frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 10 August 2022
DOI 10.3389/fnins.2022.937782

OPEN ACCESS

EDITED BY
Gianvito Urgese,
Politecnico di Torino, Italy

REVIEWED BY
Vittorio Fra,
Polytechnic University of Turin, Italy
Mantas Mikaitis,
The University of Manchester,
United Kingdom

*CORRESPONDENCE
Rachmad Vidya Wicaksana Putra
rachmad.putra@tuwien.ac.at

SPECIALTY SECTION
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

RECEIVED 06 May 2022
ACCEPTED 11 July 2022
PUBLISHED 10 August 2022

CITATION
Putra RVW, Hanif MA and Shafique M (2022) EnforceSNN: Enabling resilient and energy-efficient spiking neural network inference considering approximate DRAMs for embedded systems. *Front. Neurosci.* 16:937782. doi: 10.3389/fnins.2022.937782

EnforceSNN: Enabling resilient and energy-efficient spiking neural network inference considering approximate DRAMs for embedded systems

Rachmad Vidya Wicaksana Putra[1]*, Muhammad Abdullah Hanif[2] and Muhammad Shafique[2]

[1]Embedded Computing Systems, Institute of Computer Engineering, Technische Universität Wien, Vienna, Austria, [2]eBrain Lab, Division of Engineering, New York University Abu Dhabi (NYUAD), Abu Dhabi, United Arab Emirates

Spiking Neural Networks (SNNs) have shown capabilities of achieving high accuracy under unsupervised settings and low operational power/energy due to their bio-plausible computations. Previous studies identified that DRAM-based off-chip memory accesses dominate the energy consumption of SNN processing. However, state-of-the-art works do not optimize the DRAM energy-per-access, thereby hindering the SNN-based systems from achieving further energy efficiency gains. To substantially reduce the DRAM energy-per-access, an effective solution is to decrease the DRAM supply voltage, but it may lead to errors in DRAM cells (i.e., so-called *approximate DRAM*). Toward this, we propose *EnforceSNN*, a novel design framework that provides a solution for resilient and energy-efficient SNN inference using reduced-voltage DRAM for embedded systems. The key mechanisms of our EnforceSNN are: (1) employing quantized weights to reduce the DRAM access energy; (2) devising an efficient DRAM mapping policy to minimize the DRAM energy-per-access; (3) analyzing the SNN error tolerance to understand its accuracy profile considering different bit error rate (BER) values; (4) leveraging the information for developing an efficient fault-aware training (FAT) that considers different BER values and bit error locations in DRAM to improve the SNN error tolerance; and (5) developing an algorithm to select the SNN model that offers good trade-offs among accuracy, memory, and energy consumption. The experimental results show that our EnforceSNN maintains the accuracy (i.e., no accuracy loss for $BER \leq 10^{-3}$) as compared to the baseline SNN with accurate DRAM while achieving up to 84.9% of DRAM energy saving and up to 4.1x speed-up of DRAM data throughput across different network sizes.

KEYWORDS

spiking neural networks, high performance, energy efficiency, approximate computing, approximate DRAM, DRAM errors, error tolerance, resilience

1. Introduction

Spiking neural networks (SNNs) have demonstrated the potential of obtaining high accuracy under unsupervised settings and low operational energy due to their bio-plausible spike-based computations (Putra and Shafique, 2020). A larger SNN model is usually favorable as it offers higher accuracy than the smaller ones, as shown by our experimental results in Figure 1A. Here, the 1MB-sized network achieves only 75%, while the 200MB-sized network achieves 92% accuracy for the MNIST dataset. This MNIST dataset is a set of training and test images for handwritten digits 0–9 (Lecun et al., 1998). On the other hand, most of the SNN hardware platforms have relatively small on-chip memory, e.g., less than 100MB (Roy et al., 2017; Sen et al., 2017; Frenkel et al., 2019a,b). Therefore, running an SNN model with a larger size than the on-chip memory of SNN hardware platforms will require intensive access to the off-chip memory. Previous studies show that single access to the off-chip memory (i.e., DRAM) incurs significantly higher energy consumption than single access to the on-chip memory (i.e., SRAM) (Sze et al., 2017; Putra et al., 2021b). Moreover, previous study also identified that memory access dominate the energy consumption of SNN processing, incurring 50–75% of the total system energy across different SNN hardware platforms, as shown in Figure 1B. The reason is that DRAM access energy is significantly higher than other SNN operations (e.g., neuron operations) (Krithivasan et al., 2019). This problem is even more critical for AI applications with stringent constraints (e.g., low-cost embedded devices with a small on-chip memory size) (Shafique et al., 2021) since it leads to even more intensive DRAM accesses. Consequently, this problem hinders SNN-based embedded systems from obtaining further efficiency gains.

Targeted Research Problem: *How can we substantially decrease the DRAM access energy for the SNN inference, while maintaining accuracy*. The solution to this problem will enable efficient SNN inference for energy–constrained embedded devices and their applications for Edge-AI and Smart CPS. *Edge-AI* is the system that runs Artificial Intelligence (AI) algorithms on resource- and energy-constrained computing devices at the edge of the network, i.e., close to the source of data (Shi et al., 2016; Satyanarayanan, 2017; Yu et al., 2018; Chen and Ran, 2019; Liu et al., 2019; Cao et al., 2020). Meanwhile, *Smart CPS (Cyber-Physical System)* is the system that includes the interacting networks of computational components (e.g., computation and storage devices), physical components (e.g., sensors and actuators), and human users (Chattopadhyay et al., 2017; Griffor et al., 2017; Kriebel et al., 2018; Shafique et al., 2018).

1.1. State-of-the-art and their limitations

To decrease the energy consumption of SNN inference, state-of-the-art works have developed different techniques, which can be loosely classified as the following.

- **Reduction of the SNN operations** through approximate neuron operations (Sen et al., 2017), weight pruning (Rathi et al., 2019), and neuron removal (Putra and Shafique, 2020). These techniques decrease the number of DRAM accesses for the corresponding model parameters.
- **Quantization** by reducing the range of representable values for SNN parameters (e.g., weights) (Rathi et al., 2019; Putra and Shafique, 2020; Putra and Shafique, 2021a). These techniques reduce the amount of SNN parameters (e.g., weights) to be stored in and fetched from DRAM.

Limitations: These state-of-the-art works mainly aim at reducing the number of DRAM accesses, but do not optimize the DRAM energy-per-access and do not employ approximations in DRAM that provide an additional knob for obtaining high energy efficiency. Therefore, optimization gains offered by these works are sub-optimal, hindering the SNN inference systems from achieving the full potential of DRAM energy savings. Therefore, the effective optimization should jointly minimize the DRAM energy-per-access and the number of DRAM accesses, by leveraging the approximation in DRAM to expose the full energy-saving potential, while overcoming the negative impact of the approximation-induced errors (i.e., bit-flips in DRAM cells). Figure 1C shows the approximation-induced error rates in DRAM.

To address these limitations, *we employ approximate DRAM (i.e., DRAM with reduced supply voltage) with efficient DRAM data mapping policy and fault-aware training to substantially reduce the DRAM access energy in SNN inference systems while preserving their accuracy*. Moreover, our proposed technique can also be combined with state-of-the-art techniques to further improve the energy efficiency of SNN inference. For example, Figure 1D shows the estimated DRAM energy savings achieved by our technique when combined with the weight pruning. To highlight the potential of reduced-voltage approximate DRAM, we perform an experimental case study in the following section.

1.2. Motivational case study and key challenges

In the case study, we aim at studying (1) the dynamics of DRAM bitline voltage ($V_{bitline}$) for both the accurate and

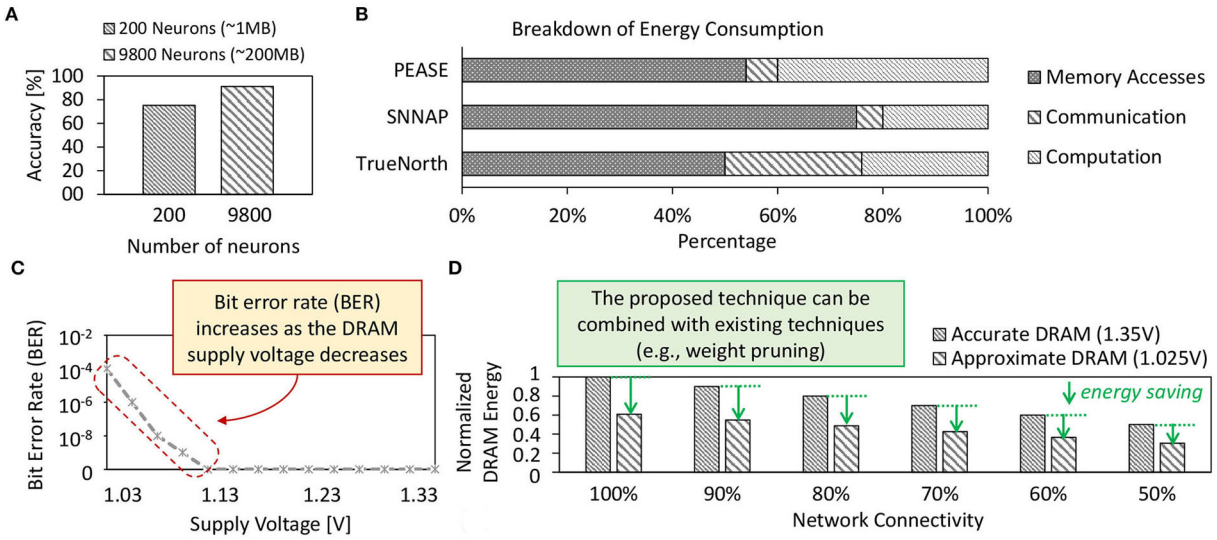

FIGURE 1
(A) Accuracy profiles of small-sized and large-sized SNN models on the MNIST dataset which are obtained from our experiments. **(B)** Breakdown of the energy consumption of SNN processing on different SNN hardware platforms, i.e., PEASE (Roy et al., 2017), SNNAP (Sen et al., 2017), and TrueNorth (Akopyan et al., 2015); adapted from studies in Krithivasan et al. (2019). **(C)** Bit error rate (BER) of approximate DRAM and its respective supply voltage V_{supply}; adapted from studies in Chang et al. (2017). **(D)** The estimated DRAM energy savings achieved by our technique when combined with the weight pruning across different rates of network connectivity (i.e., synaptic connections) for a network with 4900 excitatory neurons. The results are obtained from experiments using the LPDDR3-1600 4Gb DRAM configuration and the DRAMPower simulator (Chandrasekar, 2014).

approximate DRAM settings, and (2) the DRAM access energy for different access conditions (including a row buffer hit, miss, or conflict). Note, $V_{bitline}$ is defined as the voltage measured in each DRAM bitline when a DRAM supply voltage (V_{supply}) is applied, as shown in Figures 2A, 4C. Further details on the dynamics of $V_{bitline}$ are provided in Section 2.2.2. For DRAM access conditions, a row buffer hit means that the requested data has been loaded in the DRAM row buffer, thus the data can be accessed without additional DRAM operations. Meanwhile, a row buffers miss or conflict needs to open the requested DRAM row before the data can be loaded into the row buffer and then accessed. Further information on the DRAM access conditions is discussed in Section 2.2.1.

For the experimental setup, we employ the DRAM circuit model from the study of Chang et al. (2017) and the SPICE simulator to study the dynamics of $V_{bitline}$. The accurate DRAM operates at 1.35V of the supply voltage (V_{supply}), while the approximate one operates at 1.025V. Further details on the experimental setup are discussed in Section 5. Furthermore, we consider the LPDDR3-1600 4Gb DRAM configuration as it is representative of the low-power DRAM types for embedded systems. We employ the DRAMPower simulator to estimate the DRAM access energy because it has been validated against real measurements (Chandrasekar, 2014) and has been widely used in the computer architecture communities. Figure 2 presents the experimental results, from which we make the following key observations.

- The $V_{bitline}$ decreases as the V_{supply} decreases, hence forcing the DRAM cells to operate under lower reliability as the *weak cells* may fail to hold the correct bits. *Weak cells* are DRAM cells that fail when the DRAM parameters (e.g., voltage, timing) are reduced (Chang et al., 2017; Kim et al., 2018).
- The reduced-voltage DRAM decreases the DRAM energy-per-access across different access conditions, i.e., by up to 42% of energy reduction for each access.
- The row buffer hit has lower energy consumption than the row buffer miss or conflict. Moreover, row buffer hit also incurs less latency than the row buffer miss or conflict (Putra et al., 2020; Putra et al., 2021b). Therefore, the row buffer hit should be exploited to optimize the DRAM latency and energy.

Although employing the approximate DRAM can substantially decrease the DRAM energy-per-access, it also decreases the DRAM reliability since the bit errors increase when the V_{supply} decreases, as shown in Figure 1C. These errors may degrade the accuracy of SNN inference since they can change the weight values in DRAM, which then deteriorates the neuron behavior.

Associated Research Challenge: *How to achieve low DRAM access energy for SNN inference using approximate DRAM, while minimizing their negative impact on the accuracy.*

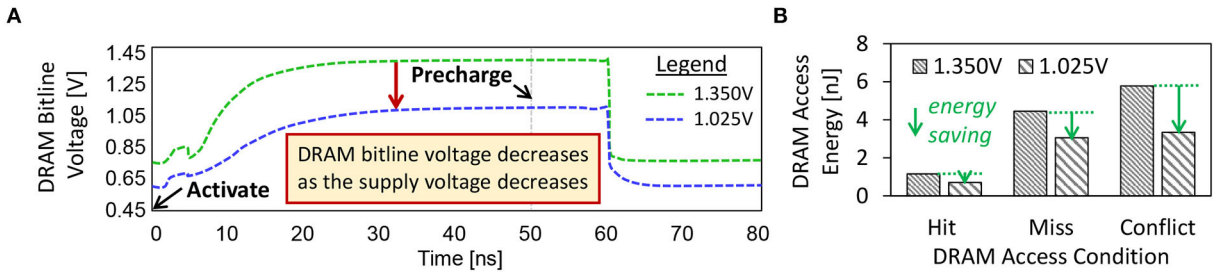

FIGURE 2
(A) The dynamics of $V_{bitline}$ under different V_{supply} values. **(B)** DRAM access energy for a row buffer hit, a row buffer miss, and a row buffer conflict, under different V_{supply} values.

1.3. Our novel contributions

To overcome the above research challenges, we propose **the EnforceSNN framework**, which enables resilient and energy-efficient SNNs considering approximate DRAMs (i.e., reduced-voltage DRAMs) for embedded systems. Based on the best of our knowledge, it is the first effort that employs approximate DRAM for improving the energy efficiency of SNN inference, while enhancing their error tolerance against bit errors in DRAM. Our EnforceSNN framework employs the following key steps.

1. **Employing weight quantization** to reduce the memory footprint for SNN weights and the number of DRAM accesses for SNN inference, thereby optimizing the DRAM access energy.
2. **Devising an efficient DRAM data mapping** to maximize row buffer hits for optimizing the DRAM energy-per-access while considering BER in DRAM.
3. **Analyzing the SNN error tolerance** to understand the SNN accuracy profile under different DRAM supply voltage and different BER values.
4. **Improving the SNN error tolerance** by developing and employing efficient fault-aware training (FAT) that considers SNN accuracy profile and bit error locations in DRAM.
5. **Devising an algorithm to select the SNN model** that offers good trade-offs among accuracy, memory, and energy consumption from the given model candidates using the proposed reward function.

Key Results: We evaluate the EnforceSNN framework for (1) classification accuracy using PyTorch-based simulations (Hazan et al., 2018) on a multi-GPU machine considering the MNIST and Fashion MNIST datasets[1], and (2) DRAM access energy using DRAMPower (Chandrasekar, 2014). We perform an epoch of unsupervised learning (60K experiments) for each retraining process considering each combination of the SNN model, workload, and training BER; then perform inference (10K experiments) for each combination of the SNN model, workload, and testing BER. The experimental results indicate that our EnforceSNN reduces the DRAM access energy by up to 84.9% and improves the speed-up up to 4.1x while maintaining the accuracy (no accuracy loss) across different network sizes for $BER \leq 10^{-3}$.

1 The research works for the unsupervised learning-based SNNs is still in the early stage and typically uses small datasets like the MNIST and the Fashion MNIST. We also adopt the same test conditions as used widely by the SNN research community.

2. Background

2.1. Spiking neural networks

Spiking Neural Networks are the neural network models that employ bio-plausible computations and use the sequences of spikes (i.e., *spike trains*) for conveying information. These spikes are encoded using a specific spike coding. Several spike coding schemes have been proposed in the literature (Gautrais and Thorpe, 1998; Thorpe and Gautrais, 1998; Kayser et al., 2009; Park et al., 2019; Park et al., 2020). Here, we use rate coding as it has been used widely and offers robustness for diverse learning rules (Diehl and Cook, 2015; Putra et al., 2021a). For the learning rule, we use the spike-timing-dependent plasticity (STDP), as it has been widely used by previous works (Diehl and Cook, 2015; Hazan et al., 2018; Saunders et al., 2018; Putra and Shafique, 2021b). In an SNN model, neurons and synapses are connected in a specific architecture (Pfeiffer and Pfeil, 2018; Mozafari et al., 2019; Tavanaei et al., 2019). Here, we consider a fully-connected architecture as it supports unsupervised learning scenarios; refer to Figure 3A. In this architecture, each input pixel is connected to all (excitatory) neurons, and the output of each neuron is connected to other neurons for providing inhibition. For the neuron model, we use the Leaky Integrate-and-Fire (LIF) as it provides bio-plausible neuronal dynamics with low computational complexity (Izhikevich, 2004; Putra et al., 2022); refer to Figure 3B.

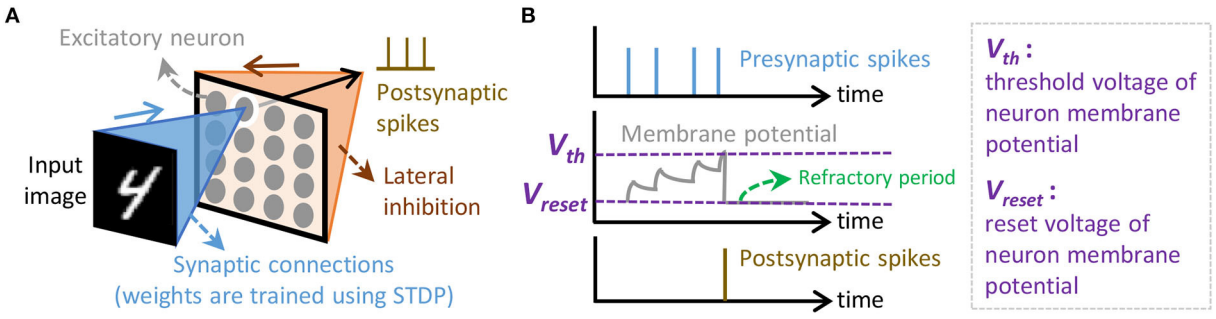

FIGURE 3
(A) The SNN architecture considered in this study, which is adapted from Putra and Shafique (2020). **(B)** The neuronal dynamics of the Leaky Integrate-and-Fire (LIF) neuron model.

2.2. Approximate DRAM

2.2.1. DRAM fundamentals

The organization of a DRAM consists of channel, module, rank, chip, bank, subarray, row, and column (Putra et al., 2020; Olgun et al., 2021); refer to Figure 4A. A single DRAM request can access data from multiple DRAM chips within the same rank in parallel. In each DRAM chip, the request is routed to a specific bank, row, and column address. When the activation (ACT) command is issued, the requested row is opened and its data are copied to the row buffer. If the read (RD) command is issued, data in the row buffer can be read. If the write (WR) command is issued, data in the row buffer can be replaced with the new one. In each DRAM request, there are different possible access conditions, i.e., a row buffer hit, miss, and conflict (Ghose et al., 2019). A row buffer hit refers to a condition when the requested row is activated and its data are already in the row buffer. Hence, the data can be accessed directly without additional operation. Otherwise, the requested row is still closed, and the condition is either a row buffer miss or conflict. A row buffer miss is defined if there is no activated row when a request happens, thus the requested row should be activated before accessing the data. Meanwhile, a row buffer conflict is defined when the requested row is still closed, but the row buffer is occupied by another activated row. Hence, the activated row should be closed using the precharging (PRE) command, before activating the requested row using the activation (ACT) command. Figure 4B illustrates the DRAM commands (i.e., ACT, RD or WR, PRE) and their timing parameters (i.e., t_{RCD}: row address to column address delay, t_{RAS}: row active time, and t_{RP}: row precharge time).

2.2.2. Reduced-voltage DRAM

We perform extensive experiments using the SPICE simulator and the DRAM circuit model from Chang et al. (2017) while considering different supply voltage (V_{supply}) values, to characterize the parameters of reduced-voltage DRAM: including the bitline voltage ($V_{bitline}$) and the respective timing parameters (i.e., t_{RP}, t_{RCD}, and t_{RAS}). The experimental results are presented in Figure 4C, and the obtained parameters are used for further DRAM energy estimation. The ready-to-access voltage is defined as the condition when $V_{bitline}$ reaches 75% of V_{supply}, which represents the minimum t_{RCD} for reliable DRAM operations, as shown by Ⓐ. The ready-to-precharge voltage is defined as the condition when $V_{bitline}$ reaches 98% of V_{supply}, which represents the minimum t_{RAS} for reliable DRAM operations, as shown by Ⓑ. Meanwhile, the ready-to-activate voltage is defined as the condition when the $V_{bitline}$ is within 2% of $V_{supply}/2$, which represents the minimum t_{RP} for reliable DRAM operations, as shown by Ⓒ.

3. Error modeling for approximate DRAM

The study of Koppula et al. (2019) has proposed four error models that closely fit the real reduced-voltage-based approximate DRAMs as the following. **Error Model-0**: the bit errors follow a *uniform random distribution* across a DRAM bank; **Error Model-1**: the bit errors follow a *vertical distribution* across the bitlines of a DRAM bank; **Error Model-2**: the bit errors follow a *horizontal distribution* across the wordlines of a DRAM bank; and **Error Model-3**: the bit errors follow a *uniform random distribution* that depends on the content of the DRAM cells. In this study, we employ the **DRAM Error Model-0**, due to the following reasons: (1) it produces errors with high similarity to the real reduced-voltage-based approximate DRAM by using the percentage of weak cells and the error probability in any weak cell; (2) it offers a reasonable approximation of other error models, including the approximation of (a) errors across bitlines similar to Error Model-1, (b) errors across wordlines similar to Error Model-2, and (c) uniform random distribution similar to Error Model-3; and (3) it provides fast error injection by

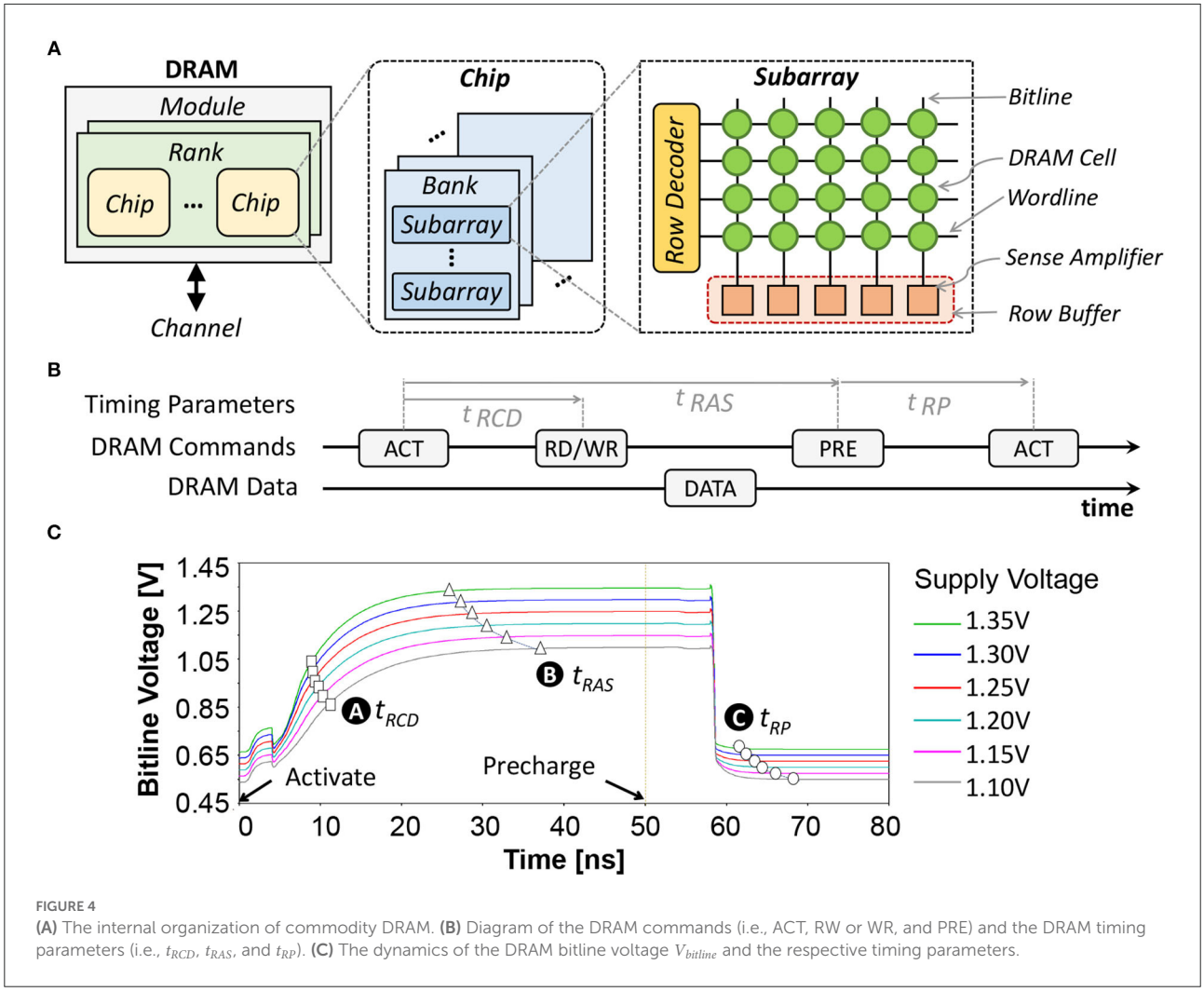

FIGURE 4
(A) The internal organization of commodity DRAM. **(B)** Diagram of the DRAM commands (i.e., ACT, RW or WR, and PRE) and the DRAM timing parameters (i.e., t_{RCD}, t_{RAS}, and t_{RP}). **(C)** The dynamics of the DRAM bitline voltage $V_{bitline}$ and the respective timing parameters.

software. Previous work (Koppula et al., 2019) also employed the DRAM Error Model-0 majorly due to similar reasons.

4. EnforceSNN framework

4.1. Overview

Our EnforceSNN framework employs several key steps as shown in Figure 5. First, we *quantize the SNN weights* to reduce memory footprint and DRAM access energy (Section 4.2). Second, we devise *an error-aware DRAM data mapping policy* to optimize the DRAM energy-per-access (Section 4.3). These optimizations contribute to 4.1x inference speed-up and 84.9% DRAM access energy saving (Section 6). Then, we *analyze the SNN error tolerance* to understand the accuracy profile of SNN inference under different BER values (Section 4.4). We leverage this information to *develop an efficient FAT technique that improves the SNN error tolerance* (Section 4.5). These fault tolerance techniques contribute to 2.71x retraining speed-up without accuracy loss for $BER \leq 10^{-3}$ (Section 6). We also *develop an SNN model selection algorithm* to find a model that provides good trade-offs among accuracy, memory, and energy consumption (Section 4.6). Details of these mechanisms are explained in the following subsections.

4.2. Quantizing the SNN weights

We perform weight quantization to substantially reduce the memory footprint and the number of DRAM accesses, which leads to DRAM energy saving. The reason is that quantization is a prominent technique for reducing the memory footprint of neural networks without decreasing the accuracy significantly (Gupta et al., 2015; Micikevicius et al., 2018). Moreover, it is the first effort to study and exploit SNN weight quantization considering approximation errors in DRAM, thereby providing new insights as compared to previous studies on SNN weight quantization. Our weight quantization considers the fixed-point

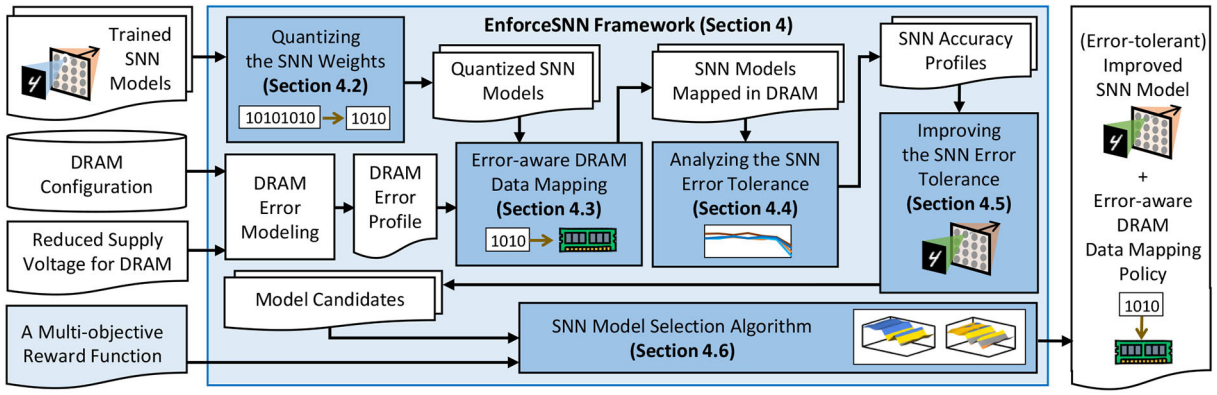

FIGURE 5
The EnforceSNN framework and its novel mechanisms (highlighted in blue boxes).

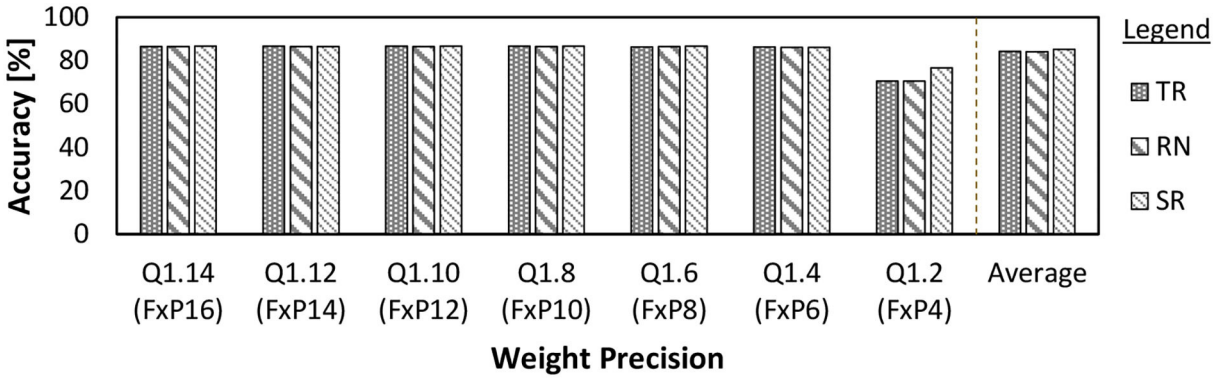

FIGURE 6
The accuracy of a 900-neuron network on the MNIST across different precision levels and rounding schemes, i.e., truncation (TR), round to the nearest (RN), and stochastic rounding (SR). Here, we follow the definition of TR, RN, and SR from the study of Hopkins et al. (2020). Note that the fixed-point format *Qi.f* can also be represented as FxP(1+*i*+*f*). These results show that employing TR on top of the FxP8 precision leads to competitive accuracy as compared to other rounding schemes (i.e., RN and SR).

format which can be represented as *Qi.f*. It denotes 1 sign bit, i integer bits, and f fractional bits, and follows the 2's complement format (i.e., signed *Qi.f*). Here, the range of representable values is $[-2^i, 2^i - 2^{-f}]$ with the precision of $\epsilon = 2^{-f}$. We select the "signed *Qi.f*" format to show that our EnforceSNN framework provides a generic solution with high applicability for different variants of bio-plausible learning rules (e.g., STDP variants) which may lead to positive or negative weight values (Rahimi Azghadi et al., 2014; Diehl and Cook, 2015). To do this, we perform a fixed-point quantization to the trained SNN weights using a specific rounding scheme, such as truncation, round to the nearest, or stochastic rounding. For a study case, we select truncation as it provides competitive accuracy with low computational complexity (Putra and Shafique, 2021a, 2022a,b). To illustrate this, we evaluate the impact of different rounding schemes on the accuracy through an experimental case study, and the results are shown in Figure 6. Truncation (TR) keeps the f bits and removes the other fractional bits. Therefore, the output fixed-point for the given real number x with *Qi.f* format is defined as $TR(x, Qi.f) = \lfloor x \rfloor$. In our SNN model, we employ the pair-based weight-dependent STDP learning rule from the study of Diehl and Cook (2015) that bounds each weight value (w) within the defined range, i.e., $w = [0, 1]$. Consequently, applying the truncation to the weights will round the value down. In this study, we consider an 8-bit fixed-point with "signed Q1.6" and 2's complement format (i.e., 1 sign bit, 1 integer bit, and 6 fractional bits), since it provides high accuracy for SNNs under unsupervised learning scenarios (Putra and Shafique, 2021a). Note, we can also employ the "unsigned *Qi.f*" format without sign bit to represent the 8-bit non-negative weights (i.e., 1 integer bit and 7 fractional bits) in the EnforceSNN if desired. For both "signed *Qi.f*" and "unsigned *Qi.f*" formats, 1 bit for the integer part is required for representing the maximum possible weight value (i.e., $w = 1$).

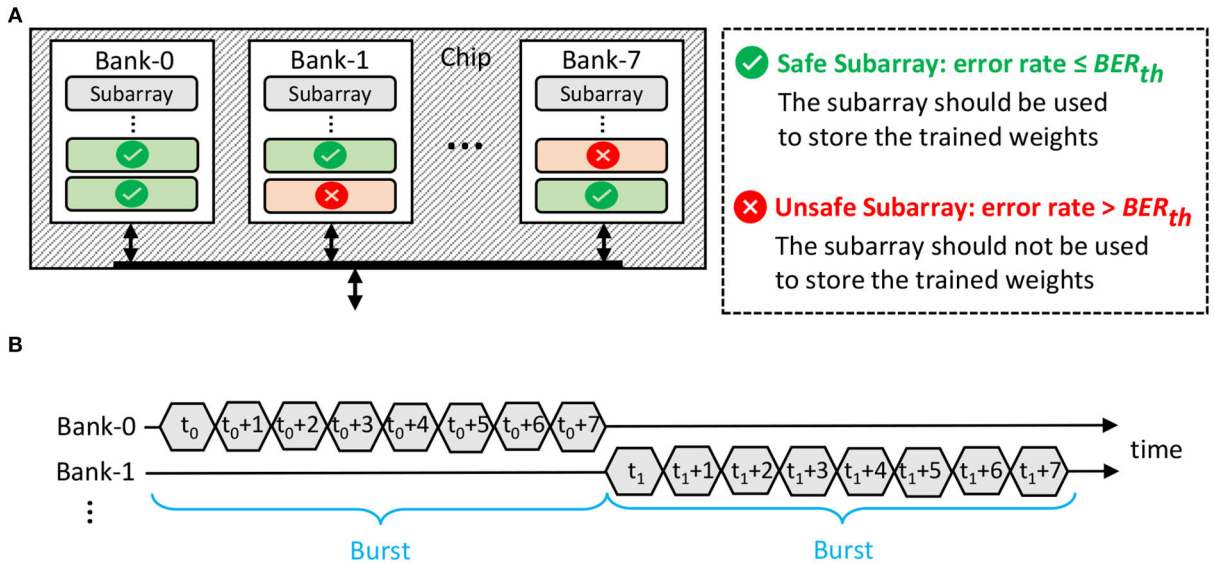

FIGURE 7
(A) Our proposed DRAM mapping policy, leveraging subarray-level granularity. **(B)** The timing diagram of the DRAM multi-bank burst feature.

Quantization Steps: We quantize only the weights through the simulated quantization approach, which represents the weight values under fixed-point format, and performing computations under floating-point format (Jacob et al., 2018; Krishnamoorthi, 2018; Gholami et al., 2021; van Baalen et al., 2022). To perform quantization, we convert the weight values from 32-bit floating-point format (FP32) to 8-bit fixed-point format (signed Q1.6) by constructing their 8-bit binary representations under 32-bit integer format (INT32), thereby conveniently performing bit-wise modification and rounding operation while considering the sign and the rounding scheme (i.e., truncation). Afterward, we convert the quantized weight values (INT32) to FP32 format through casting and then normalize the values by 2^f. Hence, the 8-bit binary representations of quantized weight values are saved in FP32 and can be used in the FP32-based arithmetic computations.

DRAM Error Injection: *If there is no DRAM error*, the quantization steps are performed and the quantized weight values (in FP32) can be used for computations in SNN processing. *If DRAM errors exist*, the quantization steps are performed while considering the DRAM error injection. These errors are injected into the 8-bit binary representations of quantized weights (in INT32) under a specific DRAM data mapping policy. Afterward, we convert the binary representations of quantized weights (in INT32) to FP32 format, so that the quantized weight values can be used for computations in SNN processing.

4.3. Error-aware DRAM data mapping policy

It is important to map the SNN model properly in DRAM to ensure that (1) the weights are minimally affected by errors in DRAM so that the accuracy is maintained, and (2) the DRAM energy-per-access is optimized. Toward this, *we devise and employ an error-aware DRAM mapping policy to place the SNN weights in DRAM, while optimizing the DRAM energy-per-access*. The proposed DRAM mapping policy is illustrated in Figure 7A, and its key ideas are explained in the following.

1. The weights are mapped in the appropriate DRAM part (e.g., chip, bank, or subarray), whose error rate meets the BER requirement, i.e., $\leq$ the maximum tolerable BER (BER_{th}). Here, we consider the subarray-level granularity for data mapping, since it allows us to exploit the following features.

    - *The multi-bank burst feature*, which is available in commodity DRAM, can be employed to increase the throughput. Its timing diagram is illustrated in Figure 7B.
    - *The subarray-level parallelism*, which is available in novel DRAM architectures (Kim et al., 2012), can also be employed to increase the throughput.

    We determine BER_{th} through experiments that investigate the accuracy profile of a network across different BER values. Figure 8 shows the experimental results for a 900-neuron

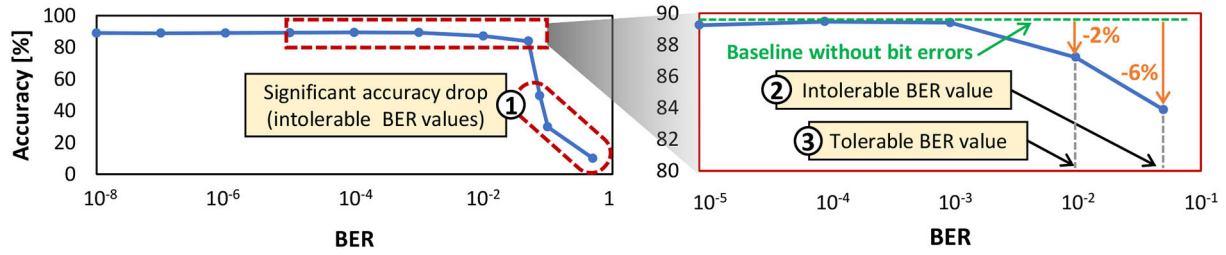

FIGURE 8
The test accuracy profile of a 900-neuron network across different BER values shows the tolerable and intolerable BER. This network has a fully-connected architecture like in Figure 3A where each input pixel is connected to all neurons, and the output of each neuron is connected to other neurons for performing inhibition, thereby providing competition among neurons to correctly recognize the input class.

network. If the accuracy scores are significantly lower than the baseline accuracy without bit errors (i.e., >3% accuracy degradation), we refer to the respective BER values as *intolerable BER*, as shown in ① and ② in Figure 8. Otherwise, we define them as *tolerable BER*. For instance, ③ in Figure 8 shows an accuracy that is associated with a tolerable BER. Based on this discussion, we define $BER_{th} = 10^{-2}$.

2. The weights are mapped in a way to maximize the row buffer hits for optimizing the DRAM energy-per-access while exploiting the multi-bank burst feature for maximizing the data throughput. The reason is that a row buffer hit incurs the lowest DRAM access energy than other access conditions (i.e., a row buffer miss or conflict), as suggested by our experimental results in Figure 2B.

To efficiently implement the above ideas, we devise Algorithm 1 with the following key steps. First, we identify the subarrays whose error rate $\leq BER_{th}$ and refer to them as *the safe subarrays*, which should be used for storing the weights. Otherwise, we refer to the subarrays whose error rate $> BER_{th}$ as *the unsafe subarrays*, which should not be used for storing the weights. This step is represented in line 7 of Algorithm 1. Second, to maximize the row buffer hits and exploit the multi-bank burst feature, the data mapping in each DRAM chip should follow the following policy (represented in lines 3-8 of Algorithm 1).

- **Step-1:** Assume that we consider mapping data in a target subarray of the target bank with the following initial indices, i.e., *subarray_index = i, bank_index = j*.
- **Step-2:** If the target subarray is a safe subarray, then we prioritize mapping the data in different columns of the same row for maximizing row buffer hits. Otherwise, this subarray is not utilized and we move to another target subarray in a different bank (*subarray_index = i, bank_index += 1*). Then, we perform **Step-2** again. If all columns in the same row across all banks are filled or unavailable, then we move to another subarray in the initial bank (*subarray_index += 1, bank_index = j*) to exploit subarray-level parallelism, if applicable.
- **Step-3:** In the target subarray, the remaining data are mapped in the same fashion as **Step-2**. When all columns in the same row of all safe subarrays across all banks are filled, then the remaining data are placed in a different row of the initial target subarray and bank (*subarray_index = i, bank_index = j*). Afterward, we perform **Step-2** to **Step-3** again until all data are mapped in a DRAM chip. If some data remain but there are no available spaces in a DRAM chip, then we move to **Step-4**.
- **Step-4:** The remaining data are mapped using **Step-1** to **Step-3** in different DRAM chips, ranks, and channels, respectively if applicable.

4.4. Analyzing the SNN error tolerance

Previous discussion highlights that bit errors in the SNN weights can degrade the accuracy, as they change the weight values and deviate the neuron behavior from the correct classification. Therefore, SNN error tolerance should be improved, so that the SNN model can achieve high accuracy even in the presence of a high error rate. To effectively enhance the SNN error tolerance, it is important to understand the SNN accuracy profile under DRAM errors. Toward this, *our EnforceSNN analyzes the accuracy profile of the SNN model considering the data mapping pattern in DRAM and different BER values*. We observe that the accuracy profile typically has acceptable accuracy (i.e., within 1% accuracy degradation from the baseline without errors) when BER is low and has notable accuracy degradation when BER is high. Therefore, we classify the accuracy profile into two regions: Ⓐ a region with acceptable accuracy, and Ⓑ a region with unacceptable accuracy, as shown in Figure 9. These insights will be leveraged for developing an efficient enhancement technique for improving the SNN error tolerance in Section 4.5.

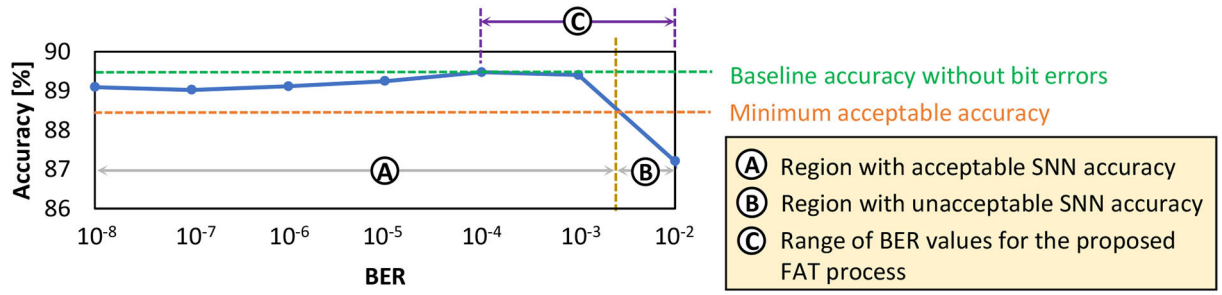

FIGURE 9
The test accuracy profile of a 900-neuron network shows the region with acceptable accuracy, the region with unacceptable accuracy, and the range of BER values for the proposed fault-aware training. Note that this network is the same as the one in Figure 8. Here, the observation focuses on the BER values that can be considered in the retraining process, thereby having a smaller range than the one in Figure 8.

Algorithm 1 The proposed DRAM mapping policy.

INPUT: **(1)** DRAM (*DRAM*): number of channel (n_{ch}), number of rank-per-channel (n_{ra}), number of chip-per-rank (n_{cp}), number of bank-per-chip (n_{ba}), number of subarray-per-bank (n_{su}), number of row-per-subarray (n_{ro}), number of column-per-row (n_{co});
(2) Bit error rate (BER): BER of a subarray (*BER_subarray*), maximum tolerable BER (BER_{th});
(3) Data (*data*);
OUTPUT: DRAM (*DRAM*);
BEGIN
Process:
for $ch = 0$ to $(n_{ch} - 1)$ **do**
  for $ra = 0$ to $(n_{ra} - 1)$ **do**
    for $cp = 0$ to $(n_{cp} - 1)$ **do**
      for $ro = 0$ to $(n_{ro} - 1)$ **do**
        for $su = 0$ to $(n_{su} - 1)$ **do**
          for $ba = 0$ to $(n_{ba} - 1)$ **do**
            if $BER\_subarray[ch, ra, cp, ba, su] \leq BER_{th}$ **then**
              for $co = 0$ to $(n_{co} - 1)$ **do**
                $DRAM[ch, ra, cp, ba, su, ro, co] \leftarrow data$;
              end for
            end if
          end for
        end for
      end for
    end for
  end for
end for
return *DRAM*;
END

4.5. Improving the SNN error tolerance

Our EnforceSNN enhances the SNN error tolerance through the fault-aware training (FAT) technique that incorporates the error profile of the approximate DRAM. We consider efficiently performing FAT for minimizing training time, energy consumption, and carbon emission during the retraining process (Strubell et al., 2019, 2020), by conducting a small yet effective number of iterations for the retraining process, while avoiding *accuracy collapse*. Accuracy collapse is defined as a significant accuracy degradation due to training divergence that is caused by introducing high BER immediately at the beginning of the retraining process (Koppula et al., 2019). The proposed FAT technique has the following key steps, which are also presented in Algorithm 2.

- **Step-1:** We define the range of BER values that will be incorporated in the training process to make the SNN model adaptable to DRAM errors, as shown by region-Ⓒ in Figure 9. We incorporate (1) BER values in region-Ⓐ that are close to the region-Ⓑ, and (2) all BER values in the region-Ⓑ, in the training process. Specifically, we consider the two highest BER values in region-Ⓐ in the training process to make the model adapt to high fault rates safely, without suffering from significant accuracy degradation (i.e., accuracy collapse) with less training time.
- **Step-2:** The bit errors in DRAM are generated for different BER values (which correspond to different V_{supply} values), based on the DRAM error model-0 that follows a uniform random distribution across a DRAM bank.
- **Step-3:** The generated bit errors are then injected into the DRAM cell locations, and the weight bits in these locations are flipped. In this step, we consider the proposed DRAM data mapping discussed in Section 4.3 for maximizing the row buffer hits and exploiting the multi-bank burst feature.
- **Step-4:** Afterward, we include the generated bit errors in the retraining by incrementally increasing the BER from the minimum rate to a maximum one following the defined range of BER values from **Step-1**. We increase the BER value after each epoch of retraining by a defined ratio (e.g., 10x of the previous error rate). In this manner, the

SNN model is gradually trained to tolerate DRAM errors from the defined lowest rate to the maximum one, thereby carefully improving the SNN error tolerance.

4.6. Algorithm for SNN model selection

From the previous steps, we may get different sizes of error-tolerant SNN models as potential solutions for the given embedded applications. Therefore, we need to consider design trade-offs to select the most appropriate model for the given accuracy, memory, and energy constraints. Toward this, *we propose an algorithm to quantify the trade-off benefits of the SNN model candidates using our proposed reward function, and then select the one with the highest benefit.* The idea of our multi-objective reward function (R) is to prioritize the model that has high accuracy, small memory, and low energy consumption. The reward R is defined as the resultant between the accuracy with the memory and energy consumption, as expressed in Equation 1. In this equation, acc_x denotes the accuracy of the investigated SNN model (x). m_{norm} denotes the normalized memory, which is defined as the ratio between the memory footprint of the investigated model (mem_x) and the floating-point model (mem_{fp}); refer to Equation 2. The memory footprint of the model is estimated by leveraging the number of weights (N_{wgh}) and the corresponding bit-width (BW_{wgh}); refer to Equation 3. Meanwhile, E_{norm} denotes the normalized energy consumption, which is defined as the ratio between the DRAM access energy of the approximate DRAM (E_{DRAM_approx}) and the accurate one ($E_{DRAM\_accurate}$) for the investigated model; refer to Equation 4. To define the significance of memory and energy consumption with respect to the accuracy when calculating R, we employ μ and ε as the adjustable trade-off variables for memory and energy consumption, respectively. Here, μ and ε are the non-negative real numbers.

$$R(acc_x, m_{norm}, E_{norm}) = acc_x - (\mu \cdot m_{norm} + \varepsilon \cdot E_{norm}) \quad (1)$$

$$m_{norm} = \frac{mem_x}{mem_{fp}} \quad (2)$$

$$mem = N_{wgh} \cdot BW_{wgh} \quad (3)$$

$$E_{norm} = \frac{E_{DRAM\_approx}}{E_{DRAM\_accurate}} \quad (4)$$

5. Evaluation methodology

Figure 10 shows the experimental setup and tools flow for evaluating our EnforceSNN framework, which is explained in the following.

Accuracy Evaluation: We employ PyTorch-based simulations (Hazan et al., 2018) with 32-bit floating-point (FP32) and 8-bit fixed-point precision (i.e., FxP8 with "signed $Q1.6$" and "unsigned $Q1.7$" formats) that run on a multi-GPU machine, i.e., Nvidia GeForce RTX 2080 Ti. For network architecture, we consider the fully-connected network with a different number of excitatory neurons, which are referred to as N-i for conciseness with i denoting the number of excitatory neurons. We use the rate coding for converting each input pixel into a spike train and use the MNIST and Fashion MNIST datasets. For comparison partners, we use the SNN model which is pre-trained without considering DRAM errors as the baseline. We perform an epoch of STDP-based unsupervised learning through 60K experiments for each retraining process considering each combination of the SNN model, dataset, and training BER. Afterward, we perform inference through 10K experiments for each combination of the SNN model, dataset, and testing BER.

DRAM Error Generation and Injection: First, we generate bit errors based on the DRAM error model-0, and inject them into the DRAM cell locations, while considering the data mapping policy in DRAM. Afterward, the weight bits that are stored in the DRAM cell locations with errors, will be flipped. For the baseline data mapping, we place the weight bits in

Algorithm 2 The proposed FAT technique.

INPUT: **(1)** Baseline pre-trained SNN: model ($model_0$), accuracy ($model_0.acc$);
(2) DRAM error model ($DRAMerr$);
(3) BER for retraining: error rates (BER), number of error rates (N_{BER});
(4) Training dataset: samples (S_{train}), number of samples (N_{train});
(5) Test dataset: samples (S_{test}), number of samples (N_{test});

OUTPUT: **(1)** Improved SNN: model ($model_1$), accuracy ($model_1.acc$);

BEGIN
Initialization:
$model_{temp} = model_0$;
$model_1 = model_0$;
$model_1.acc = 0$;
Process:
for $i = 0$ to $(N_{BER} - 1)$ **do**
  $error\_map = DRAMerr(BER[i])$; // error generation
  inject $error\_map$ into $model_{temp}$; // error injection
  for $r = 0$ to $(N_{train} - 1)$ **do**
    train $model_{temp}$ with $S_{train}[r]$; // train
  end for
  for $s = 0$ to $(N_{test} - 1)$ **do**
    test $model_{temp}$ with $S_{test}[s]$; // test
  end for
  if $model_{temp}.acc > model_1.acc$ **then**
    $model_1 = model_{temp}$;
    $model_1.acc = model_{temp}.acc$;
  end if
end for
return $model_1$;
END

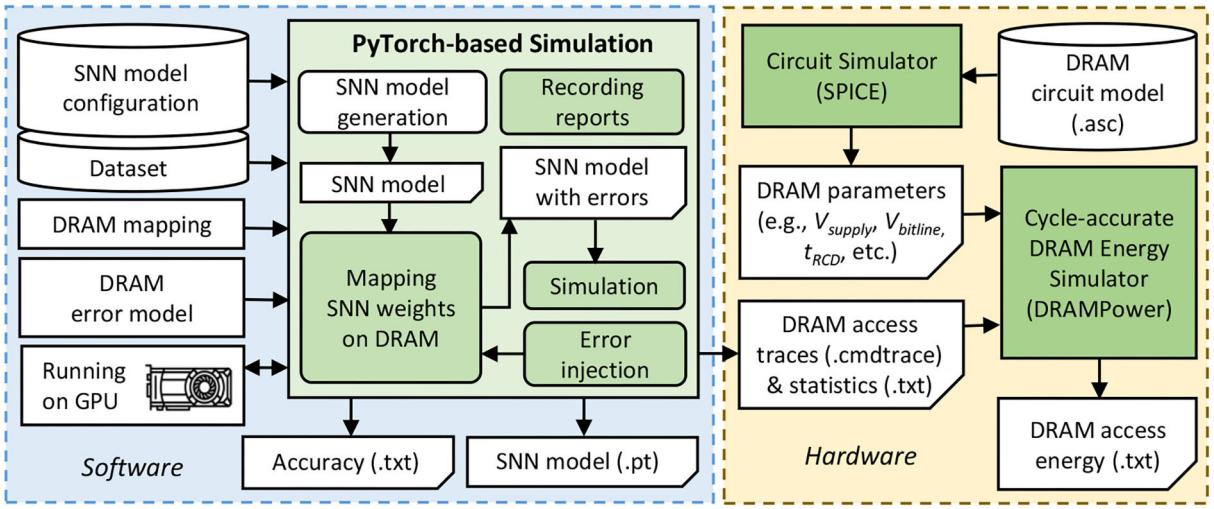

FIGURE 10
Our experimental setup and tools flow.

the subsequent address in a DRAM bank to maximize the DRAM burst feature, and if a DRAM bank is filled, then the weight bits are mapped in a different bank of the same DRAM chip. Meanwhile, we use the proposed DRAM mapping in Algorithm 1 for our EnforceSNN.

DRAM Energy Evaluation: We use the DRAM circuit model from the work of Chang et al. (2017) and the SPICE simulator to extract the DRAM operational parameters (e.g., V_{supply}, $V_{bitline}$, t_{RCD}, t_{RAS}, t_{RP}) while considering the configuration of LPDDR3-1600 4Gb DRAM which is representative for the main memory of embedded systems. The accurate DRAM operates with 1.35V of V_{supply}, while the approximate one operates with 1.025V-1.325V of V_{supply}. Afterward, we use the state-of-the-art cycle-accurate DRAMPower (Chandrasekar, 2014) that incorporates the DRAM access traces and statistics as well as the extracted DRAM parameters for estimating the DRAM access energy.

6. Results and discussion

6.1. Improvements of the SNN error tolerance

Figure 11 shows the accuracy of the baseline model and the EnforceSNN-improved model with accurate and approximate DRAM across different BER values and precision levels, i.e., FP32 and FxP8 ("signed Q1.6" and "unsigned Q1.7"), network sizes, and workloads (i.e., the MNIST and Fashion MNIST datasets). In general, we observe that the baseline model with approximate DRAM achieves lower accuracy than the baseline model with accurate DRAM, and the accuracy decreases as the BER increases. These trends are observed across different weight precision levels, network sizes, and datasets. The reason is that the weights are changed (i.e., flipped) if they are stored in the faulty DRAM cells, and these weights are not trained to adapt to such bit flips. Therefore, the corresponding neuron behavior deteriorates from the expected behavior, hence decreasing accuracy. On the other hand, the EnforceSNN-improved model with approximate DRAM improves accuracy over the baseline model with accurate and approximate DRAM, across different BER values, network sizes, and datasets, as shown in ❶. We also observe that the EnforceSNN-improved model with approximate DRAM improves the accuracy over the baseline model with accurate and approximate DRAM, even in the high error rate case (i.e., $BER = 10^{-2}$), as shown in ❷ for FP32 and ❸ for FxP8 weight precision levels. The reason is that our EnforceSNN incorporates the error profiles from the approximate DRAM across different BER values in the training process, which makes the SNN model adaptive to the presence of DRAM errors, thereby improving the SNN error tolerance. For the MNIST dataset, a high error rate (i.e., $BER = 10^{-2}$) typically decreases the accuracy of the SNN-FxP8 more than the SNN-FP32, as shown in ❹. The reason is that the MNIST dataset has a narrow weight distribution in each class to represent its digit features, hence bit errors may change the weight values significantly in the FxP8 precision than the FP32 due to its shorter bit-width. As a result, the corresponding neuron behavior deteriorates from its ideal behavior, hence degrading accuracy. For the Fashion MNIST dataset, the SNN-FxP8 may achieve higher accuracy than the SNN-FP32 in some cases, as shown in ❺. The potential reason is the following. The Fashion MNIST dataset has relatively more complex features than the MNIST dataset, hence having a wider weight distribution in

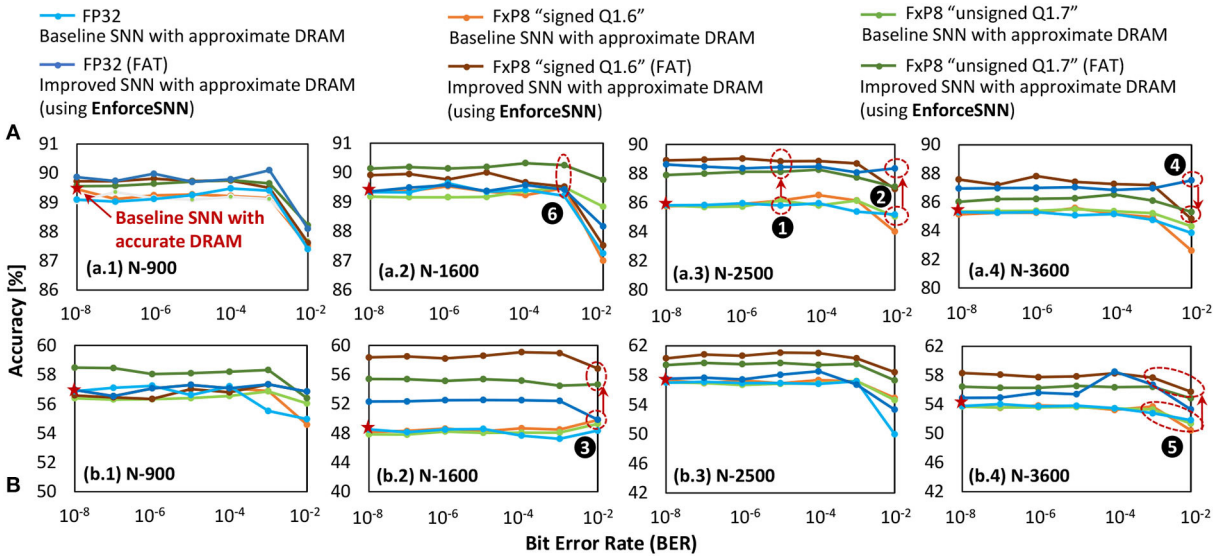

FIGURE 11
The accuracy of the baseline model with accurate and approximate DRAM, as well as the EnforceSNN-improved model with approximate DRAM for **(A)** MNIST and **(B)** Fashion MNIST datasets, across different precision levels, different BER values, and different network sizes.

each class to represent its various features which may overlap with features from other classes (i.e., non-unique features). Then, the quantization removes these non-unique features by eliminating the less significant bits of the trained weights (i.e., like the denoising effect), and the retraining makes the quantized weights adaptive to bit flips, thereby leading to higher accuracy than the non-quantized ones. Furthermore, we also observe that the accuracy of the SNN-FxP8 starts showing notable degradation at a high error rate (i.e., $BER = 10^{-2}$). For quantized models, in general, the "unsigned $Q1.7$" and "signed $Q1.6$" formats have similar trends and comparable accuracy as they represent similar weight values which differ only in the least significant fractional bit, thereby leading to similar neuron behavior and accuracy. These formats may have notable accuracy differences for some cases, such as after the retraining process, as shown by ❻. The possible reason is that these formats have different bit positions for the sign, integer, and fraction, thereby making the DRAM errors affect different weight bits and lead to different learning qualities during the respective fault-aware training.

In summary, our EnforceSNN maintains accuracy (i.e., no accuracy loss) as compared to the baseline with accurate DRAM when $BER \leq 10^{-3}$ across different datasets. Meanwhile, for higher BER values (i.e., $10^{-3} < BER \leq 10^{-2}$), our EnforceSNN still achieves higher accuracy than the baseline with accurate DRAM across different datasets. Therefore, these results show that *our EnforceSNN framework effectively improves the SNN error tolerance against DRAM errors with minimum retraining efforts*.

6.2. DRAM access energy savings and throughput improvements

Figure 12A shows the normalized energy consumption of the DRAM accesses for an inference (i.e., inferring one input sample) required by the baseline model and the EnforceSNN-improved model with accurate and approximate DRAM, across different V_{supply} values, precision levels, network sizes, and workloads. We observe that different network sizes show similar normalized DRAM access energy, hence we only show a single figure representing the experimental results for all network sizes. For accurate DRAM cases across different network sizes, the baseline model achieves 75% DRAM energy saving when it employs the quantization technique, while our EnforceSNN-improved model achieves 75.1% DRAM energy saving due to the quantization and the proposed DRAM mapping policy, as shown in ❼. Meanwhile, the difference in these DRAM energy savings comes from the DRAM mapping policy. That is, our EnforceSNN optimizes the DRAM energy-per-access by maximizing the row buffer hits and the multi-bank burst feature, thereby having fewer row buffer conflicts than the baseline which only exploits the single-bank burst feature. For the FP32 precision across different network sizes, employing the approximate DRAM in the baseline model reduces the DRAM energy savings by up to 39.2% as compared to employing the accurate DRAM. Meanwhile, employing the approximate DRAM in the EnforceSNN-improved model reduces the DRAM energy savings by up to 39.5% as compared to employing the accurate DRAM, as shown in ❽. These energy savings

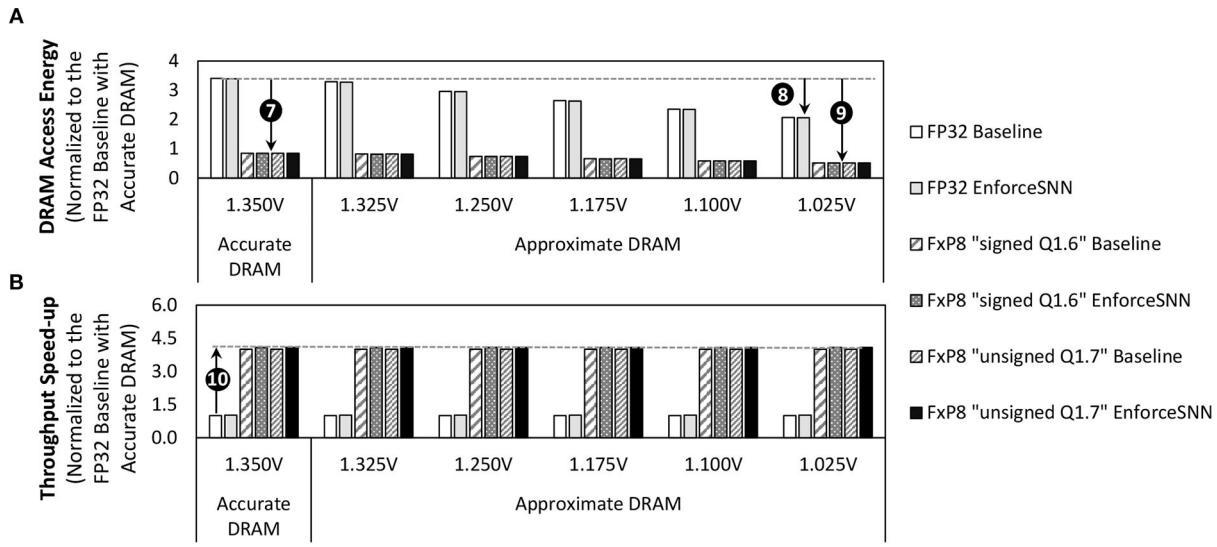

FIGURE 12
(A) The normalized DRAM access energy for an inference incurred by the baseline model and the EnforceSNN-improved model with accurate and approximate DRAM, and **(B)** the normalized speed-up of DRAM data throughput for an inference achieved by our EnforceSNN-improved model over the baseline model, across different V_{supply} values, different workloads (datasets), and different network sizes (N-900, N-1600, N-2500, and N-3600). These results are applicable for all network sizes. They are also applicable for both the MNIST and Fashion MNIST datasets, as these workloads have similar DRAM access energy, due to the same number of weights and number of DRAM accesses for an inference.

come from the reduced DRAM energy-per-access due to the reduction of operational V_{supply}. Moreover, the difference in energy savings between the baseline and our EnforceSNN also comes from the DRAM mapping policy. For the FxP8 precision (i.e., "signed $Q1.6$" and "unsigned $Q1.7$") across different network sizes, employing the approximate DRAM in the baseline model reduces the DRAM energy savings by up to 84.8% over employing the accurate one. Meanwhile, employing the approximate DRAM in the EnforceSNN-improved model reduces the DRAM energy savings by up to 84.9% over employing the accurate one, as shown in ❾. These energy savings come from the reduced weight precision and the reduced DRAM energy-per-access due to V_{supply} reduction. Moreover, the difference in energy savings between the baseline and our EnforceSNN also comes from the DRAM mapping policy. Furthermore, we also observe that our EnforceSNN-improved model obtains 4.1x throughput speed-up over the baseline model across different V_{supply} values, workloads, and network sizes; refer to ❿ in Figure 12B. It is achieved through (1) the quantization technique which reduces the number of DRAM accesses, and (2) our proposed DRAM mapping policy which optimizes the DRAM latency-per-access by maximizing the row buffer hits and the multi-bank burst features. The results also show that the "unsigned $Q1.7$" and "signed $Q1.6$" achieve comparable DRAM access energy savings and throughput improvements since they employ the same bitwidth of weights, thereby having similar DRAM access behavior.

In summary, the results in Figure 12 indicate that *our EnforceSNN framework substantially reduces the DRAM access energy by employing the reduced-voltage approximate DRAM and our efficient DRAM mapping policy, while effectively improving the DRAM data throughput mainly due to the quantization.*

6.3. Model selection under design trade-offs

Figures 13, 14 show the results of the accuracy-memory-energy trade-offs for the MNIST and Fashion MNIST datasets, respectively. In this evaluation, the quantized models consider the FxP8 precision in "signed $Q1.6$" format. For the given SNN model candidates, we observe that the models that incur small memory size typically employ FxP8 precision, as shown in Figure 13A for the MNIST and Figure 14A for the Fashion MNIST. Considering that the accuracy of the FxP8-based models is comparable to the FP32-based models, we narrow down the candidates to only the FxP8-based models.

To analyze the design trade-offs, we explore the impact of different μ and ε values on the rewards. For instance, if we consider that the accuracy should have a higher priority than the memory and energy consumption, we set μ and ε low (e.g., $\mu = 0$ and $\varepsilon = 0$). Meanwhile, if we consider that the memory should have a higher priority than the energy consumption, we set μ higher than ε (e.g., $\mu = 10$ and $\varepsilon = 0$). For both cases, the highest reward is achieved by the EnforceSNN-improved

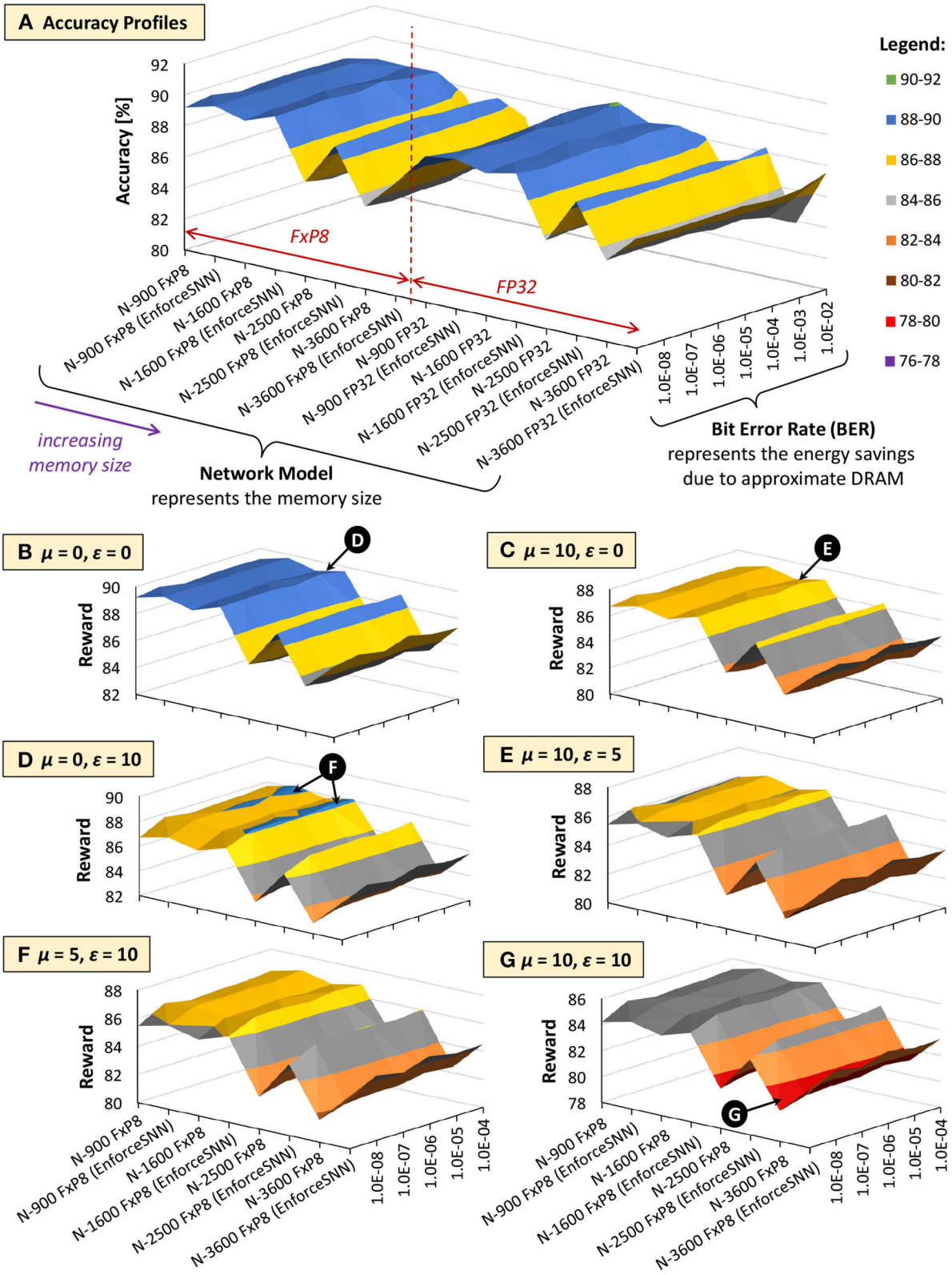

FIGURE 13
The trade-offs among accuracy, memory footprint, and energy consumption for the MNIST. **(A)** Accuracy profiles of SNN models. **(B–G)** Reward profiles of SNN models. The network sizes represent the memory sizes and the BER values represent the energy savings from approximate DRAM.

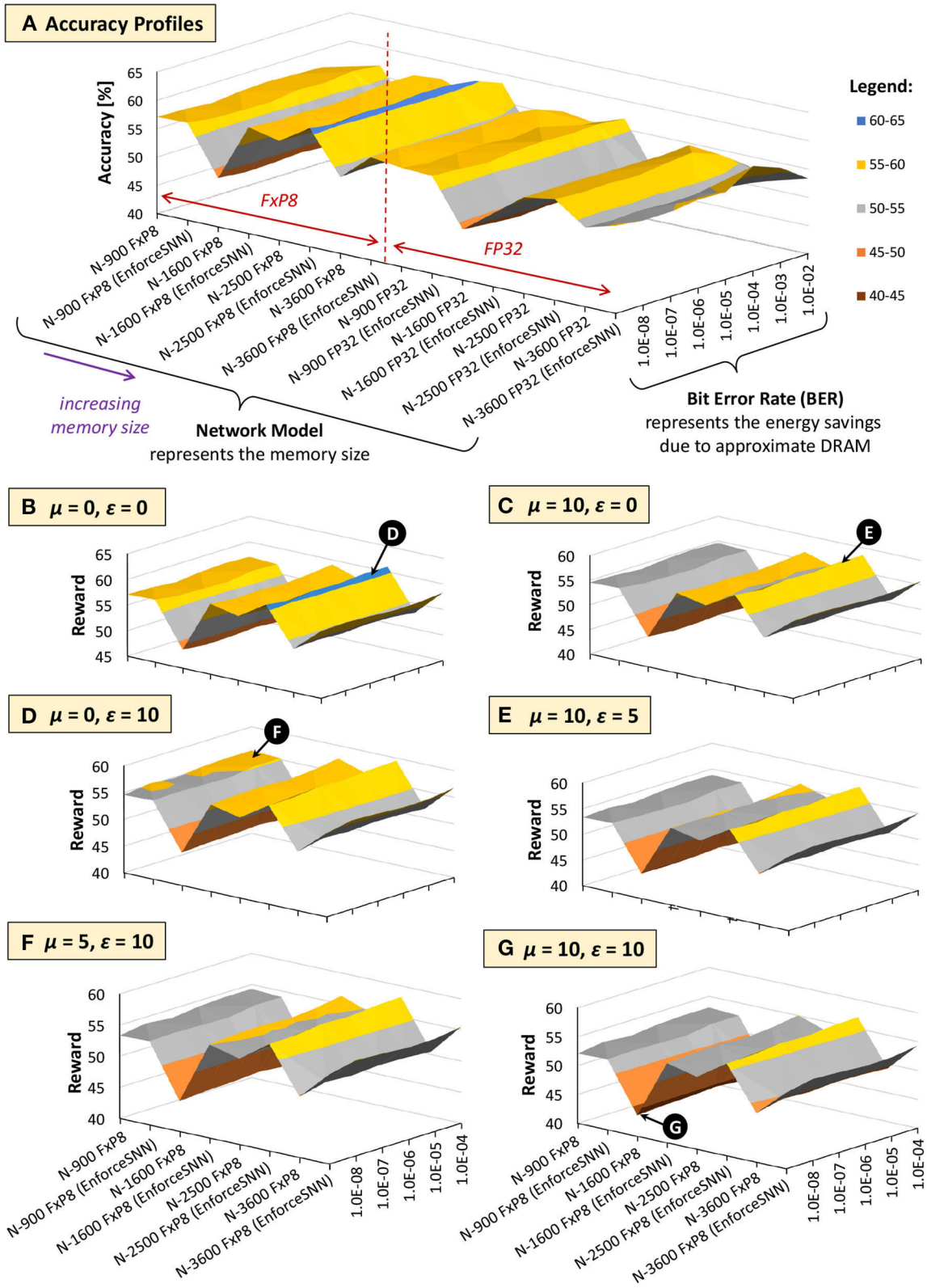

FIGURE 14
The trade-offs among accuracy, memory, and energy consumption for the Fashion MNIST. **(A)** Accuracy profiles of SNN models. **(B–G)** Reward profiles of SNN models. The network sizes represent the memory sizes, and the BER values represent the energy savings from approximate DRAM.

N-1600 FxP8 for the MNIST and the EnforceSNN-improved N-2500 FxP8 for the Fashion MNIST under 10^{-5} error rate; refer to ❶ for $\mu = 0$ and $\varepsilon = 0$ cases, and refer to ❷ for $\mu = 10$ and $\varepsilon = 0$ cases. The reason is that these models employ our efficient FAT technique to improve their error tolerance, thereby leading to high accuracy under a high error rate. We also observe that having μ higher than ε makes the high rewards shift toward smaller models, as shown by ❷ in Figure 13C. The reason is that a higher μ makes the small m_{norm} have a smaller impact on the reward reduction than the large m_{norm}, thereby maintaining the high reward values. If the energy consumption should have a higher priority than the memory footprint, we set μ lower than ε (e.g., $\mu = 0$ and $\varepsilon = 10$). The highest reward is achieved by the EnforceSNN-improved N-1600 FxP8 under 10^{-5} error rate for the MNIST and the EnforceSNN-improved N-2500 FxP8 under 10^{-4} error rate for the Fashion MNIST. In this case, we observe that high rewards are shifted toward models with smaller energy consumption (represented by higher BER); refer to ❸ in Figures 13D, 14D. The reason is that a higher ε makes the small E_{norm} have a smaller impact on the reward reduction than the large E_{norm}, thereby maintaining the high reward values. Furthermore, if the memory and energy consumption should have a higher priority than the accuracy, we set μ and ε high (e.g., $\mu = 10$ and $\varepsilon = 10$). The highest reward is achieved by the EnforceSNN-improved N-1600 FxP8 under 10^{-5} error rate for the MNIST and the EnforceSNN-improved N-2500 FxP8 under 10^{-4} error rate for the Fashion MNIST. In this case, high rewards are shifted toward models with smaller memory and energy consumption (represented by high BER), but their overall rewards decrease as the values of μ and ε increase; refer to ❹ in Figures 13E, 14E. The reason is that higher μ and ε jointly make the m_{norm} and E_{norm} decrease the reward. It means that if we want to significantly reduce the memory footprint and energy consumption, we have to accept more accuracy degradation.

In summary, the results in Figures 13, 14 show that *our EnforceSNN framework has an effective algorithm to trade off the accuracy, memory footprint, and energy consumption of the given SNN models*, thereby providing good applicability for diverse embedded applications with their respective constraints.

6.4. Optimization of the retraining costs

The conventional FAT for neural networks usually injects errors at an incremental rate during the retraining process from the minimum value to the maximum one for avoiding accuracy collapse (Koppula et al., 2019). Therefore, in this study, the conventional FAT considers $BER = \{10^{-8}, 10^{-7}, 10^{-6}, \ldots, 10^{-2}\}$, while our efficient FAT in EnforceSNN only considers $BER = \{10^{-4}, 10^{-3}, 10^{-2}\}$ in the retraining process.

Retraining Speed-ups: The conventional FAT with FxP8 (cFAT8) obtains speed-up over the one with FP32 (cFAT32) by up to 1.16x and 1.14x for the MNIST and the Fashion MNIST respectively, since the cFAT8 employs quantized weights, thereby leading to a faster error injection and learning process. Meanwhile, our efficient FAT with FP32 (eFAT32) obtains a 2.33x speed-up over the cFAT32, since our eFAT32 has fewer iterations of the retraining process. Furthermore, we also observe that our efficient FAT with FxP8 (eFAT8) obtains more speed-up, i.e., by up to 2.71x for the MNIST and 2.65x for the Fashion MNIST as shown by ❽ in Figure 15A, since our eFAT8 employs quantized weights in addition to fewer iterations of the retraining process.

Retraining Energy Savings: The cFAT8 achieves energy saving over the cFAT32 by up to 13.9% for the MNIST and 12% for the Fashion MNIST since the cFAT8 employs quantized weights which incur lower energy consumption during the error injection and learning process. Meanwhile, our eFAT32 achieves energy saving over the cFAT32 by 57.1%, as the eFAT32 performs fewer iterations of the retraining process as compared to the cFAT32. Our eFAT8 achieves further energy saving, i.e., by up to 63.1% for the MNIST and by up to 62.3% for the Fashion MNIST as shown by ❾ in Figure 15B, since it employs quantized weights in addition to fewer iterations of the retraining process, thereby leading to a higher energy saving.

Carbon Emission Reduction: The retraining process also poses additional challenges that correspond to environmental concerns, i.e., carbon emission. Recent studies have highlighted that the carbon emission from neural network training should be minimized to prevent the increasing rates of natural disasters (Strubell et al., 2019, 2020). To estimate the carbon emission of neural network training, the study of Strubell et al. (2019) proposed Equation 5 and Equation 6. In these equations, CO_2e denotes the estimated carbon (CO_2) emission during the training, which is a function of the total power during the training (p_t). Meanwhile, t is the training duration, p_c is the average power from all CPUs, p_r is the average power from all main memories (DRAMs), p_g is the average power from a GPU and g is the number of GPUs.

$$CO_2e = 0.954 \cdot p_t \tag{5}$$

$$p_t = \frac{1.58 \cdot t(p_c + p_r + g \cdot p_g)}{1000} \tag{6}$$

These equations indicate that if we assume p_c, p_r, p_g, and g are the same for different FAT techniques, then the difference will come from the training duration t. Therefore, our efficient FAT in EnforceSNN employs fewer iterations of the retraining process than the conventional FAT, thereby producing less carbon emission. Moreover, our EnforceSNN also reduces the operational power of the main memory (DRAM) through the reduced-voltage approximation approach, thereby further reducing the emission.

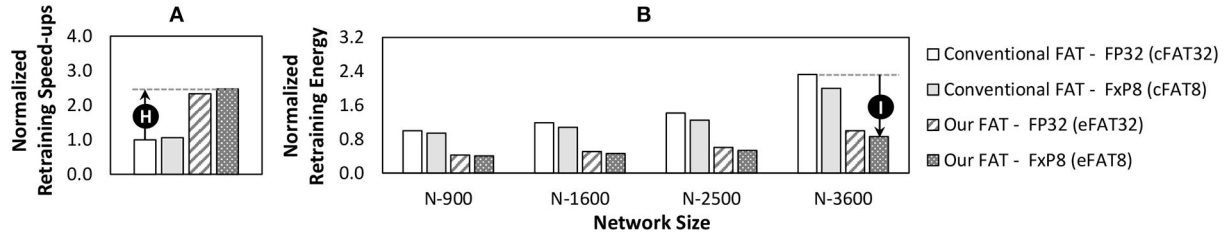

FIGURE 15
(A) The retraining speed-ups across different network sizes (i.e., N-900, N-1600, N-2500, and N-3600), and **(B)** the retraining energy for the MNIST, which are normalized to the conventional FAT with FP32 for a 900-neuron network. The results for the Fashion MNIST show similar trends to the MNIST since these workloads have similar DRAM access latency and energy due to the same number of weights and number of DRAM accesses for a training phase. Here, FxP8 represents both the "signed $Q1.6$" and "unsigned $Q1.7$" formats.

In summary, *our EnforceSNN framework effectively offers speed-up of retraining time, reduction of retraining energy, and less carbon emission than the conventional FAT technique*, thereby making it more friendly for our environments.

6.5. Further discussion

Previous studies that exploit the reduced-voltage DRAM concept mainly aim at improving the energy efficiency of mobile systems (Haj-Yahya et al., 2020), personal computing systems (Nabavi Larimi et al., 2021; Fabrício Filho et al., 2022), and server systems (David et al., 2011; Deng et al., 2011, 2012a,b; Nabavi Larimi et al., 2021). This concept is also employed for minimizing the energy consumption of deep neural networks (DNNs) (Koppula et al., 2019). Since SNNs have different data representation, computation models, and learning rules as compared to DNNs, our EnforceSNN provides a different framework with different techniques that are crafted specifically for improving the resilience and the energy efficiency of SNNs. Furthermore, the reduced-voltage DRAM is also used to generate noise (i.e., from DRAM errors) for obfuscating the intellectual property (IP) against security threats, such as IP stealing (Xu et al., 2020).

Our EnforceSNN framework can be put in the approximate computing field, especially in the context of the approximation for main memory through voltage scaling (Venkataramani et al., 2015; Mittal, 2016; Xu et al., 2016). Therefore, some of the techniques in our EnforceSNN are suitable for different domains outside SNNs: (1) DRAM voltage reduction for optimizing the DRAM access energy, (2) quantization for reducing the memory footprint, and (3) error-aware DRAM data mapping policy for minimizing the negative impact of DRAM errors on the data. These techniques are applicable for error-tolerant applications, such as image/video processing (e.g., data compression) and data analytic applications (e.g., data clustering).

7. Conclusion

We propose a novel EnforceSNN framework to achieve resilient and energy-efficient SNN inference considering reduced-voltage-based approximate DRAM, through weight quantization, error-aware DRAM mapping, SNN error-tolerance analysis, efficient error-aware SNN training, and effective SNN model selection. Our EnforceSNN achieves no accuracy loss for BER $\leq 10^{-3}$ with minimum retraining costs as compared to the baseline SNN with accurate DRAM while achieving up to 84.9% of DRAM energy saving and up to 4.1x speed-up of DRAM data throughput. In this manner, our study may enable efficient SNN inference for energy–constrained embedded devices like the Edge-AI.

Data availability statement

Publicly available datasets were used in this study. These datasets can be found at the following sites: http://yann.lecun.com/exdb/mnist/ (MNIST), http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/ (Fashion MNIST).

Author contributions

RP, MH, and MS contributed to the conception of the framework. RP developed and implemented the framework, and wrote the first draft of the manuscript. RP and MH organized the experiments. MS supervised the research work. All authors contributed to manuscript revision, read, and approved the submitted version.

Funding

This research is partly supported by the ASPIRE AARE Grant (S1561) on Towards Extreme

Energy Efficiency through Cross-Layer Approximate Computing.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., et al. (2015). TrueNorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput. Aided Design Integr. Circ. Syst.* 34, 1537–1557. doi: 10.1109/TCAD.2015.2474396

Cao, K., Liu, Y., Meng, G., and Sun, Q. (2020). An overview on edge computing research. *IEEE Access* 8, 85714–85728. doi: 10.1109/ACCESS.2020.2991734

Chandrasekar, K. (2014). *High-level power estimation and optimization of DRAMs* (Ph,D, thesis), TU Delft.

Chang, K. K., Yağlıkçı, A. G., Ghose, S., Agrawal, A., Chatterjee, N., Kashyap, A., et al. (2017). "Understanding reduced-voltage operation in modern DRAM devices: experimental characterization, analysis, and mechanisms," in *Proceedings of ACM on Measurements and Analysis of Computing Systems* (New York, NY).

Chattopadhyay, A., Prakash, A., and Shafique, M. (2017). Secure cyber-physical systems: Current trends, tools and open research problems. *Design Autom. Test Eur. Conf. Exhibit.* 2017, 1104–1109. doi: 10.23919/DATE.2017.7927154

Chen, J., and Ran, X. (2019). Deep learning with edge computing: a review. *Proc. IEEE* 107, 1655–1674. doi: 10.1109/JPROC.2019.2921977

David, H., Fallin, C., Gorbatov, E., Hanebutte, U. R., and Mutlu, O. (2011). "Memory power management via dynamic voltage/frequency scaling," in *The 8th ACM International Conference on Autonomic Computing* (Karlsruhe), 31–40.

Deng, Q., Meisner, D., Bhattacharjee, A., Wenisch, T. F., and Bianchini, R. (2012a). "CoScale: coordinating CPU and memory system DVFS in server systems," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture* (Vancouver, BC: IEEE), 143–154.

Deng, Q., Meisner, D., Bhattacharjee, A., Wenisch, T. F., and Bianchini, R. (2012b). "MultiScale: memory system DVFS with multiple memory controllers," in *The 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, 297–302.

Deng, Q., Meisner, D., Ramos, L., Wenisch, T. F., and Bianchini, R. (2011). "MemScale: active low-power modes for main memory," in *The 16th International Conference on Architectural Support for Programming Languages and Operating Systems*, 225–238.

Diehl, P., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9, 99. doi: 10.3389/fncom.2015.00099

Fabrício Filho, J., Felzmann, I., and Wanner, L. (2022). SmartApprox: learning-based configuration of approximate memories foräenergy–efficient execution. *Sustain. Comput.* 34, 100701. doi: 10.1016/j.suscom.2022.100701

Frenkel, C., Lefebvre, M., Legat, J., and Bol, D. (2019a). A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos. *IEEE Trans. Biomed. Circ. Syst.* 13, 145–158. doi: 10.1109/TBCAS.2018.2880425

Frenkel, C., Legat, J.-D., and Bol, D. (2019b). MorphIC: a 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning. *IEEE Trans. Biomed. Circ. Syst.* 13, 999–1010. doi: 10.1109/TBCAS.2019.2928793

Gautrais, J., and Thorpe, S. (1998). Rate coding versus temporal order coding: a theoretical approach. *Biosystems* 48, 57–65. doi: 10.1016/S0303-2647(98)00050-1

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference. *arXiv [Preprint] arXiv:*2103.13630. doi: 10.1201/9781003162810-13

Ghose, S., Li, T., Hajinazar, N., Cali, D. S., and Mutlu, O. (2019). "Demystifying complex workload-DRAM interactions: an experimental study," in *Proceedings of ACM on Measurements and Analysis of Computing Systems* (New York, NY).

Griffor, E., Greer, C., Wollman, D., and Burns, M. (2017). *Framework for Cyber-Physical Systems: Vol. 1, Overview*. Gaithersburg, MD: Special Publication (NIST SP), National Institute of Standards and Technology.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. (2015). "Deep learning with limited numerical precision," in *32nd International Conference on Machine Learning* (Lille), 1737–1746.

Haj-Yahya, J., Alser, M., Kim, J., Yağlikçi, A. G., Vijaykumar, N., Rotem, E., et al. (2020). "SysScale: exploiting multi-domain dynamic voltage and frequency scaling for energy efficient mobile processors," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture* (Valencia), 227–240.

Hazan, H., Saunders, D., Sanghavi, D. T., Siegelmann, H., and Kozma, R. (2018). "Unsupervised learning with self-organizing spiking neural networks," in *2018 International Joint Conference on Neural Networks* (Lausanne), 1–6.

Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., et al. (2018). Bindsnet: a machine learning-oriented spiking neural networks library in python. *Front. Neuroinform.* 12, 89. doi: 10.3389/fninf.2018.00089

Hopkins, M., Mikaitis, M., Lester, D. R., and Furber, S. (2020). Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations. *Philos. Trans. R. Soc. A* 378, 20190052. doi: 10.1098/rsta.2019.0052

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* 15, 1063–1070. doi: 10.1109/TNN.2004.832719

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., et al. (2018). "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *The IEEE Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 2704–2713.

Kayser, C., Montemurro, M. A., Logothetis, N. K., and Panzeri, S. (2009). Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns. *Neuron* 61, 597–608. doi: 10.1016/j.neuron.2009.01.008

Kim, J., Patel, M., Hassan, H., and Mutlu, O. (2018). 'Solar-DRAM: reducing DRAM access latency by exploiting the variation in local bitlines," in *2018 IEEE 36th International Conference on Computer Design* (Orlando, FL: IEEE), 282–291.

Kim, Y., Seshadri, V., Lee, D., Liu, J., and Mutlu, O. (2012). "A case for exploiting subarray-level parallelism (SALP) in DRAM," in *2012 39th Annual International Symposium on Computer Architecture* (Portland, OR), 368–379.

Koppula, S., Orosa, L., Yağlıkçı, A. G., Azizi, R., Shahroodi, T., Kanellopoulos, K., et al. (2019). "EDEN: enabling energy-efficient, high-performance deep neural network inference using approximate DRAM," in *52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH), 166–181.

Kriebel, F., Rehman, S., Hanif, M. A., Khalid, F., and Shafique, M. (2018). "Robustness for smart cyber physical systems and internet-of-things: from adaptive robustness methods to reliability and security for machine learning," in *2018 IEEE Computer Society Annual Symposium on VLSI* (Hong Kong: IEEE), 581–586.

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: a whitepaper. *arXiv [Preprint] arXiv:*1806.08342.

Krithivasan, S., Sen, S., Venkataramani, S., and Raghunathan, A. (2019). "Dynamic spike bundling for energy-efficient spiking neural networks," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design* (Lausanne: IEEE), 1–6.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Liu, F., Tang, G., Li, Y., Cai, Z., Zhang, X., and Zhou, T. (2019). A survey on edge computing systems and tools. *Proc. IEEE* 107, 1537–1562. doi: 10.1109/JPROC.2019.2920341

Micikevicius, P., Narang, S., Alben, J., Diamos, G. F., Elsen, E., García, D., et al. (2018). "Mixed precision training," in *6th International Conference on Learning Representations* (Vancouver, BC).

Mittal, S. (2016). A survey of techniques for approximate computing. *ACM Comput. Survey* 48, 1–33. doi: 10.1145/2893356

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., and Masquelier, T. (2019). Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Front. Neurosci.* 13, 625. doi: 10.3389/fnins.2019.00625

Nabavi Larimi, S. S., Salami, B., Unsal, O. S., Kestelman, A. C., Sarbazi-Azad, H., and Mutlu, O. (2021). "Understanding power consumption and reliability of high-bandwidth memory with voltage underscaling," in *2021 Design, Automation and Test in Europe Conference and Exhibition* (Grenoble) 517–522.

Olgun, A., Luna, J. G., Kanellopoulos, K., Salami, B., Hassan, H., Ergin, O., et al. (2021). PiDRAM: a holistic end-to-end FPGA-based framework for processing-in-DRAM. *arXiv [Preprint] arXiv*:2111.00082.

Park, S., Kim, S., Choe, H., and Yoon, S. (2019). "Fast and efficient information transmission with burst spikes in deep spiking neural networks," in *2019 56th Annual Design Automation Conference* (Las Vegas, NV).

Park, S., Kim, S., Na, B., and Yoon, S. (2020). "T2FSNN: deep spiking neural networks with time-to-first-spike coding," in *57th ACM/IEEE Design Automation Conference* (San Francisco, CA: IEEE), 1–6.

Pfeiffer, M., and Pfeil, T. (2018). Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci*. 12, 774. doi: 10.3389/fnins.2018.00774

Putra, R. V. W., Hanif, M. A., and Shafique, M. (2020). "DRMap: a generic DRAM data mapping policy for energy-efficient processing of convolutional neural networks," in *2020 57th ACM/IEEE Design Automation Conference* (San Francisco, CA: IEEE), 1–6.

Putra, R. V. W., Hanif, M. A., and Shafique, M. (2021a). "ReSpawn: energy-efficient fault-tolerance for spiking neural networks considering unreliable memories," in *2021 IEEE/ACM International Conference On Computer Aided Design* (Munich), 1–9.

Putra, R. V. W., Hanif, M. A., and Shafique, M. (2021b). ROMANet: Fine-grained reuse-driven off-chip memory access management and data organization for deep neural network accelerators. *IEEE Trans. Very Large Scale Integr. Syst*. 29, 702–715. doi: 10.1109/TVLSI.2021.3060509

Putra, R. V. W., Hanif, M. A., and Shafique, M. (2022). SoftSNN: low-cost fault tolerance for spiking neural network accelerators under soft errors. *arXiv [Preprint] arXiv*:2203.05523. doi: 10.48550/arXiv.2203.05523

Putra, R. V. W., and Shafique, M. (2020). FSpiNN: An optimization framework for memory-efficient and energy-efficient spiking neural networks. *IEEE Trans. Comput. Aided Design Integr. Circ. Syst*. 39, 3601–3613. doi: 10.1109/TCAD.2020.3013049

Putra, R. V. W., and Shafique, M. (2021a). "Q-SpiNN: a framework for quantizing spiking neural networks," in *2021 International Joint Conference on Neural Networks* (Shenzhen), 1–8.

Putra, R. V. W., and Shafique, M. (2021b). "SpikeDyn: a framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments," in *2021 58th ACM/IEEE Design Automation Conference* (San Francisco, CA: IEEE), 1057–1062.

Putra, R. V. W., and Shafique, M. (2022a). lpSpikeCon: enabling low-precision spiking neural network processing for efficient unsupervised continual learning on autonomous agents. *arXiv [Preprint] arXiv*:2205.12295. doi: 10.48550/arXiv.2205.12295

Putra, R. V. W., and Shafique, M. (2022b). tinySNN: towards memory- and energy-efficient spiking neural networks. *arXiv [Preprint] arXiv*:2206.08656. doi: 10.48550/arXiv.2206.08656

Rahimi Azghadi, M., Iannella, N., Al-Sarawi, S. F., Indiveri, G., and Abbott, D. (2014). Spike-based synaptic plasticity in silicon: design, implementation, application, and challenges. *Proc. IEEE* 102, 717–737. doi: 10.1109/JPROC.2014.2314454

Rathi, N., Panda, P., and Roy, K. (2019). STDP-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. *IEEE Trans. Comput. -Aided Design Integr. Circ. Syst.* 38, 668–677. doi: 10.1109/TCAD.2018.2819366

Roy, A., Venkataramani, S., Gala, N., Sen, S., Veezhinathan, K., and Raghunathan, A. (2017). "A programmable event-driven architecture for evaluating spiking neural networks," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design* (Taipei: IEEE), 1–6.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer* 50, 30–39. doi: 10.1109/MC.2017.9

Saunders, D. J., Siegelmann, H. T., Kozma, R., and Ruszink'o, M. (2018). "STDP learning of image patches with convolutional spiking neural networks," in *2018 International Joint Conference on Neural Networks* (Rio de Janeiro), 1–7.

Sen, S., Venkataramani, S., and Raghunathan, A. (2017). "Approximate computing for spiking neural networks," in *Design, Automation Test in Europe Conference Exhibition* (Lausanne), 193–198.

Shafique, M., Khalid, F., and Rehman, S. (2018). "Intelligent security measures for smart cyber physical systems," in *2018 21st Euromicro Conference on Digital System Design*, 280–287.

Shafique, M., Marchisio, A., Putra, R. V. W., and Hanif, M. A. (2021). "Towards energy-efficient and secure edge AI: a cross-layer framework ICCAD special session paper," in *2021 IEEE/ACM International Conference On Computer Aided Design* (Prague; Munich: IEEE), 1–9.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: vision and challenges. *IEEE Internet Things J.* 3, 637–646. doi: 10.1109/JIOT.2016.2579198

Strubell, E., Ganesh, A., and McCallum, A. (2019). "Energy and policy considerations for deep learning in NLP," in *Proceedings of 57th Annual Meeting of the Association for Computational Linguistics* (Florence), 3645–3650.

Strubell, E., Ganesh, A., and McCallum, A. (2020). Energy and policy considerations for modern deep learning research. *Proc. AAAI Conf. Artif. Intell.* 34, 13693–13696. doi: 10.1609/aaai.v34i09.7123

Sze, V., Chen, Y., Yang, T., and Emer, J. S. (2017). Efficient processing of deep neural networks: a tutorial and survey. *Proc. IEEE* 105, 2295–2329. doi: 10.1109/JPROC.2017.2761740

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks* 111, 47–63. doi: 10.1016/j.neunet.2018.12.002

Thorpe, S., and Gautrais, J. (1998). "Rank order coding," in *Computational Neuroscience* (New York, NY), 113–118.

van Baalen, M., Kahne, B., Mahurin, E., Kuzmin, A., Skliar, A., Nagel, M., et al. (2022). "Simulated quantization, real power savings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (New Orleans, LA), 2757–2761.

Venkataramani, S., Chakradhar, S. T., Roy, K., and Raghunathan, A. (2015). "Approximate computing and the quest for computing efficiency," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference* (San Francisco, CA), 1–6.

Xu, Q., Mytkowicz, T., and Kim, N. S. (2016). Approximate computing: a survey. *IEEE Design Test* 33, 8–22. doi: 10.1109/MDAT.2015.2505723

Xu, Q., Tanvir Arafin, M., and Qu, G. (2020). "MIDAS: Model inversion defenses using an approximate memory system," in *2020 Asian Hardware Oriented Security and Trust Symposium* (Kolkata), 1–4.

Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., et al. (2018). A survey on the edge computing for the internet of things. *IEEE Access* 6, 6900–6919. doi: 10.1109/ACCESS.2017.2778504