Scalable Spintronics-based Bayesian Neural Network for Uncertainty Estimation

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*—Typical neural networks are incapable of effectively estimating prediction uncertainty, leading to overconfident predictions. Estimating uncertainty is crucial for safety-critical tasks such as autonomous vehicle driving and medical diagnosis and treatment. Bayesian Neural Networks (BayNNs), which combine the capabilities of neural networks and Bayesian inference, are an effective approach for uncertainty estimation. However, BayNNs are computationally demanding and necessitate substantial memory resources. Computation-in-memory (CiM) architectures utilizing emerging resistive non-volatile memories such as Spin-Orbit Torque (SOT) have been proposed to increase the resource efficiency of traditional neural networks. However, training scalable and efficient BayNNs and implementing them in the CiM architecture presents its own challenges. In this paper, we propose a scalable Bayesian NN framework via *Subset-Parameter inference* and its Spintronic-based CiM implementation. Our method is evaluated on large datasets and topologies to show that it can achieve comparable accuracy while still being able to estimate uncertainty efficiently at up to $70\times$ lower power consumption and $158.7\times$ lower storage memory requirements.**

I. Introduction

Deep learning is a subfield of artificial intelligence that uses several types of neural networks (NNs) to learn from data, identify patterns, and make predictions with high accuracy [1]. In numerous disciplines, including computer vision, natural language processing, and speech-to-text applications, deep learning algorithms can attain cutting-edge performance. Consequently, they are readily deployed in diverse environments and applications, including safety-critical applications where incorrect prediction could lead to harm to humans or other critical failures.

Conventional neural networks can only provide output prediction and lack the ability to accurately convey prediction uncertainty due to their deterministic parameters and neuron activations, resulting in overconfident predictions [2]. Factoring uncertainty in prediction allows the user to make more informed decisions for a task. Uncertainty estimation is important in a variety of situations, such as in safety-critical tasks, e.g., autonomous vehicles, medical diagnosis and treatment, industrial robotics, and financial systems. It allows accurate assessment of the risk associated with a given prediction, which is important for a trustworthy deep learning system.

In the literature, there are several methods for uncertainty estimation, such as Bayesian Neural Networks (BayNNs). BayNNs are probabilistic models that combine the power of neural networks with Bayesian inference. The model assigns probability distributions over its parameters, called prior distributions, to learn posterior distribution during training as data points are observed. The uncertainty of the model's predictions can be obtained from the posterior predictive distribution.

Despite the capabilities of NNs, they usually need a huge amount of parameters, and also they are traditionally implemented in an inefficient von Neumann architecture. Consequently, memory and computation demands are high, making it difficult to meet the high-performance requirement of real-time applications or deploy them to edge devices with limited resources such as microcontrollers and smartphones. The resource demand is exacerbated for BayNNs as they naturally require more computational and memory resources compared to traditional neural networks.

To reduce the computational demands of conventional NNs, hardware and algorithmic approaches have been proposed [3]–[6]. From a hardware perspective, several specialized hardware accelerators are proposed to accelerate the most common operations in NNs, Matrix-Vector Multiplications (MVM) [7]. In particular, analog Computation in-Memory (CiM) architectures with emerging resistive non-volatile memories are promising solutions [8]. CiM architectures enable MVM operations to be executed within the memory arrays where the model parameters already reside, hence reducing the need for costly data movement. Also, compared to other NVM technology, Magnetic Random Access Memory (MRAM) stands out the most thanks to its nanosecond latency, high endurance (10^{12} cycles), and low switching energy (10 fJ) [8]. Therefore, implementing BayNNs in an MRAM-based CiM architecture leads to a highly efficient solution that combines the benefits of all in a single package.

However, there are several challenges associated with the BayNNs and their CiM implementation. Conventional BayNNs are 32-bit full-precision, but NVM has limited stable conductance states that can represent the parameters of a model. Also, the true posterior of BayNNs is computationally intractable and must be approximated. Common approximation methods such as variational inference, and ensembles are computationally intensive and require significant computing resources, see Section IV-C, making them difficult to use in real-time applications. Also, during inference time, prediction

978-3-9819263-7-8/DATE23/© 2023 EDAA

requires drawing samples from the posterior. Implementing posterior distributions with an NVM-based crossbar array may not be feasible or require crossbar structure changes. Consequently, a direct implementation of conventional BayNNs to CiM architecture is not feasible.

In this paper, we propose a scalable Bayesian neural network framework based on *subset parameter inference*. In our approach, only a small subset of overall parameters are defined as probabilistic parameters and the remainder, i.e., synaptic weights remain deterministic. The subset parameter is strategically selected so that the cost associated with Bayesian inference becomes negligible. That is to say, they can be implemented in CiM design without requiring modification to the common crossbar structure. In addition, we propose a design-space exploration that enables low-cost and fast sampling from the posterior distribution for Bayesian inference. Furthermore, we constructed a comprehensive flow from the training algorithm to CiM hardware implementation for our Bayesian approach. Consequently, our proposed *subset parameter inference* is capable of overcoming the challenges associated with BayNN and CiM implementations, while yet achieving equivalent performance and can estimate uncertainty for out-of-distribution data. Our approach is evaluated on large benchmark datasets and topologies.

The remainder of the paper is structured as follows: Section II outlines the background of our work. The algorithmic and hardware specifics for subset-parameter-based Bayesian Neural Networks are introduced in Section III. Section IV then demonstrates the evaluation of our proposed method, and Section V summarizes the paper.

II. Background

A. Bayesian NNs

Bayesian NNs consider the parameter vector $\boldsymbol{\theta}$ as a random variable (instead of a fixed vector) with a distribution $p(\boldsymbol{\theta})$. In the Bayesian sense, learning then relates to estimating the posterior distribution $\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \mathcal{D})$ given the data $\mathcal{D}$. With the help of $p(\boldsymbol{\theta} \mid \mathcal{D})$, the posterior distribution for $\mathbf{y}^*$ (given a test point $\mathbf{x}^*$) can be obtained through

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \mathcal{D}) = \int p(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathcal{D})\, \mathrm{d}\boldsymbol{\theta}.$$

Hence, the Bayesian approach allows a richer representation of the uncertainty of the prediction, as it takes the uncertainty in the estimation of $\boldsymbol{\theta}$ into account.

Unfortunately, training Bayesian NNs, that is, inferring $p(\boldsymbol{\theta} \mid \mathcal{D})$, is not as straightforward as maximizing a likelihood (conventioan NNs). Additionally, $p(\boldsymbol{\theta} \mid \mathcal{D})$ usually cannot be obtained in closed form, but can only be approximated. Common approximation techniques are with simpler distributions via the Laplace approximation or variational inference. For variational inference, a so-called variational distribution $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ with parameters $\boldsymbol{\omega}$ is chosen to substitute the intractable $p(\boldsymbol{\theta} \mid \mathcal{D})$. The parameters $\boldsymbol{\omega}$ of $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ are thereby chosen to reduce the mismatch between $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta} \mid \mathcal{D})$ via minimizing the Kullback-Leibler divergence $\mathrm{KL}\left(q_{\boldsymbol{\omega}}(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta} \mid \mathcal{D})\right)$ with respect to $\boldsymbol{\omega}$, see [9]. The specific parametric form of $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ is usually taken to be a Gaussian distribution with a diagonal covariance matrix for its computational convenience, and $\boldsymbol{\omega} = \{\boldsymbol{\mu}_\omega, \boldsymbol{\sigma}_\omega\}$ thus relates to the mean and diagonal (variances) of the covariance matrix.

When a variational distribution $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ was found this way, the distribution of $\mathbf{y}^*$ for given $\mathbf{x}^*$ can be approximated by Monte Carlo estimation using T samples, as

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \mathcal{D}) \approx \frac{1}{T} \sum_{t=1}^{T} p\left(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta}^{(t)}\right) \quad \text{with} \quad \boldsymbol{\theta}^{(t)} \sim q_{\boldsymbol{\omega}}(\boldsymbol{\theta}). \tag{1}$$

Note that many of the above concepts cannot be directly mapped to CIM-based hardware architectures. Specifically, values of parameters such as $\boldsymbol{\theta}$ have to be considered quantized, and only specific choices of $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$, from which $\boldsymbol{\theta}$ can be sampled, can efficiently be realized. The following thus proposes a CiM-based architecture-friendly realization.

B. Quantization Methods

Typically, NNs have 32-bit full-precision weights and activations. Quantization is a model compression technique that reduces the bit-width of weights and activation to as low as binary. It allows for deploying NNs on resource-constrained devices and for improving the performance of real-time applications. Binarized NN (BNN) replaces MVM operation with XNOR and bit-count operation [6]. Consequently, the model can be compressed by up to $32\times$ and the computational requirement can be reduced by up to $32\times$. Also, since MRAM has limited stable conductance states that can represent the parameters of a model, quantizing model parameters allows direct implementation of NNs to a CiM architecture. Binarizing a Bayesian NN can lead to an extremely efficient solution that can offset its inherent high cost. As a result, in this paper, we apply the Bayesian paradigm with variational approximation to a BNN. However, applying variational distribution to binary ($+1$, -1) weights is challenging and can be costly. To circumvent this, we propose a *subset parameter inference*.

C. Uncertainty in Deep Learning

NNs can perform well in ideal settings but tend to be overconfident when faced with uncertain situations, such as out-of-distribution (OOD) data. OOD data refers to the input data that is distinct from the distribution of the training data and does not belong to any of the classes observed during training. A trustworthy deep learning system should provide reliable uncertainty estimates when faced with OOD data. Ideally, the NN should "reject" or refrain from making a prediction when uncertain, allowing for human intervention in the decision-making process, referred to as "human-in-the-loop."

D. Spintronics Device

MRAM has gained significant attention for its potential use in CiM architectures [10]. The Magnetic Tunnel Junction

(MTJ) is the basic compound of MRAM devices, consisting of two ferromagnetic layers separated by a thin oxide layer. One of the layers has a pinned magnetization orientation, while the other one, called the free layer, can have its magnetization reversed. The freely magnetized layer can be reversed mainly through two writing mechanisms: Spin Transfer Torque (STT) and Spin-Orbit Torque (SOT). When the magnetizations of the two ferromagnetic layers are aligned in the same direction, the MTJ is in a low resistance state, called the parallel state (R_P). On the other hand, when the ferromagnetic layers have opposite magnetization, the MTJ is in the anti-parallel (R_{AP}) state and exhibits a high resistance value.

STT-MRAM has common read and write paths and the resistance value is determined by the direction of the current passing through the device. When the reading current is increased, the stability of the STT-MRAM is degraded, which leads to a higher error rate. The SOT-MRAM separates read and write paths, and thus the reading reliability can be improved significantly. This three-terminals device is made up of an MTJ placed on a heavy metal layer.

In the traditional crossbar design, analog computation is achieved through a parallel reading of all the bit cells. The separated read and write paths of the SOT-MRAM allow a higher resistance without impacting the reliability of the device and increase switching speed without affecting the endurance. Recently, a new technique was proposed to stack multiple MTJs on the same substrate to emulate a multi-value cell [11]. This new method is based on the combination of voltage control magnetic anisotropy effect (VCMA) and SOT effect, also known as Voltage-Gated SOT (VG-SOT). The VCMA effect utilizes an electric field to change the anisotropy at the interface of the ferromagnet and the oxide barrier layer, this effect is controlled by the voltage across the MTJ, which decreases the switching barrier and eases the writing at a given SOT current. VG-SOT allows switching the device at lower currents while maintaining the subnanosecond switching of SOT-MRAM. Furthermore, VG-SOT can also allow a multi-pillar configuration of SOT MRAM, where the VCMA effect can act as an MTJ selector [11].

E. Related Works

1) Uncertainty Estimation: In the literature, several stochastic and Bayesian approaches for uncertainty estimation in deep learning exist. Dropout-based approaches, e.g., MC-Dropout [12], and MC-DropConncet [13] are among the most prevalent techniques for estimating uncertainty. Those approaches randomly drop either neurons or weights and use weight decay regularization. However, modern NN topologies do not employ dropout as a regularization method, instead, various normalization methods, e.g., batch normalization is more commonly used. Also, the Dropout rates require careful calibration, otherwise, the accuracy degrades. Therefore, work in [14] proposed MC-Batchnorm, also has some drawbacks. This method involves passing a random subset of the training data through the neural network and recalculating the batch statistics. The disadvantage of this strategy is that the training datasets must be stored in the hardware. In addition, processing each mini-batch necessitates a significant amount of matrix-vector multiplication operations.

2) Hardware resources: In terms of material implementation, many studies have proposed the deployment of Bayesian networks on dedicated architectures. The work in [15], proposed an FPGA implementation with a novel activation function. The proposed implementation is able to achieve a good uncertainty estimate with only one forward pass through a NN. Such an approach may be effective for small NN, but not be suitable for larger models. In [16], the authors proposed a CiM implementation where the variance parameter is stored in a crossbar and the probability distribution sampling is done using several stochastic resistive (RRAM) devices. Despite the low power consumption of the RRAM devices, their implementation may suffer from inaccuracies due to inconsistent mean and variance estimation. Work in [17], [18] suggested using different crossbar arrays to represent the mean and variance with MRAM technology. However, the representation of variance and mean parameters using crossbars requires a considerable amount of pre-processing to correctly map and quantize the mean and variance into the array. In [5], [19], the concept of MC-dropout is exploited for Bayesian inference using STT-MRAM devices.

Our study aims at overcoming the aforementioned issues by implementing a stochastic quantized scale parameter with a binary deterministic weight in crossbar structures. To this end, we combine the utilization of the inherent stochasticity and the deterministic behavior of SOT-MRAM devices.

III. Bayesian Neural Network Via Sub-Parameter Inference

A. Problem Definition

In a NN, the weight matrices of convolutional and fully connected layers consume the most storage memory, as they are utilized to calculate the dot product between the input and weights. For example, in ResNet-18 topology, weight matrices consume 80.90% of total parameters while biases 16.61% and other parameters consume 2.39%. In Bayesian NNs, since the approximate variational distribution $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ with parameters $\boldsymbol{\omega}$, is applied to the weight matrices, the memory consumption increases significantly.

On the other hand, higher computational complexity comes from the need for sampling from $q_{\boldsymbol{\omega}}(\boldsymbol{\theta})$ during Bayesian inference. Also, the implementation of variational distribution $p(\mathbf{y}^* \mid \mathbf{x}^*, \mathcal{D})$ with the CiM-based hardware accelerators can be challenging and may require changes to the normal memory structure. Furthermore, due to the limited stable states of MRAM devices, quantization of mean and variance is needed. Consequently, hardware implementations in a conventional way can differ considerably from the trained model.

B. Bayesian Subset Parameter Inference

In this paper, we propose an efficient Bayesian NN framework with both deterministic and stochastic parameters. Specifically, the larger parameter sub-group, e.g., weights and

biases of linear and convolutional layers are deterministic, while a specific small parameter group, the *scale* parameter, is considered a random variable that follows a probability distribution. As a result, the memory and computational complexity are drastically reduced compared to conventional BayNNs and adaptation is CiM-hardware friendly.

For our approach, we consider the following approximation of the signature weighted sum computation of an input vector $\mathbf{x}$ with a weight matrix $\mathbf{W}$ as

$$\mathbf{x}\mathbf{W} \approx \text{sign}(\mathbf{x})\,\text{sign}(\mathbf{W}) \odot \mathbf{s}, \tag{2}$$

where $\mathbf{s}$ is a vector of learnable parameters representing *scale* and $\text{sign}(\cdot)$ is to be considered elementwise. Since we are considering Bayesian Binary Neural Networks, the activations $\mathbf{x}$ and weights $\mathbf{W}$ are binarized $\{+1, -1\}$ with sign function. In contrast, the entries of the scaling vector $\mathbf{s}$ are typically considered 32-bit (float) values, but we apply further approximation for CiM implementation.

For learning, distinct treatment is applied to the two-parameter groups. Specifically, we apply a Bayesian treatment to learning $\mathbf{s}$ via variational inference and learn a distribution $q_{\boldsymbol{\omega}}(\mathbf{s})$, while the rest of the parameters, which we denote by $\boldsymbol{\theta}$ in the following (e.g., the weights $\mathbf{W}$ for each layer), are learned via a (classical) maximum likelihood approach. The overall training objective is defined as

$$\max_{\boldsymbol{\theta},\boldsymbol{\omega}} \quad p(\mathcal{D} \mid \boldsymbol{\theta}) - \lambda \cdot \text{KL}\left(q_{\boldsymbol{\omega}}(\mathbf{s}) \,\|\, p(\mathbf{s} \mid \mathcal{D})\right), \tag{3}$$

where λ denotes a hyper-parameter that is to be set. Note that this objective cannot be directly optimized since the KL term is intractable and therefore replaced with the evidence lower bound (ELBO) approximation, which provides a lower bound on the KL [20].

Due to the hardware constraints, we consider several approximations. Specifically, the parameters $\boldsymbol{\theta}$ need to be binarized, while samples from $q_{\boldsymbol{\omega}}(\mathbf{s})$ are also quantized. To enable gradient-based learning, quantizations are only considered in forward passes (during training and inference), whereas they are disregarded while computing gradients. This is generally known as the straight-through (gradient) estimator [6].

To efficiently implement the Bayesian NN in a CiM-architecture, we take a set $\mathcal{S} = \{\mathbf{s}^{(n)} \sim q_{\boldsymbol{\omega}}(\mathbf{s}) \mid n = 1, \cdots, N\}$ of N samples from $q_{\boldsymbol{\omega}}(\mathbf{s})$. These samples are then mapped to a specific crossbar array. In operation, a stochastic sampler is used to sample one of the crossbars in each forward pass. Consequently, the distribution $q(\mathbf{s}) = \text{Choose}(\mathcal{S})$ that selects samples from $\mathcal{S}$ uniformly, approximates the distribution of the (quantized) samples from the variational distribution $q_{\boldsymbol{\omega}}(\mathbf{s})$.

Hence, through the CiM-hardware, the distribution of $\mathbf{y}^*$ given $\mathbf{x}^*$ is approximated as

$$p(\mathbf{y}^* \mid \mathbf{x}^*, \mathcal{D}) \approx \frac{1}{T}\sum_t p\left(\mathbf{y}^* \mid \mathbf{x}^*, \boldsymbol{\theta}, \mathbf{s}^{(t)}\right) \quad \text{with} \quad \mathbf{s}^{(t)} \sim q(\mathbf{s}) = \text{Choose}(\mathcal{S}). \tag{4}$$

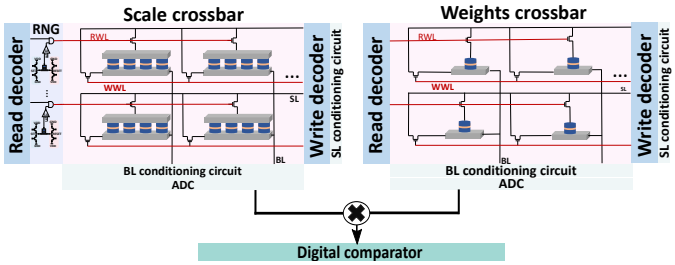
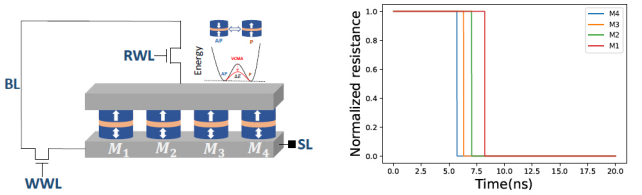


Fig. 1: Proposed spintronic architecture.



(a) Multi-value cells with 4 MTJs

(b) Multi-value cell switching

Fig. 2: Proposed multi-value SOT bit-cell.

Note that the parameters $\boldsymbol{\theta}$ (e.g., weights) are considered deterministic, while the learned distribution of the scales $\mathbf{s}$ is used to express the uncertainty in the predictions.

C. Hardware implementation

1) Bayesian Inference Architecture: To implement the proposed Bayesian inference, a new CiM architecture is proposed according to the functionality of equation (2). The architecture consists of two crossbars per layer, one maps the Bayesian scale and the other the weights. The scale crossbar is implemented using a multi-level device that stores the quantized parameters. Also, a stochastic spintronic device is used to allow the random selection of the different devices in each forward pass based on equation (4). Each crossbar is equipped with two decoders, one for reading and the other for writing. Decoders allow for the selection of multiple devices for reading and writing operations. The stochastic sampler is connected to the different signals of the reading decoder of the scale crossbar. The MVM operation in a crossbar array is achieved by activating multiple wordlines in parallel. At the output of the crossbar, the current is sensed and converted to a digital signal with a flash analog-to-digital converter (ADC). The results of the scale and weights crossbar are multiplied layer-wise, and we apply a sign function with comparators, as depicted in Figure 1. The model prediction are accumulated for T Monte-Carlo runs with the Adder ACcumulator (AAC).

2) Scale crossbar: To quantitatively represent the parameters required for our proposed BayNN approach, we have implemented a multi-level device composed of multiple MTJs placed on a single SOT track. To allow for reliable reading and writing of the cell, four MTJs were used. Each multi-level device is able to store up to five levels of conductance (eg: 4AP, 3AP-1P, 2AP-2P, 1AP-3P, 4P). To achieve a wider range of conductance levels and parameter representation, several of these multi-value cells can be used in tandem. As mentioned

earlier, this device exploits both the VCMA and SOT effects for the writing process. The parallel MTJs share the same top and bottom electrodes. Thus, only two access transistors are needed for read and write operations, as in the conventional SOT device, see Figure 2. Consequently, the cell becomes denser, and hardware resource efficiency improves. The access transistors are controlled by a Write Word-Line (WWL) and a Read Word-Line (RWL). At a given SOT current, the RWL signal can vary to program individual MTJs or multiple MTJs at a time.

3) Weights crossbar: For the crossbar that encodes weights, we decided to implement binary synapses with SOT technology, where the resistance state of the device (R_P and R_{AP}) represents the binary value ($+1$, -1). The current sum due to the activation of multiple cells will be compared to a reference. The reference serves as an activation function for the deterministic crossbar array.

4) Stochastic SOT device: By utilizing the stochastic behavior of SOT devices as a random number generator (RNG), we were able to introduce the desired random sampling. Two CMOS drivers were used to generate the bidirectional current across the heavy metal to switch between R_P and R_{AP} states. To attain a given probability with the stochastic device, "SET" and "RESET" operations were repeated to program the MTJ. After a "SET" operation, the MTJ is sensed to evaluate its state, and then "RESET". As a result, the successive "SET" and "RESET" operations generate a stochastic bitstream.

IV. Experimental Result

A. Simulation Setup

The predictive performance of the proposed method is evaluated on MNIST, Fashion-MINST, and CIFAR-10 benchmark in-distribution datasets on MLP, LeNet, and VGG based on [13] topologies. Weights and activation BayNNs are binarized according to [3] and the proposed Bayesian scale is quantized to 4-bit using the algorithm proposed in [4]. A value of 0.001 is used for the hyperparameter λ in Equation 3.

B. Algorithmic Evaluation

1) Predictive Performance: For MLP on MNIST, our results depicted in Table I show that the proposed subset parameter inference performs comparably to State-Of-The-Art (SOTA) full-precision Bayesian methods with only a 0.51% difference in accuracy. Similarly, inference accuracy is comparable, i.e., within 0.68% of full-precision and binary point estimate NNs. Generally, point estimate methods outperform Bayesian methods, therefore, the difference is slightly greater in comparison.

Furthermore, the inference accuracy of the proposed method on CNN topologies on Fashion-MNIST and CIFAR-10 is still comparable to SOTA Bayesian methods as depicted in Tables II and III. Specifically, the inference accuracy of Fashion-MNIST is 0.01% and CIFAR-10 is 2.54% lower in the worst case. Since CIFAR-10 is a much harder dataset, the difference is larger for our proposed method. However, when our proposed method is compared with SOTA binary method, our method slightly can improve the accuracy, e.g., by 0.62% for VGG. Due to the fact that binary activation is an approximation of Tanh activation, the performance of our proposed method is closer to a full-precision model with Tanh activation but with ReLU activation the difference is slightly greater.

TABLE I: Predictive Performance of MNIST dataset on four-layer MLP in comparison to related Bayesian and point estimate methods. The superscript * represents methods that are point estimates.

Method	Bit-width (W/A)	Inference Accuracy
FP (ReLU)*	32/32	98.78%
FP (Tanh)*	32/32	98.39%
MC-Dropout	32/32	98.61%
IR-Net*	1/1	98.26%
Proposed	1/1	**98.10%**

TABLE II: Illustration of the proposed method's prediction performance on the Fashion-MNIST dataset using the LeNet-5 CNN topology, compared to Bayesian and point estimate approaches with varied bit-widths of weights and activation (W/A). The superscript * denotes point estimates methods.

Method	Bit-width (W/A)	Inference Accuracy
FP (ReLU)*	32/32	92.01%
FP (Tanh)*	32/32	91.78%
MC-Dropout	32/32	91.71%
Deep Ensemble	32/32	91.68%
IR-Net*	1/1	91.71%
Proposed	1/1	**92.0%**

2) Uncertainty Estimation: Typically, we assume that the distributions of training and test data are identical. However, when the distributions of training and test data differ, e.g., when the test data is rotated or corrupted with noise, we expect the uncertainty of the model or of the prediction to be high. We have performed two experiments with varying intensity for dataset shift. In one case, we continuously rotated the image by 7 degrees in 12 steps, and in the other case, we added random uniform noise with increasing intensity. It can be seen in Figure 3 that the inference accuracy decrease and the negative log-likelihood (NLL) increase. NLL is a standard method for estimating uncertainty, and a well-trained model typically has a low NLL score. A higher NLL score than a predefined threshold, e.g., mean NLL on test data, indicates OOD data. We can detect up to 64.34% of OOD data with this approach.

C. Analysis of Hardware Implementation

The energy consumption of the proposed architecture is presented in Table IV and is compared with the related works. The multiplier, the AAC block, and comparators were synthesized with Synopsys Design Compiler using the TSMC 40nm PDK. The crossbar and the stochastic device were then simulated on Spice. Additionally, the design was scaled up using the CiM version of NVSIM, and we evaluated the energy consumption on a LeNet-5 and small VGG topology. For each topology, we performed T=10 forward passes on the MNIST dataset. Energy

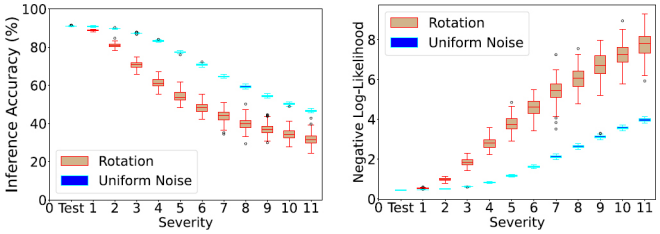


Fig. 3: Evaluation of out-of-distribution performance on Fashion-MNIST dataset. The images are rotated, and uniform noise is added to shift the distribution.

TABLE III: Prediction performance of our method is compared to Bayesian and point estimate approaches utilizing the CIFAR-10 dataset and different bit-widths of weights and activation (W/A). * denotes point estimation methods.

Topology	Method	Bit-width (W/A)	Inference Accuracy
VGG	FP (Tanh)*	32/32	91.23%
	FP (ReLU)*	32/32	93.31%
	MC-Dropout	32/32	92.79%
	IR-Net*	1/1	89.96%
	Proposed	**1/1**	**90.62%**
ResNet-18	FP (Tanh)*	32/32	91.33%
	FP (ReLU)*	32/32	93.77%
	MC-Dropout	32/32	93.44%
	IR-Net*	1/1	91.5%
	Proposed	**1/1**	**90.5%**

consumption of $0.30\,\mu J$ is reported with LeNet-5 topology and $2.00\,\mu J$ with the small VGG topology. The proposed architecture is $70\times$ more energy efficient when compared with FPGA implementation [15], $31\times$ better when compared to RRAM [16], and $2.63\times$ better when compared to an MTJ-based crossbar [17]. All studies were evaluated on the MNIST dataset but on different topologies, LeNet-5 in our case, while other studies were only evaluated on two linear layers.

In terms of memory consumption, compared to SOTA methods for uncertainty estimation, our proposed method requires $63.49\times$ lower storage memory compared to variation inference approximation [9], $158.78\times$ lower storage memory compared to the ensemble approach (with 5 ensembles) [21], and $31.76\times$ lower storage memory compared to Dropout-based [12] approximation. Furthermore, even compared to 1-bit binary NNs with point estimate parameters, our proposed BayNN requires is $\sim 1\%$ lower storage. Since we quantize the scale, our method is even lower than SOTA binary NNs which have 32-bit scales. Furthermore, We assumed one bit-cell is required for each bit of parameter storage. Variables of the model are not taken into account.

V. Conclusion

In this paper, we present a low-cost and scalable Bayesian neural network framework suitable for CiM hardware. Our method deals with larger groups of parameters in a deterministic method and Bayesian processing is only applied to a specific group of parameters, *scale*. A novel CiM architecture with two separate crossbars per layer is presented for the Bayesian inference. One crossbar stores deterministic weights, while the second array stores the Bayesian scale. A multilevel SOT-based bitcell is designed to map quantized Bayesian scale parameters. Furthermore, the stochastic behavior of the MTJ is harnessed to implement sampling from the posterior distribution of the variational distribution. Our proposed Bayesian NN is rigorously examined for its prediction performance and uncertainty quantification. We show that the prediction performance is comparable to SOTA methods with different bit-widths. Furthermore, the energy consumption and memory requirement were evaluated on large topologies. Compared to SOTA Bayesian implementation, the energy consumption is $70\times$ smaller than CMOS-based implementation and $31\times$ smaller than RRAM-based implementation. Storage Memory requirement is up to $158.78\times$ lower.

TABLE IV: Energy comparison with SOTA implementation

Method	Implementation	Energy (µJ/Image)
H.Awano et al. [15]	FPGA	21.09
A. Malhotra [16]	RRAM	9.30
K.Yang et al. [17]	Domain wall-MTJ	0.79
Proposed implementation	**SOT-MRAM**	**0.30**

References

[1] I. Goodfellow *et al.*, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *NeurIPS*, vol. 30, 2017.

[3] H. Qin *et al.*, "Forward and backward information retention for accurate binary neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2250–2259.

[4] J. Choi *et al.*, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[5] S. T. Ahmed *et al.*, "SpinDrop: Dropout-Based Bayesian Binary Neural Networks with spintronic Implementation," *To appear at IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2023.

[6] I. Hubara *et al.*, "Binarized neural networks," *NeurIPS*, vol. 29, 2016.

[7] A. Reuther *et al.*, "Ai accelerator survey and trends," in *IEEE HPEC*, 2021.

[8] B. Dieny *et al.*, "Opportunities and challenges for spintronics in the microelectronics industry," *Nature Electronics*, vol. 3, no. 8, Aug. 2020.

[9] C. Blundell *et al.*, "Weight uncertainty in neural network," in *ICML*. PMLR, 2015.

[10] S. Jung *et al.*, "A crossbar array of magnetoresistive memory devices for in-memory computing," *Nature*, vol. 601, no. 7892, 2022.

[11] J. Doevenspeck *et al.*, "Multi-pillar SOT-MRAM for Accurate Analog in-Memory DNN Inference," in *2021 Symposium on VLSI Technology*, 2021.

[12] Y. Gal *et al.*, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." PMLR, 2016.

[13] A. Mobiny *et al.*, "Dropconnect is effective in modeling uncertainty of bayesian deep networks," *Scientific reports*, 2021.

[14] M. Teye *et al.*, "Bayesian uncertainty estimation for batch normalized deep networks," in *ICML*. PMLR, 2018.

[15] H. Awano *et al.*, "BYNQNet: Bayesian Neural Network with Quadratic Activations for Sampling-Free Uncertainty Estimation on FPGA," in *DATE*, 2020.

[16] A. Malhotra *et al.*, "Exploiting Oxide Based Resistive RAM Variability for Bayesian Neural Network Hardware Design," *IEEE TNANO*, 2020.

[17] K. Yang *et al.*, "All-Spin Bayesian Neural Networks," *IEEE T-ED*, 2020.

[18] A. Lu *et al.*, "An Algorithm-Hardware Co-Design for Bayesian Neural Network Utilizing SOT-MRAM's Inherent Stochasticity," *IEEE-JXCDC*, 2022.

[19] S. T. Ahmed *et al.*, "Binary bayesian neural networks for efficient uncertainty estimation leveraging inherent stochasticity of spintronic devices," in *IEEE/ACM NANOARCH*, 2022.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] B. Lakshminarayanan *et al.*, "Simple and scalable predictive uncertainty estimation using deep ensembles," *NeurIPS*, 2017.