Intermittent Neuromorphic Wearable Systems

Junaid Ahmed Qazi, Emil Njor, Matthias Bo Stuart, Xenofon Fafoutis, Charalampos Orfanidis
Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU)
Kgs. Lyngby, Denmark
{jahqa, emjn, mbst, xefa, chaorf}@dtu.dk

***Abstract*— Wearable medical devices have become a focal point of research and development, particularly for their use in long-term monitoring of patients with chronic diseases. These devices offer significant advantages, including early detection of complications, reduced hospital readmission rates, and overall lower healthcare costs by enabling a more proactive approach to patient care. In recent years, the integration of Artificial Intelligence (AI) into wearable systems has gained considerable attention, further enhancing the capabilities of these devices. However, AI-driven wearables consume significantly more power than their traditional counterparts, which can limit the device's lifetime. To alleviate this problem, this paper presents a novel framework combining two state-of-the-art techniques for reducing embedded AI energy consumption: neuromorphic computing and intermittent computing. As a proof of concept, our framework is applied to an Electromyography (EMG) application to classify various hand gestures using the Ninapro DB2 dataset. Our findings demonstrate that the proposed framework supports the creation of wearable AI devices with state-of-the-art energy efficiency.**

I. Introduction

Creating performant wearable medical devices for long-term monitoring of chronically ill patients and the elderly has been a critical area of research for many years [1]–[3]. Long-term monitoring using wearables offers transformative advantages, including early detection of life-threatening complications, reduced hospital readmission, and subsequent lowering of overall healthcare costs [4]. These benefits contribute to improved patient outcomes and a more sustainable healthcare system.

Integrating Artificial Intelligence (AI) into these wearable medical devices is a promising avenue for further improving their capabilities [5]. For example, AI in medical wearables can be used to predict complex complications such as strokes and arrhythmia through advanced pattern recognition. A critical challenge in introducing AI in medical wearables is the limited resources (i.e., computation, memory, and energy) available on wearable systems. As AI models are significantly more resource-demanding than traditional computing systems, these resource constraints present a significant challenge to be overcome. Resource-efficient AI [6] has gained traction as a research field in recent years, introducing several novel approaches for improving the energy efficiency of AI models. Two promising approaches include neuromorphic computing and intermittent computing, which we combine into a novel framework in this paper.

Neuromorphic computing is a novel computational paradigm that seeks to mimic the processing done by neurons in the human brain. These mechanisms can be modeled in Spiking Neural Networks (SNNs) [7] [8], where intelligent decisions are born from data represented as "spikes" traversing through a modified neural network structure [9]. Spikes are discrete events over time, contrary to artificial neural networks (ANNs), where data is represented as continuous values. In an SNN, computations happen only when incoming spikes reach a threshold, effectively reducing unnecessary computations. The event-based nature of spikes allows SNNs to consume significantly less energy than ANNs on appropriate hardware [10], [11].

Intermittent computing is a paradigm where wearable systems operate intermittently based on non-continuous power sources, preserving their state across power cycles [12]. This can enable wearables to run on energy-harvesting sources such as solar or thermal energy. In intermittent computing, energy is used for computation purposes when it exceeds a set threshold. After some amount of computation, the device goes back to sleep or powers off completely and waits until a sufficient amount of energy is available to start a new cycle. Before the sleep/power off, essential variables are stored in non-volatile memory to enable the restoration of the computational state when energy is once again available. The threshold of when to start computation must be set appropriately to avoid power failures. See [13] for an illustrating example of intermittent computing used for the remote deployment of a system for monitoring the fauna in a forest with acoustic event detection.

Our framework utilizes the available resources of wearable systems more efficiently and allows the integration of AI models in even more wearable devices for healthcare. As a proof of concept, we apply our framework to an electromyography (EMG) [14], [15] application and compare its performance in terms of energy, latency, and accuracy against a non-intermittent baseline. This demonstration showcases the potential of our framework to create energy-harvesting wearable systems for long-term monitoring of critical health conditions, such as monitoring muscle activity for neuromuscular disorders [16], [17]. To the best of our knowledge, this work represents the first investigation into the feasibility and performance of intermittent computing-based SNNs, opening up a new frontier in AI-driven wearable healthcare.

II. Background

This section further elaborates on the principles of neuromorphic and intermittent computing, both serving as a

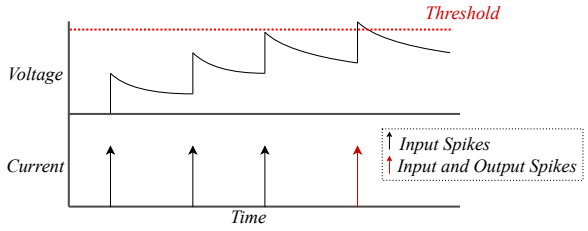

Fig. 1. Illustration of a LIF model. Change of voltage caused by the input spikes. An output spike is observed when the voltage reaches a threshold.

foundation for understanding the following sections.

A. Neuromorphic Computing

Unlike traditional computing systems, neuromorphic computers do not rely on the Von Neumann or Harvard architectures. Instead, they use neurons and synapses for memory and processing, mimicking the behavior of the biological neurons present in our brains. In neuromorphic computing, computation and memory storage occur simultaneously within the same physical units, exactly as it does in biological neurons. This eliminates the energy-demanding task of data shuffling between processing and memory units, which is a major bottleneck in traditional computers. Neurons and synapses operate as independent units, where each individual unit is capable of performing its own computations. This decentralized approach prevents congestion bottlenecks from occurring in centralized computing, thereby enabling tasks such as pattern recognition, sensor data analysis, and decision-making to take place in parallel [18].

B. Spiking Neural Networks

A common algorithm developed for neuromorphic computing is a Spiking Neural Network (SNN), which, simply put, is an ANN that uses "spikes" instead of continuous numerical values for computation. These spikes are discrete events sent between neurons in an SNN. A neuron sends out a new spike whenever a specific threshold is reached from receiving spikes from other neurons [10].

For data to be passed to an SNN, it must be encoded into spikes over time — also known as a spike train [18]. Therefore, to use common datasets, the data should be converted into spikes through a process known as "input encoding" [19]. The spikes generated from this input encoding are fed to the SNN and, over time, create a cascade of spikes throughout the SNN.

Each neuron in an SNN contains a neuron model that describes how incoming spikes contribute to building up a "membrane potential" culminating in a spike when the threshold is reached.

A common neuron model is the Integrate-and-Fire (IF) model. When an IF neuron model receives spikes, it increases the membrane potential of the neuron, and if the potential reaches a certain threshold, the neuron generates an output spike. After firing, the membrane potential is reset to a baseline value [20]. A more advanced neuron model is the Leaky Integrate-and-Fire (LIF) model, which introduces a leakage. In the LIF model, the membrane potential decreases over time if no new input spikes are received [10]. This leakage mimics the natural decay of electrical charge in biological neurons. Fig. 1 illustrates a LIF neuron. Another popular model is the Current-Based Leaky Integrate-and-Fire (CUBA-LIF), which in addition to the parameters of an LIF model, also incorporates synaptic currents. When a spike is fired, it travels through the synapses to the next neuron. These synapses act as channels that modulate the strength and timing of the signal transmission, influencing how the postsynaptic neuron's membrane potential changes [20].

All data in an SNN are encoded into spikes, including the output classification data, and thus, there is a need to decode the output spikes into a classification. There are many output decoding schemes; however, *rate decoding* and *latency decoding* are among the most widely used. In rate decoding, the neuron with the highest output spike count represents the predicted class, while in latency decoding, it is determined by the first output neuron to fire [10].

Processing in SNNs takes place over time, typically split into discrete timesteps [9]. Timesteps represent the smallest time interval over which the network is updated. During each timestep, neurons receive incoming spikes, integrate these inputs into their membrane potential, and send an output spike if their threshold is reached. This process repeats iteratively for each time step, resulting in the network evolving over time. Time steps allow SNNs to capture temporal patterns of data, and ensure all neurons within the network operate in a synchronized manner [20]. Due to the temporal nature of input spikes, precise timing is a requirement for SNNs.

C. Intermittent Computing

In intermittent computing, a device's state is preserved in non-volatile memory before energy depletion in order to restore it once sufficient energy is harvested again. This process is known as *checkpointing* and includes storing required information and variables to restore the state of the device. Traditionally, devices are designed to operate on a continuous energy source. However, with the advent of intermittent computing and energy harvesting, devices often face interruptions in power supply, leading them to revert to their initial state after a failure. Checkpointing addresses this challenge by enabling the device to return to a previous state instead of restarting from the beginning [21]. A comparison between conventional and intermittent execution of AI models is illustrated in Fig. 2.

III. Materials and Methods

This section presents the details of our neuromorphic and intermittent computing framework. We first describe the design of the framework, in particular, which variables must be saved in checkpoints based on the neuron model, output decoding and number of timesteps used. We then explain our strategy for evaluating the resource consumption of SNN, the overhead of our checkpointing framework, and propose a checkpointing placement strategy. Lastly, we introduce the

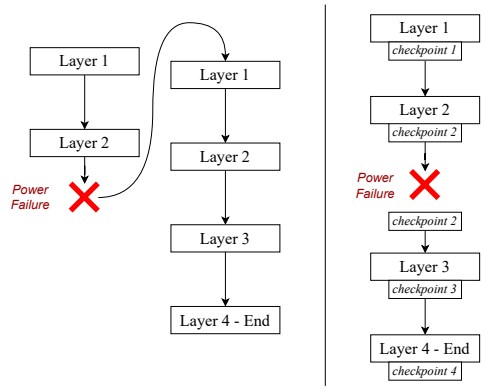

Fig. 2. Standard AI model with power failure (left) compared to an AI model with checkpointing with power failure (right).

SNN that we will use in future sections as an example of a practical use of our framework.

A. Intermittent Neuromorphic Computing Framework

To enable intermittent computation in SNNs, checkpointing is used to store only the necessary dynamic parameters, i.e., those that change during computation. As we are only considering SNN inference, we are guaranteed that the model's weights and input encoding remain constant. Therefore, the checkpointing strategy requires saving only the neuron model states, the timestep information, and the output decoding results.

1) Neuron Models and Checkpointing Strategy: We consider two neuron models within our framework: Leaky Integrate-and-Fire (LIF) and CUBA-LIF. The key distinction between them is how they process input spikes. The LIF Model considers only membrane potential changes, making membrane potential the sole dynamic parameter. The CUBA-LIF Model incorporates both membrane potential and synaptic currents as dynamic parameters, requiring both to be stored in checkpoints. Since the model consists of a second-order LIF neuron, synaptic currents, and membrane potentials change as input data propagates through the layers. These must be stored in a checkpoint at specific intervals.

2) Checkpointing Overhead Across Timesteps and Layers: Our framework only stores one SNN checkpoint at a time, where a new checkpoint overwrites the previous one when it is saved. The total checkpoint overhead in bytes can, therefore, be summarized in Table II, while Table I provides notation definitions.

3) Output Decoding Considerations: We consider two types of output decoding in the framework. The first is *rate decoding*, where the predicted class is determined by the output neuron with the highest spike count. For this output decoding to work intermittently, we must store all previous output spikes in a checkpoint. We refer to the memory size of this collection of dynamic variables as spk_{out}. The second is the *temporal decoding*, where the predicted class is determined by the first output neuron to spike. For this output decoding to work intermittently, we only need to save the first output spike. Thus, once the first output spike is fired, computation can cease. When using temporal decoding, we therefore set $spk_{out} = 0$ as no output spikes will have to be saved.

4) Timesteps and Checkpointing Dynamics: Timesteps function as a scaling factor for checkpointing overhead. If only one timestep is used, neuron model parameters do not need to be stored — only spike outputs (spk_{layer}). However, for a model using multiple timesteps, the stored parameters depend on the timestep and model layer, as shown in Fig. 3.

This structured checkpointing strategy ensures minimal memory overhead while maintaining the ability to resume inference efficiently in intermittent environments.

TABLE I

NOTATION AND DESCRIPTIONS OF PARAMETERS.

Notation	*Description*
spk_{layer}	Checkpoint spike train
spk_{out}	Final layer spike train
ts	Timesteps
$state_{all}$	State of all membrane potentials and synaptic currents
$state_{rem}$	State of membrane potentials and synaptic currents after checkpoint
$state_{prev}$	State of membrane potentials and synaptic currents before checkpoint

TABLE II

CHECKPOINT OVERHEAD ACROSS DIFFERENT TIMESTEPS AND MODEL LAYERS.

Timestep	*Model Layer*	*Checkpoint Overhead*
First	First	spk_{layer} + $state_{prev}$
	Intermediate	spk_{layer} + $state_{prev}$
	Final	$state_{all}$ + spk_{out}
Intermediate	First	spk_{layer}+$state_{all}$ + spk_{out}
	Intermediate	spk_{layer}+$state_{all}$ + spk_{out}
	Final	$state_{all}$ + spk_{out}
Final	First	spk_{layer}+$state_{rem}$ + spk_{out}
	Intermediate	spk_{layer}+$state_{rem}$ + spk_{out}
	Final	None

B. Evaluation Methodology

The literature describes several quantitative methods for calculating the energy and latency of SNNs. [22] proposes a simple approach that multiplies the energy of a single Synaptic Operation (SOP) with the number of spikes that caused an SOP. They used the Intel Loihi SOP power consumption in their calculations and estimated that their SNN is 65-135 times more energy efficient than an ANN. The methodology used for energy consumption calculations described in [23] is different, as they mention that SOP plays only a small part within the total energy consumption of an SNN. The methodology of [23] can be summarized as follows:

$$E_{inf} = E_{mem} + E_{ops+addr} \quad (1)$$

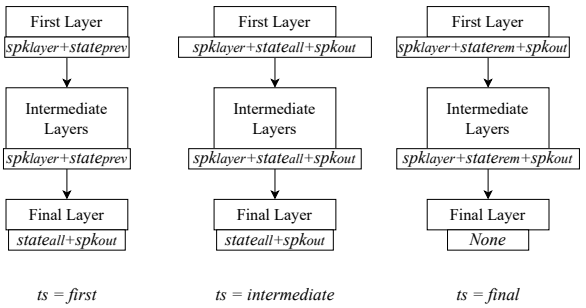

Fig. 3. Checkpointing strategy based on timestep and model layer.

The energy consumption of running SNN inference is denoted by E_{inf}, the energy consumed by memory access is represented by E_{mem}, and the energy associated with synaptic operations and addressing mechanisms is expressed as $E_{ops+addr}$. We use this method for our evaluation. At the time of writing, to the best of our knowledge, there is no established methodology for calculating the inference latency of an SNN in the literature.

Regarding the energy and latency overhead of checkpoints, we assume that they can be generalized as writing data to non-volatile memory (NVMe). Using this assumption, the energy consumption for a checkpoint of a given amount of bytes can be calculated according to eq. (2).

$$E_{cp} = Overhead_{cp} \times E_{NVMe} \tag{2}$$

Where E_{cp} is the energy consumption of a checkpoint, $Overhead_{cp}$ is the total overhead in bytes for a checkpoint, and E_{NVMe} is the NVMe energy consumption for a byte. Similarly, the latency of a checkpoint can be calculated according to eq. (3).

$$T_{cp} = Overhead_{cp} \times T_{NVMe} \tag{3}$$

Where T_{cp} is the latency of a checkpoint, $Overhead_{cp}$ is the total overhead in bytes for a checkpoint, and T_{NVMe} is the NVMe Latency for a byte.

For the values of E_{NVMe} and T_{NVMe}, we refer to the datasheet of a widely available NVMe module [24]. This specific module was selected based on its balance between energy consumption, latency, and total storage capacity. It is important to note that the choice of NVMe module can significantly impact system viability and should therefore be carefully considered when designing a hardware implementation.

In certain scenarios, the overhead of writing a checkpoint might be larger than the total inference overhead of the model, making checkpointing unnecessary. We evaluate this by comparing the energy overhead of the (non-checkpointing) SNN with the energy overhead of the checkpointing SNN as seen in Eq. 4.

$$E_{cp} + E_{remain} < E_{inf} \tag{4}$$

Where E_{remain} is the energy consumption required to finish inference after a checkpoint, and E_{inf} is the total inference energy consumption. Using Eq. 4, we can determine which parts of the SNN are feasible locations for a checkpoint. The evaluation of the latency is solely based on T_{cp}.

Eq. 4 defines regions of the model where checkpointing is meaningful. However, it does not determine the optimal checkpointing frequency. We say that a checkpoint is "optimal" if the energy overhead of creating a checkpoint is less than the energy overhead of computation since the last checkpoint. Using this strategy, a checkpoint will never take more energy to create than the energy overhead of reaching the same computational state in the case of a loss of power. An optimal checkpointing strategy places checkpoints whenever Eq. 5 is true.

$$E_{before} > E_{cp} \tag{5}$$

where E_{before} represents the total energy consumption since the previous checkpoint.

C. SNN Design

In order to provide a proof of concept of our framework, we need an SNN to apply it to. For that purpose, we create a Convolutional Spiking Neural Network (CSNN), where the first layer is convolutional, followed by three fully connected layers, as seen in Table III. The model layers were selected through iterative testing and by reviewing the literature for similar models [25] [26]. Specific layer configurations were selected through hyperparameter optimization. This was done both for the convolution and fully connected layers, and for any neuron model parameters. We find that ten timesteps result in the highest performance. Finally, we implement the model using the SNNTorch framework [10].

To train the model, we use the Ninapro DB2 dataset [25] that consists of surface Electromyography (sEMG) signals recorded using 12 electrodes of 40 patients performing 49 different hand and wrist gestures (e.g., finger movements, grasping gestures). The EMG data in this dataset have been filtered to remove power line interference using a Hampel filter. Furthermore, the data streams of all electrodes were synchronized using high-resolution timestamps. To reduce model training time, we created a patient-specific model using the data of a random patient and the first 12 gestures of exercise B in the dataset, as illustrated in Fig. 4. The dataset is not in the spike domain and thus requires input encoding. Rate encoding and decoding were selected for this task through iterative testing. Moreover, a second-order Leaky Integrate-and-Fire neuron model was chosen due to its high performance for temporal data.

IV. Results

This section presents the performance evaluation of the introduced framework on the previously introduced SNN model. We first examine the model's accuracy, followed by an analysis of energy consumption per layer. Next, we quantify the overhead introduced by checkpointing in terms of energy and latency, highlighting variations based on checkpoint locations. Finally, we compare the model's overall energy and latency with and without checkpointing,

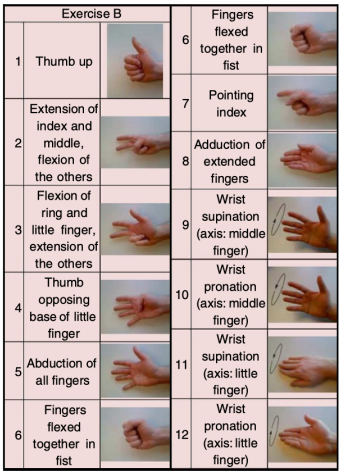

Fig. 4. The selected hand movements from the Ninapro DB2 dataset [25]. Original image shared under ©CC BY 4.0

TABLE III
OVERVIEW OF THE SNN MODEL, CONSISTING OF A SINGLE CONVOLUTION LAYER, FOLLOWED BY THREE FULLY-CONNECTED LAYERS

Layer	*Input*	*Output*
Input	-	$12 \times 1 \times 200$
Convolution	$12 \times 1 \times 200$	$12 \times 1 \times 32$
Flatten	$12 \times 1 \times 32$	1×384
FC1	1×384	1×512
FC2	1×512	1×256
FC3 - Output	1×256	1×12

demonstrating the trade-offs involved in optimizing resource usage for wearable systems.

Fig. 5 depicts the performance of our SNN model in terms of test and training accuracy. The model is trained over 100 epochs, and the test accuracy stabilizes around epoch 20 at 70%. The accuracy seems to be in line with the state of the art [27]. Note that the objective of this paper is not to create a high-performance model but rather to investigate whether the intermittent neuromorphic computing paradigm is a viable option for health applications.

Table IV presents the energy consumption of the inference for each layer without checkpoints. Layer 1, the convolution layer, exhibits the highest energy consumption according to the methodology in [23]. Both kernel size and filter size were found to impact the convolution layer's total energy consumption. As a result, efforts were made to keep these parameters low enough to minimize energy consumption, while ensuring model performance was not significantly compromised. Table V shows the overhead added when we use checkpoints in terms of energy and latency. Observe that the overhead alters significantly between checkpoint locations.

Using the framework defined in the previous section, we estimate the optimal checkpoint locations in the model as shown in Table VI. Fig. 6 illustrates the energy consumption

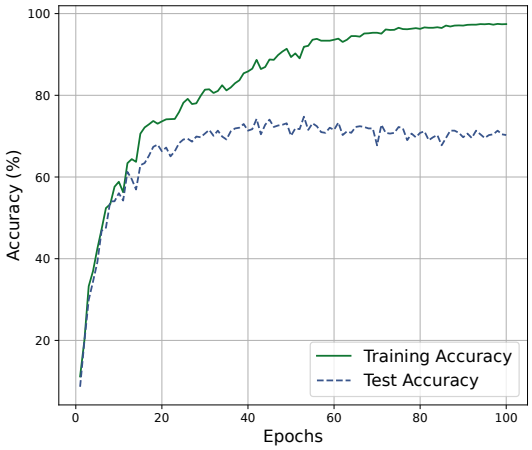

Fig. 5. Train and test accuracy of the SNN model over 100 epochs.

TABLE IV
TOTAL ENERGY FOR EACH MODEL LAYER

Layer	*Energy (μJ)*
1	427.78
2	56.05
3	17.83
4	1.35
Total Energy	503.01

of the SNN model with and without using our framework as well. Note here that using the framework adds minor energy overhead, which should be tolerable in many wearable systems.

Fig. 7 presents the inference latency of the model when operating with checkpoints. While the energy overhead seems negligible, the time delay does not. Although this can seem daunting, it is important to consider that it is common for intermittent computing applications to have periods where they do not operate (i.e., are in SLEEP mode) and just harvest energy. Therefore, health applications that tolerate those limitations should also tolerate the increased inference latency. A discussion about applications that tolerate inter-

TABLE V
OVERHEAD OF CHECKPOINTING IN DIFFERENT LOCATIONS

Timestep	*Checkpoint*	*Energy (μJ)*	*Latency (ms)*
First	1	27.88	25.34
	2	64.20	58.37
	3	81.67	74.24
	4	81.97	74.52
Intermediate	1	82.82	75.29
	2	83.10	75.54
	3	82.54	75.03
	4	81.97	74.52
Final	1	55.78	50.71
	2	20.02	18.2
	3	1.43	1.30
	4	0.00	0.00

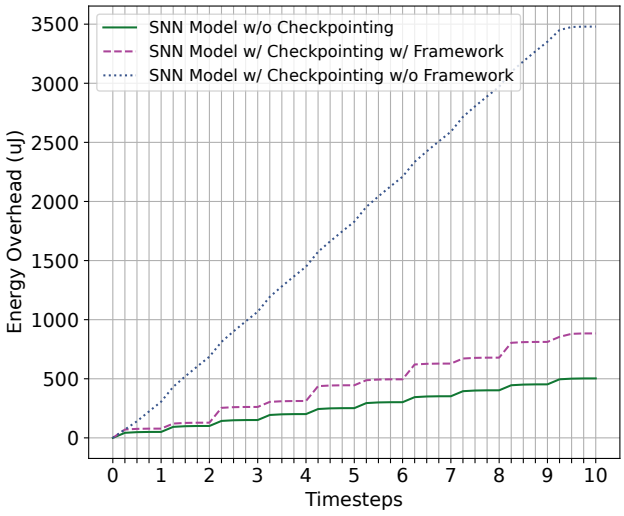

Fig. 6. Energy overhead comparison of the model without checkpointing, with checkpointing after every layer, and with checkpointing.

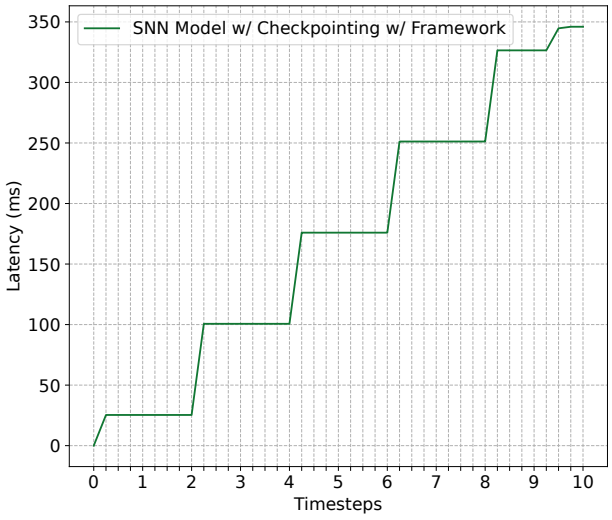

Fig. 7. Latency overhead of the SNN model with checkpointing.

mittent neuromorphic computing follows in the next section.

TABLE VI
CHECKPOINT LOCATIONS IN THE SNN MODEL FOUND THROUGH THE EVALUATION METRICS IN THE FRAMEWORK

Checkpoint Locations
Timestep 1, Checkpoint 1
Timestep 3, Checkpoint 1
Timestep 5, Checkpoint 1
Timestep 7, Checkpoint 1
Timestep 9, Checkpoint 1
Timestep 10, Checkpoint 2
Timestep 10, Checkpoint 3

V. DISCUSSION

In this paper, we have introduced a framework that can systematically determine the necessary data to be stored in an intermittent computing checkpoint in an SNN based on the location within the model and the type of model being used. Additionally, we have developed methodologies to identify optimal checkpoint locations and evaluate the impact of checkpointing on computational overhead.

Table IV illustrates that the model has a total energy consumption of 503.01 µJ, in which the first layer, convolution, is the largest contributor. Table V indicates the energy overhead added due to the checkpoints, in which we see that the overhead depends heavily on the timestep and checkpoint location. First and final timestep checkpoints have a lower energy overhead compared to checkpoints in intermediate timesteps. This happens as the model's state, i.e., synaptic currents and membrane potentials, does not have to be stored within a checkpoint in certain cases. Using the proposed framework, the checkpoint frequency and location were evaluated and illustrated in Table VI.

Fig. 6 illustrates the energy consumption of the SNN model. Using the proposed framework to add checkpoints, the energy consumption is increased from 503.01 µJ to 883.61 µJ. Adding a checkpoint after each layer results in a significantly higher increase of 3479.35 µJ. Although adding checkpoints after each layer minimizes data loss in the event of a power failure, its substantial energy overhead makes it impractical for wearable devices. In contrast, the framework achieves an optimized balance, maintaining reasonable energy overhead while effectively reducing potential data loss after power failure.

Fig. 7 and Table V show that the added latency from checkpointing is non-negligible. These delays can be reduced by using sophisticated memory technologies such as magnetoresistive RAM (MRAM) and resistive RAM (ReRAM). However, intermittent computing systems are powered mainly by energy harvesting. These energy sources are inherently unpredictable, and the time to harvest sufficient energy to execute a number of instructions might be much larger than the latency introduced by checkpointing. While significant relative to the inherent latency of model inference, the latency introduced by checkpointing is comparatively minor to the latency incurred by intermittent computing operations from a non-continuous power source.

To this end, it is essential to highlight that intermittent neuromorphic computing is designed for application scenarios that can tolerate non-operational time intervals. For instance, when monitoring chronic muscular disorders such as Amyotrophic lateral sclerosis (ALS) [28], short gaps may be tolerable if the overall trend of muscle activity can still be captured. Likewise, Duchenne Muscular Dystrophy [29] monitoring can tolerate some non-functional periods where the system is harvesting energy [17]. On the other hand, intermittent neuromorphic computing is unsuitable for applications such as freezing of gait detection in patients with Parkinson's disease, which require constant monitoring and low latency responses [30].

VI. Conclusion

In this paper, we explored the feasibility of integrating intermittent computing into neuromorphic systems, specifically Spiking Neural Networks (SNNs), for wearable applications in health. We implemented a framework to efficiently place checkpoints in an SNN, and to include only the bare necessary data in order to enable intermittent operation. Our findings indicate that the checkpointing overhead is highly dependent on model architecture, checkpoint placement, and the operational constraints of intermittent computing.

While intermittent computing introduces overhead in terms of energy consumption and latency, these trade-offs are often tolerable in health applications where continuous operation is not strictly required. In particular, applications such as neuromuscular disorder monitoring can benefit from intermittent neuromorphic without significant loss of information.

Future work will focus on evaluating the implemented framework on real neuromorphic hardware. Additionally, exploring techniques that could enhance the practicality of intermittent neuromorphic computing for real-world deployments, such as advanced memory technologies to mitigate introduced delays. By researching these approaches, we aim to further advance intermittent neuromorphic wearable systems for healthcare applications.

References

[1] M. T. Mardini, Y. Iraqi, and N. Agoulmine, "A survey of healthcare monitoring systems for chronically ill patients and elderly," *Journal of Medical Systems*, vol. 43, p. 50, Jan 2019.

[2] V. Patel, A. Orchanian-Cheff, and R. Wu, "Evaluating the validity and utility of wearable technology for continuously monitoring patients in a hospital setting: Systematic review," *JMIR Mhealth Uhealth*, vol. 9, p. e17411, Aug 2021.

[3] Y. Guo, X. Liu, S. Peng, X. Jiang, K. Xu, C. Chen, Z. Wang, C. Dai, and W. Chen, "A review of wearable and unobtrusive sensing technologies for chronic disease management," *Computers in Biology and Medicine*, vol. 129, p. 104163, 2021.

[4] G. Mattison, O. Canfell, D. Forrester, C. Dobbins, D. Smith, J. Töyräs, and C. Sullivan, "The influence of wearables on health care outcomes in chronic disease: Systematic review," *J Med Internet Res*, Jul 2022.

[5] N. Gautam, S. N. Ghanta, J. Mueller, M. Mansour, Z. Chen, C. Puente, Y. M. Ha, T. Tarun, G. Dhar, K. Sivakumar, Y. Zhang, A. A. Halimeh, U. Nakarmi, S. Al-Kindi, D. DeMazumder, and S. J. Al'Aref, "Artificial intelligence, wearables and remote monitoring for heart failure: Current and future applications," *Diagnostics*, vol. 12, 2022.

[6] É. Buteau, G. Gagné, W. Bonilla, M. Boukadoum, P. Fortier, and B. Gosselin, "Tinyml for real-time embedded HD-EMG hand gesture recognition with on-device fine-tuning," in *2024 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1–6, 2024.

[7] T. Liu, Y. Ning, P. Liu, Y. Zhang, Y. Chua, W. Chen, and S. Zhang, "Modularity facilitates classification performance of spiking neural networks for decoding cortical spike trains," in *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 1–4, 2023.

[8] B. S. Sai Badrinatha Reddy, P. K, and B. J. Kailath, "SNN with gradient-based backpropagation algorithm for ECG arrhythmia classification with LIF neuron and AdEx neuron," in *2024 28th International Symposium on VLSI Design and Test (VDAT)*, pp. 1–6, 2024.

[9] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, no. 9, 1997.

[10] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023.

[11] S. Sai, S. Sharma, and V. Chamola, "Explainable AI-empowered Neuromorphic Computing Framework for Consumer Healthcare," *IEEE Transactions on Consumer Electronics*, pp. 1–1, 2024.

[12] S. Ahmed, B. Islam, K. S. Yildirim, M. Zimmerling, P. Pawełczak, M. H. Alizai, B. Lucia, L. Mottola, J. Sorber, and J. Hester, "The Internet of Batteryless Things," *Communications of the ACM*, vol. 67, no. 3, pp. 64–73, 2024.

[13] B. Islam, Y. Luo, and S. Nirjon, "Amalgamated intermittent computing systems," in *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, IoTDI '23, (New York, NY, USA), p. 184–196, Association for Computing Machinery, 2023.

[14] J. Heaffey, E. Koutsos, and P. Georgiou, "Wearable device for remote EMG and muscle fatigue monitoring," in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2015.

[15] R. P. Mathews, H. Jafari Sharemi, I. Habibagahi, J. Jang, A. Ray, and A. Babakhani, "Towards a miniaturized, low power, batteryless, and wireless bio-potential sensing node," in *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 404–408, 2022.

[16] J. C. Deenen *et al.*, "The Epidemiology of Neuromuscular Disorders: A Comprehensive Overview of the Literature," *Journal of Neuromuscular Diseases*, vol. 2, pp. 73–85, 2015.

[17] V. Kartsch, S. Benatti, M. Mancini, M. Magno, and L. Benini, "Smart wearable wristband for EMG based gesture recognition powered by solar energy harvester," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018.

[18] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "SPAN: spike pattern association neuron for learning spatio-temporal spike pattern," *International Journal of Neural Systems*, vol. 22, no. 04, p. 1250012, 2012. PMID: 22830962.

[19] D. Auge, J. Hille, M. Etienne, and A. Knoll, "A survey of encoding techniques for signal processing in spiking neural networks," *Neural Processing Letters*, vol. 53, p. 4693–4710, 2021.

[20] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Brain Sciences*, vol. 12, no. 7, 2022.

[21] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent Computing: Challenges and Opportunities," in *Leibniz International Proceedings in Informatics, Lipics*, vol. 71 of *2nd Summit on Advances in Programming Languages, SNAPL 2017*, p. 8, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

[22] U. K. N. G. W. Gamage, L. Zanatta, M. Fumagalli, C. Cadena, and S. Tolu, "Event-Based Classification of Defects in Civil Infrastructures with Artificial and Spiking Neural Networks," in *Lecture Notes in Computer Science*, vol. 17, pp. 629–640, Springer, 2023.

[23] E. Lemaire, L. Cordone, A. Castagnetti, P. E. Novac, J. Courtois, and B. Miramond, "An analytical estimation of spiking neural networks energy efficiency," in *29th International Conference on Neural Information Processing, ICONIP 2022*, vol. 13623 of *Lecture Notes in Computer Science*, (Virtual Event), pp. 574–587, Springer, 2023.

[24] Fujitsu Semiconductor Limited, "Mb85rc256v - 256k (32k × 8) bit i²c interface fram." https://www.fujitsu.com/uk/Images/MB85RC256V-20171207.pdf, 2017. Accessed: 2025-04-30.

[25] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.-G. M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Müller, "Electromyography data for non-invasive naturally-controlled robotic hand prostheses," *Scientific Data*, vol. 1, no. 1, p. 140053, 2014.

[26] A. Vitale, E. Donati, R. Germann, and M. Magno, "Neuromorphic edge computing for biomedical applications: Gesture classification using EMG signals," *IEEE Sensors Journal*, vol. 22, no. 20, 2022.

[27] C. Frenkel, D. Bol, and G. Indiveri, "Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence," *Proceedings of the IEEE*, vol. 111, pp. 623–652, 2023.

[28] E. L. Feldman, S. A. Goutman, S. Petri, L. Mazzini, M. G. Savelieff, P. J. Shaw, and G. Sobue, "Amyotrophic lateral sclerosis," *The Lancet*, vol. 400, pp. 1363–1380, Oct 2022.

[29] B.-Y. Youn, Y. Ko, S. Moon, J. Lee, S.-G. Ko, and J.-Y. Kim, "Digital biomarkers for neuromuscular disorders: A systematic scoping review," *Diagnostics*, vol. 11, no. 7, 2021.

[30] A. Moore, J. Li, C. H. Contag, L. J. Currano, C. O. Pyles, D. A. Hinkle, and V. S. Patil, "Wearable surface electromyography system to predict freeze of gait in Parkinson's disease patients," *Sensors*, 2024.