IpSpikeCon: Enabling Low-Precision Spiking Neural Network Processing for Efficient Unsupervised Continual Learning on Autonomous Agents

Rachmad Vidya Wicaksana Putra*, Muhammad Shafique†

*Institute of Computer Engineering, Technische Universität Wien (TU Wien), Vienna, Austria
†Division of Engineering, New York University Abu Dhabi (NYUAD), Abu Dhabi, United Arab Emirates

Email: rachmad.putra@tuwien.ac.at, muhammad.shafique@nyu.edu

Abstract—Recent advances have shown that Spiking Neural Network (SNN)-based systems can efficiently perform unsupervised continual learning due to their bio-plausible learning rule, e.g., Spike-Timing-Dependent Plasticity (STDP). Such learning capabilities are especially beneficial for use cases like autonomous agents (e.g., robots and UAVs) that need to continuously adapt to dynamically changing scenarios/environments, where new data gathered directly from the environment may have novel features that should be learned online. Current state-of-the-art works employ high-precision weights (i.e., 32 bit) for both training and inference phases, which pose high memory and energy costs thereby hindering efficient embedded implementations of such systems for battery-driven mobile autonomous systems. On the other hand, precision reduction may jeopardize the quality of unsupervised continual learning due to information loss. Towards this, we propose IpSpikeCon, a novel methodology to enable low-precision SNN processing for efficient unsupervised continual learning on resource-constrained autonomous agents/systems. Our IpSpikeCon methodology employs the following key steps: (1) analyzing the impacts of training the SNN model under unsupervised continual learning settings with reduced weight precision on the inference accuracy; (2) leveraging this study to identify SNN parameters that have a significant impact on the inference accuracy; and (3) developing an algorithm for searching the respective SNN parameter values that improve the quality of unsupervised continual learning. The experimental results show that our IpSpikeCon can reduce weight memory of the SNN model by 8x (i.e., by judiciously employing 4-bit weights) for performing online training with unsupervised continual learning and achieve no accuracy loss in the inference phase, as compared to the baseline model with 32-bit weights across different network sizes.

Index Terms—Spiking neural networks, SNNs, unsupervised learning, continual learning, memory efficiency, energy efficiency, autonomous agents, embedded systems.

I. INTRODUCTION

With the advances of neuromorphic computing, SNN-based systems have shown potential for having efficient learning capabilities due to their bio-plausible learning rules, like the Spike-Timing-Dependent Plasticity (STDP) [1]–[8]. However, the knowledge learned from the offline training process may become obsolete over time and in unpredictable operational conditions, which leads to low accuracy at run time. This is crucial when SNN systems are deployed in the dynamically changing scenarios/environments, i.e., where new data collected directly from the operational environments have novel information/features that should be learned online [9] [10]; see Fig. 1. Moreover, this data is typically unlabeled and may not be proportionally distributed, thereby making it difficult for the SNN systems to learn different tasks/classes [11]. To address these issues, recent works have proposed strategies for enabling the unsupervised continual learning in SNNs [9]–[12], which are beneficial for use cases like autonomous agents (e.g., robots and UAVs) that need to continuously adapt to dynamically changing environments [10]. However, autonomous agents typically have tight constraints (e.g., available memory), thereby requiring an efficient embedded implementation of SNN systems.

Targeted Problem: If and how can we efficiently implement SNN systems with unsupervised continual learning capabilities on autonomous agents with tight memory constraints, while maintaining the inference accuracy close to that of the baseline implementation. An efficient solution to this problem will enable highly memory- and power/energy-efficient autonomous agents that are adaptive to unpredictable dynamic environments.

A. The State-of-the-Art and Their Limitations

The research for unsupervised continual learning in SNN systems is still at an early stage. Therefore, the state-of-the-art works still focus on improving the quality of learning, while considering relatively simple datasets (e.g., MNIST) [9]–[12]. These works employ high-precision weights (i.e., 32 bits) for both training and inference phases to achieve high accuracy, but incur high memory and energy costs, thereby hindering the implementation of such SNN systems for battery-powered autonomous agents. Towards this, quantization is a potential technique for efficiently reducing the memory footprint of SNNs, and hence the energy consumption [8] [13]. However, the impacts of weight quantization on the accuracy of unsupervised continual learning in SNN systems have not been explored yet. To show the limitations of state-of-the-art and the targeted problem, we perform an experimental case study in Section I-B.

B. Case Study and Scientific Challenges

For the case study, we perform experiments that provide dynamic scenarios to the network by feeding consecutive tasks/
The solution should employ simple yet effective enhancements to compensate for the information loss due to weight quantization, thereby enabling energy-efficient learning.

C. Our Novel Contributions

To address the above challenges, we propose IpSpikeCon, a novel methodology that enables low-precision SNNs, which supports continual learning on autonomous agents/systems. To the best of our knowledge, this work is the first effort that aims at reducing the weight precision of SNNs, while maintaining the quality of unsupervised continual learning under dynamic scenarios. Following are the key steps of our IpSpikeCon methodology (see an overview in Fig. 3).

- Analyzing the characteristics of SNN accuracy profiles of each given task under different quantization levels, when the given SNN employs unsupervised continual learning.
- Identifying the SNN parameters that have significant impact on the accuracy. It leverages the accuracy analysis to determine SNN parameters and their adjustment rules to get better neuronal dynamics for unsupervised continual learning, and hence accuracy.
- Devising an algorithm for determining parameter values that effectively improve the learning quality. It leverages the parameter adjustment rules for guiding the algorithm to refine the parameter values for achieving high accuracy in dynamic scenarios/environments.

Key Results: We evaluate our IpSpikeCon methodology using a Python-based framework [14] on a multi-GPU machine. The experimental results show that our IpSpikeCon improves the quality of unsupervised continual learning for a quantized SNN through effective parameter adjustments, thereby achieving no accuracy loss in the inference while reducing the weight memory (e.g., by 8x when the SNN employs 4-bit weights), as compared to the non-quantized SNN (i.e., with 32-bit weights).

Fig. 2. (a) The SNN architecture that supports unsupervised continual learning, i.e., a single-layer fully-connected network. (b) Weight memory of a 400-neuron network with different learning conditions: No Unsupervised Continual Learning (No UCL) with 32-bit weights; Baseline UCL with 32-bit weights, adapted from [10]; and Baseline UCL with 4-bit weights, using quantization. (c) Accuracy of a 400-neuron network with different learning conditions. In this work, we consider the UCL algorithm from the work of [10].

Our experiments consider the network architecture shown in Fig. 2(a) and different learning conditions: (1) No Unsupervised Continual Learning (No UCL); (2) Baseline UCL with 32-bit weights, adapted from [10]; and (3) Baseline UCL with 4-bit weights. Further details of the experimental setup are presented in Section IV. The experimental results are shown in Fig. 2(b)-(c), from which we make the following key observations.

- The unsupervised continual learning improves the accuracy under dynamic scenarios, due to its carefully crafted weight potentiation/depression strategy to learn new features, while retaining old yet important ones.
- Reduction of weight precision can significantly save the SNN weight memory, e.g., reducing precision from 32-bit to 4-bit weights enables 8x weight memory saving. However, it may degrade the quality of unsupervised continual learning due to knowledge/information loss.
- A network with 4-bit weights and baseline learning may achieve higher accuracy than a network with 32-bit weights but no continual learning, thereby showing the potential of memory reduction for a network under dynamic scenarios.

These observations highlight the following key challenges to devise an efficient solution for the targeted problem.

- Quantization should be performed judiciously to remove non-significant information in each weight, hence retaining most of the important information and maintaining the learning quality (i.e., high accuracy).

A. Spiking Neural Networks (SNNs)

SNNs have the potential for energy-efficient processing in both the training and inference phases due to their highly bio-plausible computational model, i.e., spike-based information and computation [13]–[18]. Each SNN model is formed by several design components, i.e., network architecture, neuron and synapse models, spike coding, as well as learning rule [8].

In this work, we employ the network architecture/topology of Fig. 2(a), as it has been widely used for SNNs with
unsupervised continual learning capabilities [9]–[11]. This network is trained so that each excitatory neuron can recognize a specific task/class. For spike/information coding, several coding schemes have been proposed in the literature, such as temporal, rank-order, rate, phase, and burst coding [19]–[22]. Here, we employ the rate coding to convert information into spike trains, since it can achieve high accuracy in unsupervised learning settings [23]. For the neuronal model, we employ the Leaky Integrate-and-Fire (LIF) model, as it incurs the lowest computational complexity while providing highly bio-plausible neuronal dynamics, as compared to other neuronal models [24]. The neuronal dynamics of the LIF model are shown in Fig. 4 and explained in the following.

- LIF neuron increases the membrane potential ($V_{\text{mem}}$) each time there is an incoming spike, and otherwise, the $V_{\text{mem}}$ decreases exponentially with the rate of $V_{\text{decay}}$.
- If the $V_{\text{mem}}$ reaches the membrane threshold potential ($V_{\text{th}}$), the LIF neuron generates a spike, then the $V_{\text{mem}}$ goes back to the reset potential ($V_{\text{reset}}$), and the neuron state enters the refractory period ($T_{\text{ref}}$).
- To evenly distribute the spiking activity across all excitatory neurons (i.e., homeostasis), the membrane threshold potential is defined as $V_{\text{th}} = V_{\text{th}} + \theta$ (so-called adaptive membrane threshold potential) [25] with $\theta$ denotes the adaptation potential. The $\theta$ is added to $V_{\text{th}}$ each time the neuron generates a spike, and otherwise, $V_{\text{th}}$ decreases with the rate of $\theta_{\text{decay}}$. Meanwhile, the synapse is modeled by the conductance and weight ($w$). The conductance is increased by $w$ when there is learning activity (i.e., weight potentiation). Depending upon the learning rule, the $w$ may be decreased with the rate of weight decay ($w_{\text{decay}}$) if there is no learning activity (i.e., weight depression). For the learning rule, we employ the pair-based STDP learning, which is adapted from [10].

![Fig. 4. Overview of the neuronal dynamics of a LIF model.](image)

**III. THE lpSPIKECON METHODOLOGY**

**A. Overview**

Our lpSpikeCon methodology employs the following key steps (see an overview in Fig. 5).

- Analyzing the SNN accuracy profiles (Section III-B) by performing the following mechanisms.
  - Quantizing the weights of a given network.
  - Training the network with quantized weights using the unsupervised continual learning under dynamic scenarios.
  - Observing the accuracy profiles of each given task/class under different quantization levels.
- Identifying the SNN parameters and their adjustment rules for unsupervised continual learning (Section III-C) by performing the following mechanisms.
  - Leveraging the accuracy analysis to select SNN parameters that have a significant impact on accuracy.
  - Devising the adjustment rules for the selected parameters to derive better neural dynamics (e.g., membrane threshold potential) for unsupervised continual learning.
- Devising an algorithm for refining SNN parameter values for unsupervised continual learning (Section III-D). It is performed by leveraging the adjustment rules to gradually increase/decrease parameter values that improve the learning quality under quantized weights.

The lpSpikeCon-enhanced SNN systems are then scheduled to update their offline-trained knowledge regularly at run time, e.g., based on the user-defined duration of an inference mode. The update is performed through online training using data gathered from the operational environments. After completing the training mode, the systems are back to the inference mode. In this manner, the SNN systems are expected to adapt better to diverse operational environments than the offline-trained ones.

**B. Quantization**

Quantization is a prominent technique to reduce the size of an SNN model (i.e., weight memory) that incurs a relatively low overhead, since it only needs to reduce the data precision [13]. It can be represented in a fixed-point format ($Q_i, f$), which consists of 1 sign bit, $i$ integer bits, $f$ fractional bits, and follows the 2’s complement format. Therefore, the $Q_i, f$ format has a range of representable values of $[-2^i, 2^i - 2^{-f}]$ with a precision of $e = 2^{-f}$. In the quantization process, a rounding scheme is required, and we consider the simple and widely used one, i.e., truncation [26] [27]. Truncation simply keeps $f$ bits and removes the remaining bits from the fractional part. The output of fixed-point quantization for a given real number $x$ with configuration $Q_i, f$ is defined as $T(x, Q_i, f) = \lfloor x \rfloor$. To simulate the unsupervised continual learning under quantized weights, we perform quantization during the training phase using the simulated quantization approach [28]; see Fig. 5.

![Fig. 5. Overview of the quantization scheme.](image)
Afterwards, we observe the profiles of inference accuracy for each given task/class under different levels of weight quantization, to understand the sources of accuracy degradation and get insights on how to address it.

**Algorithm 1** Training and testing under dynamic scenarios

**INPUT:** (1) SNN model \((\text{model}_{\text{in}})\); (2) Dataset \((D)\): dataset for class-i \((D[i])\), training set for class-i \((D[i].\text{train\_set})\), testing set for class-i \((D[i].\text{test\_set})\).

**OUTPUT:** (1) Trained SNN model \((\text{model}_{\text{out}})\); (2) Accuracy (acc)

**BEGIN**

**Initialization:**
1: \(\text{model}_{\text{out}} = \text{model}_{\text{in}}\);
2: \(\text{task} = \text{class}(D)\);
3: for \(i \in \text{task}\) do
4: \(\text{model}_{\text{out}} \leftarrow \text{train}(\text{model}_{\text{out}}, D[i].\text{train\_set})\);
5: for \(k = 0\) to \(i\) do
6: \(\text{acc}[i,k] = \text{test}(\text{model}_{\text{out}}, D[k].\text{test\_set})\);
7: end for
8: end for
9: return \(\text{model}_{\text{out}}\);

**END**

The experimental results are presented in Fig. 7 from which we make the following key observations.

- In general, reduced weight precision degrades the accuracy of unsupervised continual learning due to information loss. The accuracy degradation is noticeable especially for tasks that are learned at the later training sequence, as shown by labels (3), (2), and (1) in Fig. 7.

- Lower weight precision also leads to lower accuracy for more tasks. For instance, a model with 4-bit weights suffers from very low accuracy \((\leq 20\%)\) on four tasks, i.e., digit-6, digit-7, digit-8, and digit-9 (see label (3) in Fig. 7), while a model with 8-bit weights suffers from very low accuracy only on task digit-9 (see label (1) in Fig. 7).

These observations lead to several insights for devising an efficient solution for the targeted problem, as follows.

- Reduced weight precision has a less memory capacity for storing unique information from new tasks, thereby making the quantized SNN model difficult to recognize novel features from new tasks.

- Different SNN models with different levels of weight precision have different accuracy profiles. Therefore, such models require specialized settings for achieving high accuracy.

The above discussion indicates that, the potential solution is by employing parameter adjustments that provide more available memories for storing novel information from new tasks, while considering specialized settings for different levels of weight precision. In this manner, costly additional components and/or operations can be avoided, which leads to efficient learning.

C. Identifying SNN Parameters and Their Adjustment Rules

Analysis in Section III-B suggests that the efficient solution is by employing parameter adjustments for the given quantized SNN model. Therefore, it is important to identify SNN parameters that have a significant impact on accuracy, and their effective adjustment rules for improving learning quality of the quantized model.

**SNN Parameters:** To identify SNN parameters that should be adjusted for better learning quality, we leverage the analysis from Section III-B. The analysis shows that the reduced weight precision makes the model difficult to learn tasks that appear at the later training sequence. This means that the weight bits are strongly associated with the previously learned tasks, and not flexible enough to change their context to another task. To address this, we adjust the neuronal dynamics so that the synaptic plasticity becomes more flexible for learning new tasks, especially when low weight precision is employed. To do this, we adjust two parameters to change the flexibility of neuronal dynamics for learning activity: adaptive membrane threshold potential \((V_{\text{th}})\) and weight decay rate \((w_{\text{decay}})\). This selection is based on the following reasons.

- \(V_{\text{th}}\) determines how far \(V_{\text{mem}}\) should be increased to generate a spike, which then triggers weight potentiation for learning. The distance between \(V_{\text{th}}\) and \(V_{\text{mem}}\) is inversely proportional to the flexibility for learning activity.

- \(w_{\text{decay}}\) determines how fast each synaptic weight is depressed for removing the learned information, and providing available memory for storing novel information from new tasks.

**Adjustment Rules:** To properly adjust the selected SNN parameters (i.e., \(V_{\text{th}}\) and \(w_{\text{decay}}\)) for achieving better learning quality of a quantized SNN model, we propose the following parameter adjustment rules.

- In the non-quantized model, \(V_{\text{th}}\) is set with a value that prevents catastrophic forgetting\(^1\). Therefore, to make \(V_{\text{th}}\)

\(^1\)Catastrophic forgetting means that the system learns information from new data, but quickly forgets the previously learned ones [29–31]
an algorithm that leverages our adjustment rules to refine the selected SNN parameter values ($V_{th}$ and $w_{\text{decay}}$) for achieving better learning quality, especially under dynamic scenarios. This algorithm is developed based on the following ideas, and the detailed steps are provided in Alg. 2:

- The range of values for each selected SNN parameter ($V_{th}$ or $w_{\text{decay}}$) should be defined for guiding the design space exploration. The lower-bound value for $V_{th}$ is $V_{th}^l$, while the upper-bound value for $w_{\text{decay}}$ is $w_{\text{decay}}^u$.
- The accuracy for each task ($acc_{\text{task}}$) should not be less or equal ($\leq$) than the defined value. Therefore, if we consider $acc_{\text{low}}$ as the borderline of low accuracy, then the acceptable accuracy for each task is defined as follows.

$$acc_{\text{task}} > acc_{\text{low}}$$  \hfill (1)

- The average accuracy of the lpSpikeCon-enhanced SNN model ($acc_{\text{avg}}^*$) should be within an acceptable accuracy loss. Therefore, if we consider $acc_{\text{avg}}$ as the average accuracy of the non-quantized SNN model with baseline unsupervised continual learning, and $acc_{\text{class}}$ as the acceptable accuracy loss, then the $acc_{\text{avg}}^*$ is defined as follows.

$$acc_{\text{avg}}^* > (acc_{\text{avg}} - acc_{\text{class}})$$  \hfill (2)

IV. EVALUATION METHODOLOGY

Fig. 9 shows the experimental setup for evaluating lpSpikeCon methodology. We employ a Python-based framework [14] that runs on a multi-GPU machine (i.e., Nvidia RTX 2080 Ti) and an Embedded-GPU machine (i.e., Nvidia Jetson Nano) to perform evaluations on different platforms with different memory and compute capabilities. We employ a single-layer fully-connected network shown in Fig. 2(a) with different network sizes (i.e., 200 and 400 neurons), since it has shown the capabilities for performing unsupervised continual learning under resource- and power-constrained embedded platforms [10]. We consider MNIST dataset as workload, since it has been widely used for evaluating unsupervised continual learning in the SNN community [9–12]. For the baseline unsupervised continual learning, we consider the learning strategy from [10]. The evaluation is performed under both the non-dynamic and dynamic scenarios. Non-dynamic scenarios are provided by feeding the network with training samples whose tasks/classes
are randomly distributed. It aims at simulating conventional offline training where all training samples are already available. Meanwhile, dynamic scenarios are provided by feeding the network with consecutive tasks/classes, where each task/class has the same number of samples, and without re-feeding previous tasks/classes. It aims at simulating an extreme condition where the deployed system receives tasks in a consecutive manner from the environment.

Fig. 9. The experimental setup for evaluating our lpSpikeCon methodology.

V. RESULTS AND DISCUSSION

A. Maintaining Accuracy under Quantized Weights

Dynamic Scenarios: We evaluate accuracy considering different network sizes (i.e., 200 and 400 neurons) and different weight precision levels (i.e., 32, 16, 14, 12, 8, 6, and 4 bits), and the experimental results are provided in Fig. [10] These results show that lower weight precision leads to lower accuracy for more recognition tasks due to information loss. For instance, in a 200-neuron network, 6-bit weights lead to very low accuracy (i.e., ≤ 20%) for recognizing any tasks/classes; see labels 3 and 4 for a 200-neuron network, and labels 7 and 8 for a 400-neuron network. Second, the lpSpikeCon-enhanced SNNs also achieve no accuracy loss on average when compared to the non-quantized SNNs with baseline unsupervised continual learning, across different network sizes. For instance, in a 200-neuron network, average accuracy for the 6-bit and 4-bit weights with lpSpikeCon is 65%, which is slightly higher than the 32-bit weights with baseline learning (i.e., 62%); see labels A and B. A similar pattern is also observed in a 400-neuron network, as the average accuracy for the 6-bit and 4-bit weights with lpSpikeCon are 68% and 67% respectively, which are slightly higher than the 32-bit weights with baseline learning (i.e., 66%); see labels C and D. These accuracy improvements are due to proper adjustments on the selected SNN parameters (i.e., \(V_{th}\) and \(w_{decay}\)). These adjustments trigger the neuronal dynamics in the SNN system to quickly provide available memory for learning new tasks through the “increased \(w_{decay}\)” approach, and trigger learning activity for any incoming tasks (including the new tasks) through the “decreased \(V_{th}\)” approach. From the results, we also observe that in a certain case, an lpSpikeCon-enhanced model may have lower average accuracy than the non-enhanced model under the same weight precision. For instance, in the 200-neuron network with 8-bit weights, our lpSpikeCon-enhanced model achieves 63% accuracy (see label B), while the non-enhanced one achieves 65% (see label D). The reason is that, the lpSpikeCon considers the borderline of low accuracy \((acc_{low})\) as a constraint to determine the parameter adjustments and the output model. Therefore, since the non-enhanced model has lower accuracy than the defined \(acc_{low}\) for one task, this model is not considered as the solution (see label D).

Non-Dynamic Scenarios: We evaluate accuracy considering different network sizes (i.e., 200 and 400 neurons) and different weight precision levels (i.e., 32, 16, 14, 12, 8, 6, and 4 bits), and the experimental results are provided in Fig. [11] These results show that, our lpSpikeCon-enhanced SNNs can achieve comparable accuracy to the non-enhanced SNN counterparts (i.e., SNNs which employ baseline unsupervised continual learning under the same weight precision). For instance, in the 400-neuron network with 6-bit weights, our lpSpikeCon-enhanced SNN achieves 78% accuracy and the non-enhanced one achieves 75% accuracy; while in the 4-bit weights case, the lpSpikeCon-enhanced SNN achieves 71% accuracy and the non-enhanced one achieves 77% accuracy, as highlighted by label E. The reason is that, all the SNNs which employ baseline continual learning. Such high accuracy under non-dynamic scenarios.

B. Weight Memory Savings for Efficient SNN Systems

Fig. [12] shows the memory requirements of different SNN models under different levels of weight precision. These results show that weight quantization in our lpSpikeCon methodology can significantly decrease the memory footprint, since fewer bits are required to represent all weight parameters of an SNN.
model. For instance, a model with 8-bit weights reduces the weight memory by 4x (as shown by label 1), while a model with 4-bit weights reduces the weight memory by 8x (as shown by label 2), as compared to the non-quantized model which employs 32-bit weights. Reduced memory requirement is not only leading to a smaller area in hardware implementation, but also decreasing the number of (off-chip and on-chip) memory accesses. For instance, a non-quantized model requires a single DRAM-based off-chip memory access for obtaining a 32-bit weight, while a quantized model with 8-bit weight can access four weights from a single 32-bit DRAM access [32]. Memory access reduction is important to enable energy-efficient SNN-based autonomous agents mainly for two reasons. First, memory accesses typically dominate the energy consumption of SNN systems, i.e., about 50%-75% of total energy consumption of an SNN accelerator [34]. Second, the
online training with unsupervised continual learning requires frequent memory accesses for updating the weight values at run time. Therefore, memory access reduction can significantly save the overall system energy, which is especially beneficial for memory- and energy-constrained autonomous agents/systems.

The above results and discussion highlight our lpSpikeCon methodology provides several benefits as compared to the state-of-the-art works, including better learning quality under dynamic environments, smaller memory footprint, and higher energy efficiency. Furthermore, our lpSpikeCon can be extended further by incorporating network-specific parameters that have significant impacts on the accuracy. For instance, networks with multiple layers may have additional parameters that should be considered for better adjustments towards adapting to new/unseen features.

VI. CONCLUSION

We propose the lpSpikeCon methodology to enable low-precision SNN processing for efficient unsupervised continual learning on autonomous agents, through three key steps: (1) analysis of the SNN accuracy profiles, (2) identification of SNN parameters and their adjustment rules, and (3) refinements of parameter values for learning activity. As results, our lpSpikeCon significantly reduces weight memory of an SNN model, while maintaining accuracy in both dynamic and non-dynamic scenarios, as compared to the non-quantized model. Therefore, our lpSpikeCon methodology may enable memory- and energy-efficient autonomous agents/systems that are adaptive to diverse operational environments.

ACKNOWLEDGMENT

This work was partially supported by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010. This work was also partially supported by the project “EDL: Auto: An Automated Framework for Energy-Efficient Embedded Deep Learning in Autonomous Systems”, funded by the NYUAD Research Enhancement Fund (REF).

REFERENCES

[1] M. Pfister and T. Pfetsch, “Deep learning with spiking neurons: Opportunities and challenges,” Frontiers in Neuroscience, vol. 12, 2018.
[2] A. Tavanaei et al., “Deep learning in spiking neural networks,” NeuroWorks, vol. 111, pp. 47–63, 2019.
[3] M. Davies et al., “LoSi: A neuromorphic manycore processor with on-chip learning,” IEEE Micro, vol. 38, no. 1, pp. 82–99, January 2018.
[4] H. Hazan et al., “Unsupervised learning with self-organizing spiking neural networks,” in Int. Joint Conf. on Neural Networks (IJCNN), 2018.
[5] D. J. Saunders et al., “Stdp learning of image patches with convolutional spiking neural networks,” in Int. Joint Conf. on Neural Networks, 2018.
[6] H. Hazan et al., “Lattice map spiking neural networks (lm-snns) for clustering and classifying image data,” Annals of Math. and AI, 2019.
[7] D. J. Saunders et al., “Locally connected spiking neural networks for unsupervised feature learning,” Neural Networks, vol. 119, 2019.
[8] R. V. W. Putra and M. Shafique, “Fspinn: An optimization framework for memory- and energy-efficient spiking neural networks,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 39, no. 11, pp. 3601–3613, 2020.
[9] P. Panda et al., “Asp: Learning to forget with adaptive synaptic plasticity in spiking neural networks,” IEEE J. on Emerging and Selected Topics in Circuits and Systems (JETCAS), vol. 8, no. 1, pp. 51–64, March 2018.
[10] R. V. W. Putra and M. Shafique, “Spikedyen: 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 (DAC), 2021, pp. 1057–1062.
[11] J. M. Allred and K. Roy, “Unsupervised incremental stdp learning using forced firing of dormant or idle neurons,” in Int. Joint Conf. on Neural Networks (IJCNN), 2016, pp. 2492–2499.
[12] J. M. Allred and K. Roy, “Controlled forgetting: Targeted stimulation and dopaminergic plasticity modulation for unsupervised lifelong learning in spiking neural networks,” Frontiers in Neuroscience, vol. 14, p. 7, 2020.
[13] R. V. W. Putra and M. Shafique, “Q-spinn: A framework for quantizing spiking neural networks,” in Int. Joint Conf. on Neural Networks (IJCNN), 2021, pp. 1–8.
[14] H. Hazan et al., “Bindsnet: A machine-learning-oriented spiking neural networks library in python,” Frontiers in Neuroinformatics, 2018.
[15] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” Neural Networks, vol. 10, no. 9, pp. 1659–1671, 1997.
[16] R. V. W. Putra, M. A. Hanif, and M. Shafique, “Recon: Energy-efficient fault-tolerance for spiking neural networks considering unreliable memories,” in 2021 IEEE/ACM Int. Conf. on Computer Aided Design (ICCAD), 2021, pp. 1–9.
[17] M. Shafique et al., “Towards energy-efficient and secure edge ai: A cross-layer framework iccad special session paper,” in 2021 IEEE/ACM Int. Conf. on Computer Aided Design (ICCAD), 2021, pp. 1–9.
[18] R. V. W. Putra, M. A. Hanif, and M. Shafique, “Sparkxd: A framework for resilient and energy-efficient spiking neural network inference using approximate dram,” in 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021, pp. 379–384.
[19] J. Gauthrais and S. Thorpe, “Rate coding versus temporal order coding: a theoretical approach,” Biosystems, vol. 48, no. 1, pp. 57–65, 1998.
[20] C. Kayser et al., “Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns,” Neuron, vol. 61, 2009.
[21] S. Thorpe and J. Gauthrais, “Rank order coding,” in Computational neuroscience. Springer, 1998, pp. 113–118.
[22] S. Park et al., “Fast and efficient information transmission with burst spikes in deep spiking neural networks,” in 56th Annual Design Automation Conference (DAC), 2019, p. 53.
[23] R. V. W. Putra, M. A. Hanif, and M. Shafique, “Softsnn: Low-cost fault tolerance for spiking neural network accelerators under soft errors,” arXiv preprint arXiv:2203.05523, 2022.
[24] E. M. Izhikevich, “Which model to use for cortical spiking neurons?” IEEE Trans. on Neural Networks (TNN), vol. 15, no. 5, Sep. 2004.
[25] P. Diehl and M. Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity,” Frontiers in Computational Neuroscience, vol. 9, p. 99, 2015.
[26] M. Hopkins et al., “Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations,” Philosophical Transactions of the Royal Society A, vol. 378, 2020.
[27] S. Gupta et al., “Deep learning with limited numerical precision,” in Int. Conf. on Machine Learning (ICML), 2015, p. 1737–1746.
[28] R. Krishnamoorthi, “Quantizing deep convolutional networks for efficient inference: A whitepaper,” arXiv, vol. 1806.08342, 2018, p. 7, 2020.
[29] Z. Chen and B. Liu, “Lifelong machine learning,” Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 12, no. 3, 2018.
[30] G. I. Parisi et al., “Continual lifelong learning with neural networks: A review,” Neural Networks, vol. 113, pp. 54 – 71, 2019.
[31] M. McCloskey and N. J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” Psychology of Learning and Motivation, vol. 24, pp. 109 – 165, 1989.
[32] R. V. W. Putra et al., “Dramap: A generic dram data mapping policy for energy-efficient processing of convolutional neural networks,” in 57th ACM/IEEE Design Automation Conference (DAC), 2020, pp. 1–6.
[33] R. V. W. Putra et al., “Romanet: Fine-grained reuse-driven off-chip memory access management and data organization for deep neural network accelerators,” IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 4, pp. 702–715, 2021.
[34] S. Kothivasan et al., “Dynasp: Spike bundling for energy-efficient spiking neural networks,” in Int. Symp. on Low Power Electronics and Design (ISLPED), July 2019, pp. 1–6.