with probability density functions p and q . Then, the KL divergence between two log-normal distributions is:

$$\begin{aligned} \text{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})] &= \int_0^\infty p \cdot \ln\left(\frac{p}{q}\right) = \mathbb{E}\left[\ln\left(\frac{p(P)}{q(P)}\right)\right] \\ &= \mathbb{E}\left[\ln\left(\frac{\sigma_q}{\sigma_p} \exp\left(\frac{(\ln(P) - \mu_q)^2}{2\sigma_q^2} - \frac{(\ln(P) - \mu_p)^2}{2\sigma_p^2}\right)\right)\right] \\ &= \ln\left(\frac{\sigma_q}{\sigma_p}\right) + \frac{1}{2\sigma_1^2}[(\mu_p - \mu_q)^2 + \sigma_p^2 - \sigma_q^2] \end{aligned} \quad (6)$$

For the posterior distribution $q(\mathbf{w}|\theta)$ and the prior $P(\mathbf{w})$ used in the proposed methodology, the mean values μ_q and μ_p of the above two log-normal distributions are zero; thus, the above formula can be simplified further:

$$\text{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})] = \ln\left(\frac{\sigma_q}{\sigma_p}\right) + \frac{1}{2\sigma_1^2}(\sigma_p^2 - \sigma_q^2) \quad (7)$$

Combined with the observed likelihood cost data, we can re-formulate the objective function of variational Bayesian inference. Each step of the optimization of the Bayesian neural network is as follows:

1. Sample $\epsilon \sim \mathcal{N}(0, 1)$.
2. Let the network parameters be $\mathbf{w} = \mu \cdot e^{\log(1+\exp(\rho)) \cdot \epsilon}$, where μ denotes the expected weight values that are error-free.
3. Let the variational parameters be $\theta = (\mu, \rho)$.
4. Let the objective function be $f(\mathbf{w}, \theta) = \text{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})]$.
5. Calculate the gradient with respect to the parameter μ :

$$\Delta_\mu = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} e^{\log(1+\exp(\rho)) \cdot \epsilon} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu}. \quad (8)$$

6. Calculate the gradient with the standard deviation parameter ρ :

$$\Delta_\rho = \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\mu \cdot \epsilon \cdot e^{\log(1+\exp(\rho)) \cdot \epsilon}}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho}. \quad (9)$$

7. Update the variational parameters μ and σ .

It is noted that the $\partial f(\mathbf{w}, \theta) / \partial \mathbf{w}$ of the gradients Δ_μ and Δ_ρ is shared and is exactly the gradients found by the usual backward propagation algorithm in a neural network. Then, the variational parameters can be learned from scaling and shifting the usual gradients as above. During forward/backward propagation, we adopt the minibatch technique for stochastic variational inference to average and propagate the gradients [26,29]. A typical local re-parameterization trick is also employed to save the computational cost in the variational Bayesian inference [26].

5. Evaluation

5.1. Experimental Setup

In our experiments, we utilize three commonly used neural network models, Multi-Layer Perceptron (MLP), LeNet, and AlexNet, and two widely used datasets: MNIST and CIFAR-10. These models represent different levels of complexity and have been widely employed in various applications. The classification accuracy is the measurement metric for our proposed methodology. Furthermore, the robustness with respect to the severity of

variations can be also measured by the relative inference accuracy degradation between the cases with and without variations. The lower the degradation in the inference accuracy, the better the fault tolerance capability our methodology offers. The proposed Bayesian neural network framework with a log-normal prior and posterior distributions and its training setup were modified on the basis of the github repository (<https://github.com/kumar-shridhar/PyTorch-BayesianCNN>, accessed on 10 January 2021). The key innovations of this paper involve the modeling of the log-normal variational posterior and the derivation of the KL divergence between the variational posterior and the true posterior.

In the mentioned learning task, the training of BNN models is performed using the Adam [30] optimizer with specific hyperparameters (learning rate: 0.001, batch size: 64, epochs: 100). For the neuromorphic computing hardware, we adopted a similar setup to that in the literature [11], where each layer of the NN models can be mapped to one or more memristor crossbar arrays of size 64×64 . By considering the hardware implementation using memristor crossbar arrays and accounting for the effects of process variation, the experiments aim to demonstrate the effectiveness of the proposed design methodology in achieving an efficient and robust classification performance.

5.2. Experimental Results

We first explore how the proposed methodology can compensate for deviations induced by process variations. Figure 3a,b presents a comparison of the inference performance of two neural network models, LeNet on MNIST and AlexNet on CIFAR-10, under different process variations. According to the relationship between the ideal weight and the programmed weight, i.e., $\mathbf{w} \leftarrow \mathbf{w}_0 \cdot e^{\mathcal{N}(0, \sigma^2)}$, we use the standard deviation σ (e.g., the X-axes in the figures) to indicate the severity of deviations on the memristor conductances in the experiments.

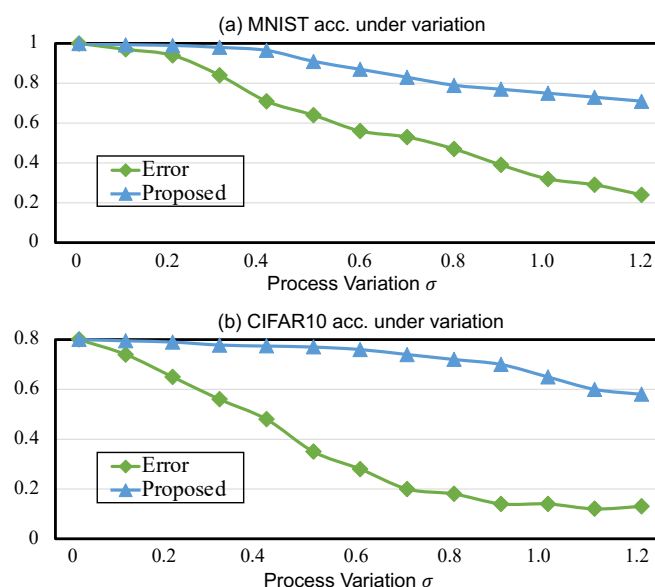


Figure 3. Comparison of inference accuracy under different process variations (σ) without/with the proposed methodology: (a) LeNet on MNIST and (b) AlexNet on CIFAR-10.

Compared to the baselines (green lines in Figure 3), which suffer from a great loss in accuracy as σ increases, the proposed methodology can reduce the inference degradation for MNIST and CIFAR-10 classification tasks up to 47% and 42%, respectively. Thus, the execution models of memristor crossbar arrays trained using the BNN method present overwhelming advantages in terms of the inference performance, enabling more stable computational effectiveness against the uncertainties of process variations. In Figure 3a, under different levels of deviations (e.g., σ is increased from 0.1 to 0.4), the accuracy of MNIST remains stable at almost 0 inference degradation. This phenomenon is also present

in the inference performance on CIFAR-10 in Figure 3b. By comparing Figure 3a with Figure 3b, AlexNet presents better fault tolerance than LeNet, in which the accuracy on CIFAR-10 can reach over 75% within $\sigma = 0.6$. This indicates that the deeper network has a better healing capability and thus is more robust against uncertainties in the weights.

We further explore the tolerance of the proposed methodology to random noise. Even if the weights are well trained and process variations are accounted for, the procedures of mapping the intended conductances to devices and executing the inference may still suffer from a certain level of perturbation. Thus, both process variations and noise can contribute to the conductance deviations from the intended values, which eventually results in an inference accuracy degradation from the algorithmic level evaluation. Considering that both the read noise and write noise follow a Gaussian distribution [31], the actual weight is indeed a sum of the log-normal and the Gaussian noise, where the standard deviation of Gaussian noise was set to 5% of the nominal values based upon past experience [13].

We then comprehensively evaluated the overall robustness when suffering from the combined impacts of process variations and noise. Figure 4a,b compares the inference accuracy under different amounts of process variations as well as 5% Gaussian noise for the aforementioned two NN models. By employing the proposed methodology, both models can reduce the accuracy loss by up to $\sim 30\%$ across the whole range of σ . However, for the baseline plotted in Figure 4a, when the variation effects are slightly larger, i.e., $\sigma \geq 0.5$, the accuracy decreases by 40%. Obviously, the effects of noise further degrade the accuracy by comparing Figure 4 with Figure 3. Impressively, a smaller accuracy gap, i.e., $\sim 5\%$ for LeNet-MNIST, can be maintained when $\sigma \leq 0.4$ and the noise is within 5%, which is considered as a reliable fault tolerance range.

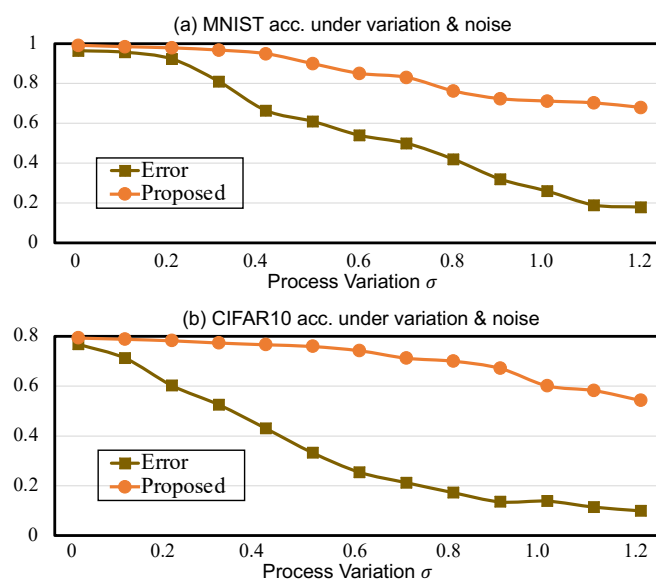


Figure 4. Comparison of inference accuracy under the combined effects of process variations (σ) and noise without/with the proposed methodology: (a) LeNet on MNIST and (b) AlexNet on CIFAR-10.

Finally, we compared the performance of the proposed methodology with other state-of-the-art methods [11]. Ref. [11] implemented a fault-tolerant framework, FTNNA, by replacing the original classifiers in NN models with a novel collaborative logistic classifier to improve its error-correction capability. This comparison allows for an assessment of the effectiveness and competitiveness of the proposed framework in relation to existing approaches. For a fair comparison, the setup of the proposed framework was consistent with [11], while the reported inference results of FTNNA were taken directly from the original paper. As shown in Table 1, it is clear that the proposed framework outperforms FTNNA by 4~7% in terms of accuracy across the whole range of process variation σ , even without involving retraining/remapping.

Table 1. Comparison of inference accuracy for MLP-MNIST under different variations between the proposed methodology and a state-of-the-art method [11].

Method	Process Variation σ				
	0.2	0.4	0.6	0.8	1.0
FTNNA [11]	94%	90%	82%	77%	73%
Proposed	99%	97%	89%	81%	77%

6. Conclusions

This paper proposes a novel design methodology based on a Bayesian neural network to improve the fault tolerance of neuromorphic computing systems. The proposed methodology creatively combines the variational Bayesian inference technique with a fault-aware variational posterior distribution to accommodate the underlying deviations in memristors through one BNN training process. The experimental results show that the proposed framework can effectively rectify accuracy losses of up to $\sim 40\%$ due to process variations and stochastic noise without involving expensive retraining and remapping.

Author Contributions: Conceptualization, D.G., X.X. and D.W.; methodology, D.G. and D.W.; software, D.G.; validation, D.G.; formal analysis, D.G.; investigation, D.G. and D.W.; resources, D.G. and D.W.; data curation, D.G.; writing—original draft preparation, D.G.; writing—review and editing, D.G., X.X. and D.W.; visualization, D.G.; supervision, D.G. and D.W.; project administration, D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Chen, W.H.; Li, K.X.; Lin, W.Y.; Hsu, K.H.; Li, P.Y.; Yang, C.H.; Xue, C.X.; Yang, E.Y.; Chen, Y.K.; Chang, Y.S.; et al. A 65 nm 1 Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors. In Proceedings of the 2018 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 11–15 February 2018; pp. 494–496.
- Song, L.; Qian, X.; Li, H.; Chen, Y. Pipelayer: A pipelined reram-based accelerator for deep learning. In Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, TX, USA, 4–8 February 2017; pp. 541–552.
- Gokmen, T.; Onen, M.; Haensch, W. Training deep convolutional neural networks with resistive cross-point devices. *Front. Neurosci.* **2017**, *11*, 538. [\[CrossRef\]](#)
- Shafiee, A.; Nag, A.; Muralimanohar, N.; Balasubramanian, R.; Strachan, J.P.; Hu, M.; Williams, R.S.; Srikumar, V. ISAAC: A convolutional neural network accelerator with in situ analog arithmetic in crossbars. In Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Republic of Korea, 18–22 June 2016; Volume 44, pp. 14–26.
- Jain, S.; Sengupta, A.; Roy, K.; Raghunathan, A. RxNN: A framework for evaluating deep neural networks on resistive crossbars. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *40*, 326–338. [\[CrossRef\]](#)
- Roy, S.; Sridharan, S.; Jain, S.; Raghunathan, A. TxSim: Modeling training of deep neural networks on resistive crossbar systems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 730–738. [\[CrossRef\]](#)
- Visweswariah, C.; Ravindran, K.; Kalafala, K.; Walker, S.G.; Narayan, S.; Beece, D.K.; Piaget, J.; Venkateswaran, N.; Hemmett, J.G. First-order incremental block-based statistical timing analysis. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2006**, *25*, 2170–2180. [\[CrossRef\]](#)
- Xiong, J.; Zolotov, V.; He, L. Robust extraction of spatial correlation. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2007**, *26*, 619–631. [\[CrossRef\]](#)
- Liu, M.; Xia, L.; Wang, Y.; Chakrabarty, K. Fault tolerance in neuromorphic computing systems. In Proceedings of the 24th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 21–24 January 2019; pp. 216–223.
- Li, B.; Yan, B.; Liu, C.; Li, H. Build reliable and efficient neuromorphic design with memristor technology. In Proceedings of the 24th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 21–24 January 2019; pp. 224–229.
- Liu, T.; Wen, W.; Jiang, L.; Wang, Y.; Yang, C.; Quan, G. A fault-tolerant neural network architecture. In Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6.

12. Liu, B.; Li, H.; Chen, Y.; Li, X.; Wu, Q.; Huang, T. Vortex: Variation-aware training for memristor X-bar. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 8–12 June 2015; pp. 1–6.
13. Zhu, Y.; Zhang, G.L.; Wang, T.; Li, B.; Shi, Y.; Ho, T.Y.; Schlichtmann, U. Statistical training for neuromorphic computing using memristor-based crossbars considering process variations and noise. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 1590–1593.
14. Chen, L.; Li, J.; Chen, Y.; Deng, Q.; Shen, J.; Liang, X.; Jiang, L. Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 19–24.
15. Pouyan, P.; Amat, E.; Rubio, A. Memristive crossbar memory lifetime evaluation and reconfiguration strategies. *IEEE Trans. Emerg. Top. Comput.* **2016**, *6*, 207–218. [[CrossRef](#)]
16. Rehman, M.M.; Rehman, H.M.M.U.; Gul, J.Z.; Kim, W.Y.; Karimov, K.S.; Ahmed, N. Decade of 2D-materials-based RRAM devices: A review. *Sci. Technol. Adv. Mater.* **2020**, *21*, 147–186. [[CrossRef](#)] [[PubMed](#)]
17. Rehman, M.M.; ur Rehman, H.M.M.; Kim, W.Y.; Sherazi, S.S.H.; Rao, M.W.; Khan, M.; Muhammad, Z. Biomaterial-based nonvolatile resistive memory devices toward ecofriendliness and biocompatibility. *ACS Appl. Electron. Mater.* **2021**, *3*, 2832–2861. [[CrossRef](#)]
18. Lin, M.Y.; Cheng, H.Y.; Lin, W.T.; Yang, T.H.; Tseng, I.C.; Yang, C.L.; Hu, H.W.; Chang, H.S.; Li, H.P.; Chang, M.F. DL-RSIM: A simulation framework to enable reliable ReRAM-based accelerators for deep learning. In Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, USA, 5–8 November 2018; pp. 1–8.
19. Hu, M.; Strachan, J.P.; Li, Z.; Grafals, E.M.; Davila, N.; Graves, C.; Lam, S.; Ge, N.; Yang, J.J.; Williams, R.S. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. In Proceedings of the 2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 5–9 June 2016; pp. 1–6.
20. Ye, N.; Mei, J.; Fang, Z.; Zhang, Y.; Zhang, Z.; Wu, H.; Liang, X. BayesFT: Bayesian Optimization for Fault Tolerant Neural Network Architecture. In Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 5–9 December 2021; pp. 487–492.
21. Ye, N.; Cao, L.; Yang, L.; Zhang, Z.; Fang, Z.; Gu, Q.; Yang, G.Z. Improving the robustness of analog deep neural networks through a Bayes-optimized noise injection approach. *Commun. Eng.* **2023**, *2*, 25. [[CrossRef](#)]
22. Graves, A. Practical variational inference for neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Granada, Spain, 12–15 December 2011; pp. 2348–2356.
23. Blei, D.M.; Kucukelbir, A.; McAuliffe, J.D. Variational inference: A review for statisticians. *J. Am. Stat. Assoc.* **2017**, *112*, 859–877. [[CrossRef](#)]
24. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight uncertainty in neural networks. *arXiv* **2015**, arXiv:1505.05424.
25. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
26. Kingma, D.P.; Salimans, T.; Welling, M. Variational dropout and the local reparameterization trick. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2575–2583.
27. Atanov, A.; Ashukha, A.; Struminsky, K.; Vetrov, D.; Welling, M. The deep weight prior. *arXiv* **2018**, arXiv:1810.06943.
28. Chen, C.Y.; Shih, H.C.; Wu, C.W.; Lin, C.H.; Chiu, P.F.; Sheu, S.S.; Chen, F.T. RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme. *IEEE Trans. Comput.* **2014**, *64*, 180–190. [[CrossRef](#)]
29. Tomczak, J.; Welling, M. VAE with a VampPrior. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Playa Blanca, Spain, 9–11 April 2018; pp. 1214–1223.
30. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
31. Agarwal, S.; Plimpton, S.J.; Hughart, D.R.; Hsia, A.H.; Richter, I.; Cox, J.A.; James, C.D.; Marinella, M.J. Resistive memory device requirements for a neural algorithm accelerator. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 929–938.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.