Spatial-SpinDrop: Spatial dropout-based binary Bayesian neural network with spintronics implementation

Soyed Tuhin Ahmed, Kamal Danouchi, Michael Hefenbrock, Guillaume Prenat, Lorena Anghel, Mehdi Tahoori

Spatial-SpinDrop: Spatial Dropout-based Binary Bayesian Neural Network with Spintronics Implementation

Soyed Tuhin Ahmed[‡‡†], Kamal Danouchi[‡], Michael Hefenbrock[§], Guillaume Prenat[‡], Lorena Anghel[‡], Mehdi B. Tahoori[†]
[†]Karlsruhe Institute of Technology, Karlsruhe, Germany, [‡‡]corresponding author, email: soyed.ahmed@kit.edu
[‡]Univ. Grenoble Alpes, CEA, CNRS, Grenoble INP, and IRIG-Spintec, Grenoble, France [§]RevoAI GmbH, Karlsruhe, Germany

***Abstract*—Recently, machine learning systems have gained prominence in real-time, critical decision-making domains, such as autonomous driving and industrial automation. Their implementations should avoid overconfident predictions through uncertainty estimation. Bayesian Neural Networks (BayNNs) are principled methods for estimating predictive uncertainty. However, their computational costs and power consumption hinder their widespread deployment in edge AI. Utilizing Dropout as an approximation of the posterior distribution, binarizing the parameters of BayNNs, and further to that implementing them in spintronics-based computation-in-memory (CiM) hardware arrays provide can be a viable solution. However, designing hardware Dropout modules for convolutional neural network (CNN) topologies is challenging and expensive, as they may require numerous Dropout modules and need to use spatial information to drop certain elements. In this paper, we introduce MC-SpatialDropout, a spatial dropout-based approximate BayNNs with spintronics emerging devices. Our method utilizes the inherent stochasticity of spintronic devices for efficient implementation of the spatial dropout module compared to existing implementations. Furthermore, the number of dropout modules per network layer is reduced by a factor of $9\times$ and energy consumption by a factor of $94.11\times$, while still achieving comparable predictive performance and uncertainty estimates compared to related works.**

***Index Terms*—MC-Dropout, Spatial Dropout, Bayesian neural network, Uncertainty estimation, Spintronic**

I. Introduction

NEURAL NETWORKS are brain-inspired computational methods that, in some cases, can even outperform human counterparts [1]. Consequently, applications of NNs have increased rapidly in recent years and have become the cornerstone of modern computing paradigms. Furthermore, NNs are commonly deployed in real-time safety-critical tasks such as computer-aided medical diagnostics, industrial robotics, and autonomous vehicles.

Conventional (point estimate) Neural Networks (NNs) typically learn a single point value for each parameter. However, they do not account for the uncertainty in the data nor in the model, leading to overconfident predictions and in turnn to safety violations. This is particularly true when the data generation process is noisy or the training data is either incomplete or insufficient to capture the complexity of the actual phenomenon being modelled. In safety-critical domains where machine learning systems make human-centered decisions, an uncertainty measure is essential for informed decision-making.

On the other hand, Bayesian Neural Networks (BayNNs), which put prior distributions over the model parameters and learn the posterior distribution using approximation techniques (e.g., Monte Carlo (MC)-Dropout [2]), present a systematic method for training uncertainty-aware neural networks. However, the computational costs and high-performance requirements of BayNNs can be prohibitive for edge devices.

Therefore, dedicated NN hardware accelerators such as Compute-in-Memory (CiM) architectures with emerging Non-Volatile resistive Memories (NVMs) have been explored. CiM architectures enable the Matrix-Vector Multiplication (MVM) operation of NNs to be carried out directly inside the memory, overcoming the memory limitations of traditional von-Neumann architectures. Among the NVM technologies, Spin-Transfer-Torque Magnetic Random Access Memory (STT-MRAM) is particularly appealing due to its nanosecond latency, high endurance (10^{12} cycles), and low switching energy (10 fJ) [3].

Additionally, algorithmic approaches, such as Binarization which typically reduces the bit precision of NNs to 1-bit, lead to smaller computational time and model size. Therefore, they are an attractive options for BayNNs to mitigate their inherent costs. Moreover, this approach allows for the direct mapping of BayNN parameters to STT-MRAM-based CiM hardware.

Existing work [4], [5] proposed to binarize the parameters of BayNNs and implement them on STT-MRAM-based CiM hardware resulting in a highly efficient solution. Although this approach can achieve high algorithmic performance and hardware efficiency compared to existing works, designing Dropout modules in the case of convolutional NN (CNN) topologies is challenging and expensive due to the nature of implementation.

In this paper, we present an algorithm-hardware co-design approach that not only solves the challenges of implementing the Dropout-based BayNNs approach, but also reduces the number of Dropout modules required per layer. The main contributions of this paper are as follows:

- We propose *MC-SpatialDropout*, which uses spatial Dropout for Bayesian approximation. Our method is mathematically equivalent to the MC-Dropout-based approach, enabling uncertainty-aware predictions.
- We present an STT-MRAM-based CiM architecture for the proposed *MC-SpatialDropout*-based BayNNs. Our approach leverages the inherent stochasticity of STT-MRAM for the Dropout module and deterministic behavior for parameter storage. This allows the reuse of the array designed for conventional binary NNs (BNNs), and only the peripheral circuitry is adapted for Bayesian inference.
- We also propose reliable and adaptable sensing scheme for stochastic STT-MRAM specifically designed to implement the dropout concept for both linear and convolutional layers.

Our method is targeting CNN topologies and reduces the number of Dropout modules in a layer by $9\times$ and energy consumption by $94.11\times$, while maintaining comparable predictive performance and uncertainty estimates.

The remainder of this paper is organized as follows: Section II provides the background for our work, Section III describes the proposed MC-SpatialDropout, Section IV presents both the algorithmic and hardware results for our approach and finally, in Section V, we conclude the paper.

II. Background

A. Spintronics

MRAM have gained significant attention due to their fast switching, high endurance, and CMOS compatibility [6]. The main component of MRAM devices is the Magnetic Tunnel Junction (MTJ), which comprises two ferromagnetic layers: the reference layer and the free layer, separated by a thin insulating layer. The magnetization of the reference layer is fixed in one direction, while the free layer can have its magnetization reversed between two stable positions: parallel or antiparallel to that of the reference layer. The resistance of the stack depends on the relative orientations of the layer magnetizations, with a high resistance state in the antiparallel configuration and a low resistance state in the parallel configuration.

B. Uncertainty in Deep Learning

Uncertainty estimation is vital in deep learning, especially for safety-critical applications, as it provides insight into the model's confidence in its predictions, enhancing the trustworthiness of decision-making. There are two main types of uncertainty: epistemic, which results from the limitations of the model and can be reduced with more data or improved architectures, and aleatoric, which arises from noise in the data and cannot be mitigated. Obtaining uncertainty estimates bolsters robustness by identifying out-of-distribution (OOD) data points and avoiding overconfident predictions. OOD data refers to data whose distribution is completely different from the training (in-distribution (ID)) data. In this paper, we focus on aleatoric uncertainty estimation and evaluate the effectiveness of our method for OOD detection.

C. Bayesian NNs

BayNNs offer a principled approach to uncertainty estimation in neural networks. Several approximation methods exist for BayNNs, such as variational inference and Markov Chain Monte Carlo methods.

One popular approximation technique is Monte Carlo Dropout (MC-Dropout), which leverages dropout for Bayesian inference. Dropout [7] is a common regularization technique used to reduce overfitting and neuron co-adaptation by randomly setting neuron outputs to zero during training. The dropout operation can be described as $\hat{\mathbf{Z}} = \mathbf{M} \odot \mathbf{Z}$, where $\mathbf{M}$ is a binary mask generated by sampling from a Bernoulli distribution, $\odot$ represents element-wise multiplication, and $\mathbf{Z}$ and $\hat{\mathbf{Z}}$ are intermediate activation and dropped out intermediate activation of a layer, respectively.

MC-Dropout provides an approximation of the true posterior distribution with relatively low computational and memory overhead compared to other methods such as variational inference (VI) [8] and the ensemble approach [9]. This is because the ensemble approach requires inference in multiple NNs, and VI requires learning the parameters of the variational distribution, which require storage. Since the MC-Dropout method has the same number of parameters as conventional NNs, it leads to minimal additional computation and memory requirements, making it suitable for a wide range of applications, including those with limited resources.

The optimization objective for MC-Dropout can be represented as

$$\mathcal{L}(\boldsymbol{\theta})_{\text{MC-Dropout}} = \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) + \lambda \sum_{l=1}^{L} ||\boldsymbol{\theta}_l||_2^2 \tag{1}$$

where $\mathcal{L}(\boldsymbol{\theta}, \mathcal{D})$ represents the task-specific loss function, such as categorical cross-entropy for classification or mean squared error for regression, and $\|\boldsymbol{\theta}\|_2^2$ is the regularization term. Also, $\boldsymbol{\theta}$ summarizes all learnable parameters, i.e., $\boldsymbol{\theta} = \{\mathbf{W}_l, \mathbf{b}_l \mid l = 1, \cdots, L\}$, where $\mathbf{W}_l$ denote the weight matrices and $\mathbf{b}_l$ the biases for the layer l. During inference, dropout is applied multiple times, and the outputs are averaged to obtain the predictive distribution. Hence, the posterior predictive distribution over the output y, i.e.,

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \tag{2}$$

is approximated by

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{M}_t) \quad \text{with} \quad \mathbf{M}_t \sim \mathcal{B}(\rho). \tag{3}$$

Here, $\mathcal{D}$ denotes the dataset, $\mathbf{x}$ is the input, $\mathbf{y}$ is the output, and the entries of $\mathbf{M}_t$ are independently sampled from a Bernoulli distribution with (dropout) probability ρ.

D. Mapping of Convolutional Layers to CiM Architecture

To perform the computation inside the CiM architecture, a critical step is the mapping of the different layers of the NN to crossbar arrays. Standard NNs contain mainly Fully Connected (FC) layers and convolutional layers. While the mapping of FC layers is straightforward in a crossbar array as the shape of the weight matrices is 2D ($\mathbb{R}^{m \times n}$), mapping convolutional layers is challenging due to their 4D shapes ($\mathbb{R}^{K \times K \times C_{in} \times C_{out}}$). Here, K denotes the shape of kernels, and C_{in} represents the number of input channels. Implementing convolutional layers requires implementing multiple kernels with different shapes and sizes.

There are two popular mapping strategies for mapping the convolutional layer exists. In the mapping strategy ①, each kernel of shape $K \times K \times C_{in}$ is unrolled to a column of the crossbar [10]. On the other hand, in the mapping strategy ②, each kernel is mapped to $K \times K$ smaller crossbars with a shape of $C_{in} \times C_{out}$ [11].

III. Proposed Method

A. Problem Statement and Motivation

The convolution operation is performed differently in CiM architectures compared to GPUs. In CiM architectures, moving windows (MWs) with a shape of $K \times K$ are applied to each input feature map (IFM) in one cycle (see Fig. 1(a)). In the next cycle, the MWs will "slide over" the IFMs with a topology-defined stride S for N cycles. Assuming $K > S$, some of the elements in the MWs for the next $K - S$ cycles will be the same as in the previous cycles, a concept known as

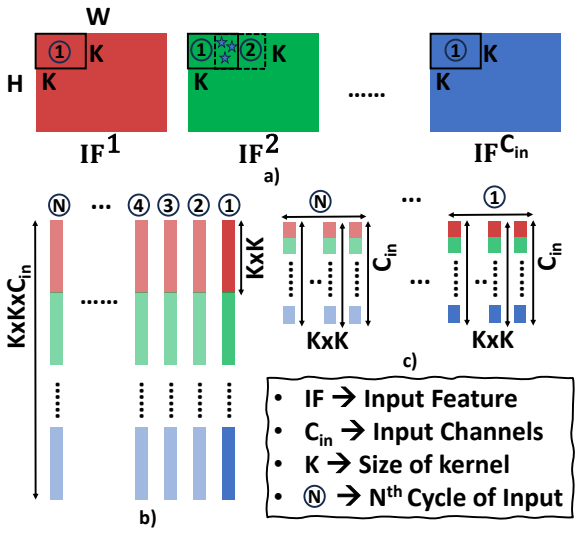

Fig. 1. a) Input feature map of a convolutional layer, b) moving windows from all the input feature maps are flattened for the conventional mapping, c)

weight sharing. This is illustrated by the green input feature (IF) in Fig. 1(a).

The Dropout module designed in [4], [5] drops each element of the MWs with a probability P in each cycle. Therefore, it essentially re-samples the dropout mask of each MW of IFMs in each cycle. Consequently, the dropout masks of the shared elements in the MWs will change in each input cycle, leading to inconsistency. An ideal Dropout module should only generate dropout masks for new elements of the MWs. Designing a Dropout module that drops each element of the MWs depending on the spatial location of the MWs in the IFMs is challenging and may lead to complex circuit design. Additionally, the number of rows in crossbars typically increases from one layer to another due to the larger C_{in}. Consequently, the number of Dropout modules required will be significantly higher.

Furthermore, the MWs are reshaped depending on the weight mapping discussed in Section II-D. For mapping strategy ①, the MWs from IFMs are flattened into a vector of length $K \times K \times C_{in}$. However, for mapping strategy ②, IFMs are flattened into $K \times K$ vectors of length C_{in}, as depicted in Fig. 1(a) and (b). As a result, designing a generalizable Dropout model is challenging.

B. MC-SpatialDropout as Bayesian Approximation

In an effort to improve the efficiency and accuracy of Bayesian approximation techniques, we propose the MC-SpatialDropout method. The proposed MC-SpatialDropout technique expands upon the MC-Dropout [2] and MC-SpinDrop [4], [5] methods by utilizing spatial dropout as a Bayesian approximation. Our approach drops an entire feature with a probability p. This means that all the elements of a feature map in Fig. 1(a) are dropped together. However, each feature map is dropped independently of the others. As a result, the number of Dropout modules required for a layer will be significantly reduced, and the design effort of the dropout module will also be lessened.

The primary objective of this approach is to address the shortcomings of MC-Dropout arising from its independent treatment of elements of the features. In contrast, MC-SpatialDropout exploits the spatial correlation of IFs, which is particularly advantageous for tasks involving image or spatial data. By doing so, it facilitates a more robust and contextually accurate approximation of the posterior distribution. This enables the model to capture more sophisticated representations and account for dependencies between features.

In terms of the objective function for the MC-SpatialDropout, Soyed et al. [4], [5] showed that minimizing the objective function of MC-Dropout (see Equation (1)) is not beneficial for BNNs and suggested a BNN-specific regularization term. In this paper, instead of defining a separate loss function for MC-SpatialDropout, we define the objective function as:

$$\mathcal{L}(\boldsymbol{\theta})_{\text{MC-SpatialDropout}} = \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) + \lambda \sum_{l=1}^{L} ||\mathbf{W}_l||_2^2. \quad (4)$$

Therefore, the objective function is equivalent to Equation (1) for MC-Dropout. However, the second part of the objective function is the regularization term applied to the (real valued) "proxy" weights ($\mathbf{W}_l$) of BNN instead of binary weights. It encourages $\mathbf{W}_l$ to be close to zero. By keeping a small value for the λ, it implicitly ensures that the distribution of weights is centered around zero. Also, we normalize the weights by

$$\bar{\mathbf{W}}_l = \frac{\mathbf{W}_l - \mu_l^{\mathbf{W}}}{\sigma_l^{\mathbf{W}}}, \quad (5)$$

to ensure, the weight matrix has zero mean and unit variance before binarization. Where $\mu^{\mathbf{W}}$ and $\sigma^{\mathbf{W}}$ are the mean and variance of the weight matrix of the layer l. This process allows

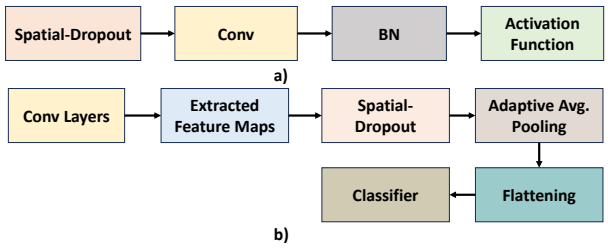

Fig. 2. Block diagram of the location of the proposed MC-SpatialDropout in a) a layer-wise fashion, b) a topology specific fashion.

applying L2 regularization in BNN training and [12] showed that it improves inference accuracy by reducing quantization error. Since our work is targeted for BNN, regularization is only applied to the weight matrixes.

The difference is that our method approximate Equation (2) by:

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}, \hat{\mathbf{M}}_t) \quad \text{with} \quad \hat{\mathbf{M}}_t \sim \mathcal{B}(\rho). \quad (6)$$

Here, during training and Bayesian inference, the dropout mask $\hat{\mathbf{M}}_t$ sampled spatially correlated manner for the output feature maps (OFMs) of each layer from a Bernoulli distribution with (dropout) probability ρ. The dropout masks correspond to whether a certain spatial location in the OFMs (i.e., a certain unit) is dropped or not.

For Bayesian inference, we perform T Monte Carlo sampling to approximate the posterior distribution. Each Monte Carlo sample corresponds to forward passing the input $\mathbf{x}$ through the NN with unique spatial dropout masks $\hat{\mathbf{M}}_t \quad t = 1, \cdots, T$, resulting in a diverse ensemble of networks. By averaging the predictions from the Monte Carlo samples, we effectively perform Bayesian model averaging to obtain the final prediction.

Proper arrangement of layers is important for the MC-SpatialDropout based Bayesian inference. The Spatial Dropout layer can be applied before each convolutional layer in a layerwise MC-SpatialDropout method. Additionally, the Spatial Dropout layer can be applied to the extracted features of a CNN topology in a topology-wise MC-SpatialDropout method. Fig. 2 shows the block diagram for both approaches.

C. Designing Spatial-SpinDrop Module

As mentioned earlier, in the proposed MC-SpatialDropout, feature maps can independently be dropped with a probability p. Due to the nature of input application in CiM architectures, this implicitly means dropping different regions of crossbars depending on the mapping strategy. These challenges are associated with designing the Dropout module for the proposed MC-SpatialDropout based BayNN.

For the mapping strategy ①, as depicted in Fig. 1(b), each $K \times K$ subset of input comes from a feature map. This means that if an input feature is dropped, the corresponding $K \times K$ subset of input should also be dropped for all C_{out} and N cycles of inputs. This implies that dropping each $K \times K$ row of a crossbar together for N cycles is equivalent to applying spatial dropout. However, each group of rows should be dropped independently of one another. Additionally, their dropout mask should be sampled only in the first cycle. For the remaining $N-1$ cycles of input, the dropout mask should remain consistent.

In contrast, in the mapping strategy ② (see Fig. 1(c)), the elements of a MW are applied in parallel to each $K \times K$ crossbar at the same index. As a result, dropping an IF would lead to dropping each index of rows in all the $K \times K$ crossbars together. Similarly, each row of a crossbar is dropped independently of one another, and the dropout mask is sampled

at the first input cycle and remains consistent for the remaining $N-1$ cycles of input.

Furthermore, if the spatial dropout is applied to the extracted feature maps of a CNN (see Fig. 2), then depending on the usage of the adaptive average pool layer, the design of the Spin-SpatialDrop will differ. If a CNN topology does not use an adaptive average pool layer, then $H \times W$ groups of rows are dropped together. This is because the flattening operation essentially flattens each IF into a vector. These vectors are combined into a larger vector representing the input for the classifier layer. However, since input for the FC layer is applied in one cycle only, there is no need to hold the dropout mask. The Spin-SpatialDrop module for the mapping strategy ① can be adjusted for this condition.

Lastly, if a CNN topology does use an adaptive average pool layer, then the SpinDrop module proposed by [4], [5] can be used. This is because the adaptive average pool layer averages each IF to a single point, giving a vector with total C_{out} elements.

Therefore, the Dropout module for the proposed MC-SpatialDropout should be able to work in four different configurations. Consequently, we propose a novel spintronic-based spatial Dropout design, called *Spatial-SpinDrop*.

The Spatial-SpinDrop module leverages the stochastic behavior of the MTJ for spatial dropout. The proposed scheme is depicted in Fig. 3. In order to generate a stochastic bitstream using the MTJ, the first step involves a writing scheme that enables the generation of a bidirectional current through the device. This writing circuit consists of four transistors, allocated to a "SET" and a "RESET" modules. The "SET" operation facilitates the stochastic writing of the MTJ, with a probability corresponding to the required dropout probability. On the other hand, the "RESET" operation restores the MTJ to its original state. During the reading operation of the MTJ, the resistance of the device is compared to a reference element to determine its state. The reference resistance value is chosen such as it falls between the parallel and anti-parallel resistances of the MTJ.

For the reading phase, a two-stage architecture is employed for better flexibility and better control of the reading phase for the different configurations discussed earlier. The module operates as follows: after a writing step in the MTJ, the signal V_{pol} allows a small current to flow through the MTJ and the reference cell (*REF*), if and only if the signal $hold$ is activated. Thus, the difference in resistance is translated into a difference in voltages (V_{MTJ} and V_{ref}). The second stage of the amplifier utilizes a StrongARM latch structure [13] to provide a digital representation of the MTJ state. The *Ctrl* signal works in two phases. When *Ctrl = 0*, $\overline{Out}$ and Out are precharged at *VDD*. Later, when *Ctrl = 1*, the discharge begins, resulting in a differential current proportional to the gate voltages (V_{MTJ} and V_{ref}). The latch converts the difference of voltage into two opposite logic states in $\overline{Out}$ and Out. Once the information from the MTJ is captured and available at the output, the signal $hold$ is deactivated to anticipate the next writing operation. To enable the dropout, a series of AND gates and transmission gates are added, allowing either access to the classical decoder or to the stochastic word-line (WL).

As long as the $hold$ signal is deactivated, no further reading operation is permitted. Such a mechanism allows the structure to maintain the same dropout configuration for a given time and will be used during $N-1$ cycles of inputs to allow the dropping of the IF in strategies ① and ②. In the first strategy, the AND gate receives as input $K \times K$ WLs from the same decoder, see Fig. 4(a). While in strategy ②, the AND gate receives one row per decoder, as presented in Fig. 4(b).

For the last two configurations, the $hold$ signal is activated for each reading operation, eliminating the need to maintain the dropout mask for $N-1$ cycles.

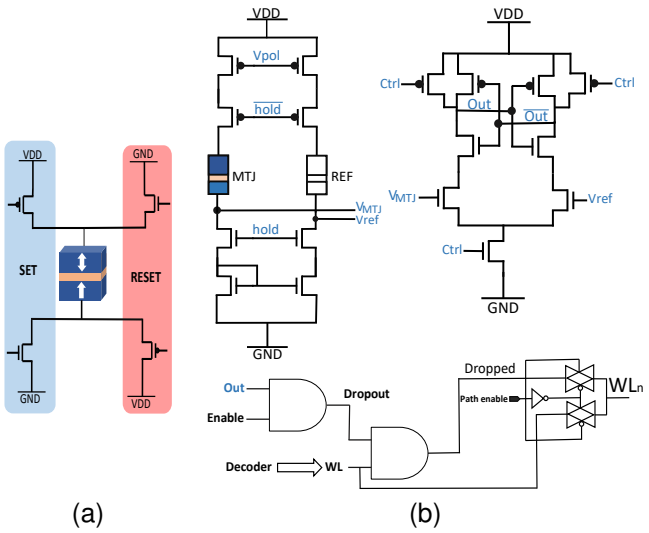

Fig. 3. (a) Writing and (b) reading schemes for the MTJ.

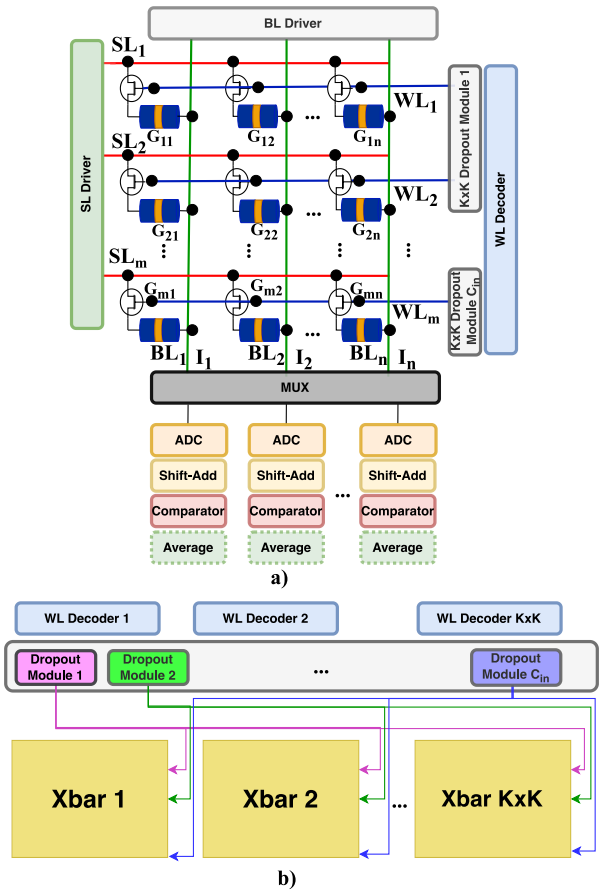

Fig. 4. Crossbar design for the MC-SpatialDropout based on mapping strategy (a) ① and (b) strategy②. In (b), only the Dropout module and WL decoder are shown, Everything else is abstracted.

D. MC-SpatialDropout-Based Bayesian Inference in CiM

The proposed MC-SpatialDropout-Based Bayesian inference can be leveraged on the two mapping strategies discussed in Section II-D. In both strategies, one or more crossbar arrays with MTJs at each crosspoints are employed in order to encode the binary weights into the resistive states of the MTJs.

Specifically, for the mapping strategy ①, we divide the WLs of the crossbar into $K \times K$ groups and connect one dropout module to each group, as shown in Fig.4(a). In Fig. 3(b), this strategy involves connecting $K \times K$ WLs to an AND gate. The AND gate receives the signal delivered by the decoder as its input. This configuration allows for the selective activation or deactivation of a group of WLs. To facilitate the activation of multiple consecutive addresses in the array, an adapted WL decoder is utilized. The bit-line and source-line drivers were used to manage the analog input and output for the MVM operation. Also, a group-wise selection of WLs is performed concurrently and the intermediate result for MVM operation is accumulated into an accumulator block until all the WLs are selected for each layer. We utilized MUXes to select the different bit-lines that are sensed and converted by ADC. The shift-adder modules are used to shift and accumulate the partial sums coming from the array. Finally, a digital comparator and averaging block are used to implement the activation function. For the last layer, the average operation is performed with an averaging block.

For the mapping strategy ②, a similar architecture to strategy ① is employed. The key distinction relies upon the utilization of $K \times K$ crossbars in parallel to map the binary weights of a layer. Also, the dropout modules are connected to a similar WL index in each of the crossbar arrays, as shown in Fig. 4(b). Here, the same AND gate in the Dropout module receives signals from different decoders and the result is sent to each row of the $K \times K$ crossbars. For instance, the first WL of each crossbar of a layer connects the same Dropout module. All the WLs decoders are connected to a dropout block in gray in Fig. 4(b) comprising C_{in} dropout modules. It is worth mentioning that the dropout is used during the reading phase only, therefore, the dropout module is deactivated during the writing operation and WL decoders are used normally.

IV. Results

A. Simulation Setup

We evaluated the proposed MC-SpatialDropout on predictive performance using VGG, ResNet-18, and ResNet-20 topologies on the CIFAR-10 dataset. All the models were trained with SGD optimization algorithm, minimizing the proposed learning objective (4) with λ chosen between 1×10^{-5} and 1×10^{-7}, and the binarization algorithm from [12] was used. Also, all the models are trained with $\rho = 15\%$ dropout probability. The validation dataset of the CIFAR-10 is split 80:20 with 20% of the data used for the cross-validation and 80% used for evaluation.

To assess the effectiveness of our method in handling uncertainty, we generated six additional OOD datasets: 1) Gaussian noise ($\hat{\mathcal{D}}_1$): Each pixel of the image is generated by sampling random noise from a unit Gaussian distribution, $\mathbf{x} \sim \mathcal{N}(0, 1)$, 2) Uniform noise ($\hat{\mathcal{D}}_2$): Each pixel of the image is generated by sampling random noise from a uniform distribution, $\mathbf{x} \sim \mathcal{U}(0, 1)$, 3) CIFAR-10 with Gaussian noise ($\hat{\mathcal{D}}_3$): Each pixel of the CIFAR-10 images is corrupted with Gaussian noise, 4) CIFAR-10 with uniform noise ($\hat{\mathcal{D}}_4$): Each pixel of the CIFAR-10 images is corrupted with uniform noise, 5) SVHN: Google street view house numbers dataset, and 6) STL10: a dataset containing images from the popular ImageNet dataset. Each of these OOD datasets contains 8000 images, and the images have the same dimensions as the original CIFAR-10 dataset (32×32 pixels). During the evaluation phase, an input is classified as OOD or ID as follows:

$$\begin{cases} \text{OOD}, & \text{if } \max\left(\mathcal{Q}\left(\frac{1}{T}\sum_{t=1}^{T}\mathbf{y}_t\right)\right) < 0.9 \\ \text{ID}, & \text{otherwise.} \end{cases} \tag{7}$$

TABLE I
Predictive Performance of the proposed MC-SpatialDropout method in comparison with SOTA methods on CIFAR-10.

Topology	Method	Bit-width (W/A)	Bayesian	Inference Accuracy
ResNet-18	FP	32/32	No	93.0%
	RAD [14]	1/1	No	90.5%
	IR-Net [12]	1/1	No	91.5%
	SpinDrop [4], [5]	1/1	Yes	90.48%
	Proposed	1/1	Yes	**91.34%**
ResNet-20	FP	32/32	No	91.7%
	DoReFa [15]	1/1	No	79.3%
	DSQ [16]	1/1	No	84.1%
	IR-Net [12]	1/1	No	85.4%
	Proposed	1/1	Yes	**84.71%**
VGG	FP	32/32	No	91.7%
	LAB [17]	1/1	No	87.7%
	XNOR [18]	1/1	No	89.8%
	BNN [19]	1/1	No	89.9%
	RAD [14]	1/1	No	90.0%
	IR-Net [12]	1/1	No	90.4%
	SpinDrop [4], [5]	1/1	Yes	91.95%
	Proposed	1/1	Yes	**90.34%**

Here, $\mathbf{y}_t$ is the softmax output of the stochastic forward pass at MC run t with T MC runs, the function $\mathcal{Q}(\cdot)$ calculates the 10th percentile across a set of values, and the function $\max(\cdot)$ determines the maximum confidence score across output classes. Overall, OOD or ID is determined by whether the maximum value from the 10th percentile of the averaged outputs is less than 0.9 (for OOD) or not (for ID). The intuition behind our OOD detection is that the majority of confidence score of the T MC runs is expected to be high and close to one another (low variance) for ID data and vice versa for OOD data.

The hardware-level simulations for the proposed method were conducted on the Cadence Virtuoso simulator with 28nm-FDSOI STMicroelectronics technology library for the respective network topologies and dataset configurations.

B. Predictive Performance and Uncertainty Estimation

The predictive performance of the approach is close to the existing conventional BNNs, as shown in Table I. Furthermore, in comparison to Bayesian approaches [4], [5], our proposed approach is within 1% accuracy. Furthermore, the application of Spatial-SpinDrop before the convolutional layer and at the extracted feature maps can also achieve comparable performance ($\sim$ 0.2%), see Fig. 2. This demonstrates the capability of the proposed approach in achieving high predictive performance. However, note that applying Spatial-SpinDrop before all the convolutional layers can reduce the performance drastically, e.g., accuracy reduces to 75% on VGG. This is because at shallower layers, the number of OFMs is lower in comparison, leading to a high chance that most of the OFMs are being omitted (dropped). Also, as shown by [4], [5], BNNs are more sensitive to the dropout rate. Therefore, a lower Dropout probability between $10 - 20\%$ is suggested.

In terms of OOD detection, our proposed method can achieve up to 100% OOD detection rate across various model architectures and six different OOD datasets ($\hat{\mathcal{D}}_1$ through $\hat{\mathcal{D}}_6$), as depicted in Table II. There are some variations across different architectures and OOD datasets. However, even in these cases, our method can consistently achieve a high OOD detection rate, with the lowest detection rate being 64.39% on the ResNet-18 model with $\hat{\mathcal{D}}_4$ dataset and Spatial-SpinDrop applied to extracted feature maps. However, when the Spatial-SpinDrop is applied to the convolutional layers of the last residual block, OOD detection rate on $\hat{\mathcal{D}}_4$ dataset improved to 97.39%, a 33.00% improvement. Therefore, we suggest applying the Spatial-SpinDrop to the last convolutional layers to achieve a higher OOD detection rate at the cost of a small accuracy reduction. Consequently, the result suggests

TABLE II
EVALUATION OF THE PROPOSED MC-SPATIALDROPOUT METHOD IN DETECTING OOD.

Topologie	$\hat{\mathcal{D}}_1$	$\hat{\mathcal{D}}_2$	$\hat{\mathcal{D}}_3$	$\hat{\mathcal{D}}_4$	$\hat{\mathcal{D}}_5$	$\hat{\mathcal{D}}_6$
ResNet-18	99.56%	99.94%	96.1%	81.68%	83.02%	64.39%
ResNet-18[i]	100%	100%	100%	92.26%	99.98%	97.39%
ResNet-20	97.2%	100%	90.79%	87.94%	99.03%	99.81%
VGG	99.99%	100%	92.9%	78.91%	99.81%	100%

[i] Spatial-Dropout applied to final two convolutional layers.

TABLE III
LAYER-WISE OVERHEAD ANALYSIS OF THE PROPOSED METHOD IN COMPARISON TO SPINDROP [4], [5].

Layer-wise application of spatial Dropout					
Method	Mapping Strategy	# of Dropout Modules	Area	Power Consumption	Sampling Latency
SpinDrop	①	$K*K*C_{in}$	$79833.6\mu m^2$	$51.84mW$	$15ns$
	②	$K*K*C_{in}$	$79833.6\mu m^2$	$51.84mW$	$15ns$
Proposed	①	C_{in}	$8870.4\mu m^2$	$5.76mW$	$15ns$
	②	C_{in}	$8870.4\mu m^2$	$5.76mW$	$15ns$
Topology-wise application of spatial Dropout					
Method	Adaptive Avg. Pool	# of Dropout Modules	Area	Power Consumption	Sampling Latency
SpinDrop	Used	C_{out}	$17740.8\mu m^2$	$11.52mW$	$15ns$
	Not Used	$K*K*C_{out}$	$159667.2\mu m^2$	$103.68mW$	$15ns$
Proposed	Used	C_{out}	$17740.8\mu m^2$	$11.52mW$	$15ns$
	Not Used	C_{out}	$17740.8\mu m^2$	$11.52mW$	$15ns$

that the MC-SpatialDropout method is a robust and reliable approach to OOD detection across various model architectures and datasets.

C. Overhead Analysis

The proposed Spatial-SpinDrop modules were evaluated for the area, power consumption, and latency as shown in Table III and compared with the SpinDrop approach presented in [4], [5]. These evaluations were conducted using a crossbar array with dimensions of 64×32 and scaled for the VGG topology. In layer-wise application of spatial Dropout, the Dropout modules applied to convolutional layers of the last VGG block. Also, for topology-wise application of spatial Dropout, Dropout modules are applied to the extracted feature maps. In our evaluation, a configuration of $C_{in} = 256$, $K = 3$ and $C_{out} = 512$ is used.

At first, in terms of area, the SpinDrop method requires one dropout module per row in the crossbar structure, while our method only requires one dropout module per $K \times K$ group of rows. Therefore, the area and the power consumption of dropout modules are reduced by a factor of 9. In terms of latency for the dropout modules, we achieve $15ns$ in all cases. Indeed, to generate 1 bit, for a given number of rows, the dropout module needs to be written, however, such latency can be further decreased by increasing the writing voltages of the MTJ. Furthermore, in the case, the adaptive average pool layer is not used, the power consumption and the area for the SpinDrop approach increases greatly ($\times 9$). While in the proposed approach, the adaptative Average pool layer does not impact the total energy and area, as mentioned in Section III-C and shown in Table III.

Table IV compares the energy consumption of the proposed approach with the State-Of-The-Art implementation based on the MNIST dataset. For the evaluation, we used NVSIM, and we estimated the total energy for a LeNet-5 architecture to be consistent with the approach presented in [4]. When compared to the SpinDrop approach in [4] our approach is $2.94\times$ more energy efficient. Furthermore, when compared to RRAM technology, our solution is $13.67\times$ more efficient. Finally, in a comparison with classic FPGA implementation, the proposed approach achieves substantial energy savings of up to $94.11\times$.

TABLE IV
ENERGY EFFICIENCY COMPARISON OF HARDWARE IMPLEMENTATIONS

Related works	Technology	Bit resolution	Energy
R.Cai et al. [20]	FPGA	8-bit	18.97 µJ/Image
X.Jia et al. [21]	FPGA	8-bit	46.00 µJ/Image
H.Awano et al. [22]	FPGA	7-bit	21.09 µJ/Image
A. Malhotra et al. [23]	RRAM	4-bit	9.30 µJ/Image
S.T.Ahmed et al. [4]	STT-MRAM	1-bit	2.00 µJ/Image
Proposed implementation	**STT-MRAM**	**1-bit**	**0.68** µJ/Image

V. CONCLUSION

In this paper, we present MC-SpatialDropout, an efficient spatial dropout-based approximation for Bayesian neural networks. The proposed method exploits the probabilistic nature of spintronic technology to enable Bayesian inference. Implemented on a spintronic-based Computation-in-Memory fabric with STT-MRAM, MC-SpatialDropout achieves improved computational efficiency and power consumption.

ACKNOWLEDGMENTS

This work was supported by a joint ANR-DFG grant Neuspin Project ANR-21-FAI1-0008.

REFERENCES

[1] D. Kiela *et al.*, "Dynabench: Rethinking benchmarking in nlp," *arXiv preprint arXiv:2104.14337*, 2021.
[2] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.
[3] Z. Wang *et al.*, "Resistive switching materials for information processing," *Nature Reviews Materials*, vol. 5, no. 3, pp. 173–195, 2020.
[4] S. T. Ahmed *et al.*, "Binary bayesian neural networks for efficient uncertainty estimation leveraging inherent stochasticity of spintronic devices," in *IEEE/ACM NANOARCH*, 2022.
[5] ——, "Spindrop: Dropout-based bayesian binary neural networks with spintronic implementation," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 150–164, 2023.
[6] T. Y. Lee *et al.*, "World-most energy-efficient MRAM technology for non-volatile RAM applications," in *2022 International Electron Devices Meeting (IEDM)*, Dec. 2022, pp. 10.7.1–10.7.4, iSSN: 2156-017X.
[7] N. Srivastava *et al.*, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[8] C. Blundell *et al.*, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.
[9] B. Lakshminarayanan *et al.*, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.
[10] T. Gokmen *et al.*, "Training deep convolutional neural networks with resistive cross-point devices," *Frontiers in neuroscience*, vol. 11, p. 538, 2017.
[11] X. Peng *et al.*, "Optimizing weight mapping and data flow for convolutional neural networks on rram based processing-in-memory architecture," in *IEEE ISCAS*. IEEE, 2019, pp. 1–5.
[12] H. Qin *et al.*, "Forward and backward information retention for accurate binary neural networks," in *IEEE CVPR*, 2020.
[13] B. Razavi, "The strongarm latch [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 7, no. 2, pp. 12–17, 2015.
[14] R. Ding *et al.*, "Regularizing activation distribution for training binarized deep networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 408–11 417.
[15] S. Zhou *et al.*, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.
[16] R. Gong *et al.*, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the ICCV*, 2019, pp. 4852–4861.
[17] L. Hou *et al.*, "Loss-aware binarization of deep networks," in *International Conference on Learning Representations (ICLR)*, 2017.
[18] M. Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV 2016*. Springer, 2016, pp. 525–542.
[19] I. Hubara *et al.*, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.
[20] R. Cai *et al.*, "Vibnn: Hardware acceleration of bayesian neural networks," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 476–488, 2018.
[21] X. Jia *et al.*, "Efficient Computation Reduction in Bayesian Neural Networks Through Feature Decomposition and Memorization," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 32, Apr. 2021.
[22] H. Awano *et al.*, "Bynqnet: Bayesian neural network with quadratic activations for sampling-free uncertainty estimation on fpga," in *Proceeding of DATE*. IEEE, 2020, pp. 1402–1407.

[23] A. Malhotra *et al.*, "Exploiting oxide based resistive ram variability for bayesian neural network hardware design," *IEEE Trans. on Nano.*, vol. 19, pp. 328–331, 2020.