POLITECNICO DI TORINO
Repository ISTITUZIONALE

Variable-precision neuromorphic state space model for on-edge activity classification

Original
Variable-precision neuromorphic state space model for on-edge activity classification / Leto, B., Urgese, G., Macii, E., Fra, V.. - In: FUTURE GENERATION COMPUTER SYSTEMS. - ISSN 0167-739X. - ELETTRONICO. - 176:(2025), pp. 1-10. [10.1016/j.future.2025.108193]

Availability:
This version is available at: 11583/3004483 since: 2025-11-07T15:23:48Z

Publisher:
ELSEVIER

Published
DOI:10.1016/j.future.2025.108193

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Elsevier postprint/Author's Accepted Manuscript

© 2025. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/.The final authenticated version is available online at: http://dx.doi.org/10.1016/j.future.2025.108193

(Article begins on next page)

27 November 2026

Future Generation Computer Systems 176 (2026) 108193



Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs



Variable-precision neuromorphic state space model for on-edge activity classification

Benedetto Leto, Gianvito Urgese, Enrico Macii, Vittorio Fra *

Politecnico di Torino, Turin, 10129, Italy

ARTICLE INFO

Keywords:
State space model
Neuromorphic computing
Edge AI
Model compression

ABSTRACT

Neuromorphic computing is rising as a promising paradigm for efficient AI, leveraging event-driven computation to achieve low-power and high-performance computing. Due to the real-time processing required by edge devices with minimal power consumption, optimizing neuromorphic models for on-edge applications can be crucial to address the issue of power efficiency and resource-constraint devices. This work explores the definition of a neuromorphic state space model and its deployment on non-dedicated hardware. Structured sparsity and quantization techniques are leveraged to enhance the model's efficiency. By compressing synaptic operations and memory footprint, we demonstrate how neuromorphic models can be adapted for on-edge deployment, ensuring low-latency and memory efficient inference. This study highlights the potential of neuromorphic models as a scalable solution for real-world embedded systems with limited resources.

1. Introduction

In an era of pervasive interconnections, intelligent environments demand swift, effective and flexible interactive systems; yet challenges such as latency and bandwidth restrictions, along with concerns regarding data privacy, may hinder their effective implementation through cloud-based services. Edge Artificial Intelligence (AI) offers a potential solution by facilitating data processing at the point of collection, enabling fast responses without relying on remote servers. Such shift is crucial for applications like autonomous driving [1], intelligent power distribution networks [2], and wearable health monitoring devices [3], where real-time capabilities and privacy preservation can be of paramount importance [4]. Edge AI poorly dependent on, or independent of, cloud connectivity has indeed been shown to significantly decrease latency and minimise bandwidth expenses while simultaneously enhancing security measures [5]. Furthermore, the processing of data at the local level ensures operational capacity even in regions with limited connectivity [6]. However, traditional edge AI is confronted with significant challenges concerning energy efficiency and computational complexity, particularly in the context of processing the substantial and variable data streams generated by intelligent environments. Neuromorphic computing, through the adoption of Spiking Neural Networks (SNNs), can be seen as a source of potential solutions to these issues [7,8]. Inspired by the human brain, SNNs process information in a discretized fashion by means of events, also referred to as spikes, in analogy with the action potentials of biological neurons [9]. Furthermore, unlike Artificial Neural Networks (ANNs), SNNs naturally account for temporal dynamics: biologically inspired neuron models integrate spikes over time, and produce output events whether an internal threshold is reached. As explained in [10], these simplified neuron models feature fundamental similarities with State Space Models (SSMs), as an internal state variable is used in both cases to describe the evolution in time depending on the input. However, while SSMs do not inherently imply sparse information transmission based on discrete events, SNNs do, and this event-driven functioning has been shown to allow highly relevant energy saving: neural communication remains quiet in the absence of an informative input, thus reducing the number of operations and the overall energy cost [11–15]. As a consequence, the adoption in the context of edge AI can be easily foreseen to provide significant advantages for continuously operational devices in critical applications such as point-of-care industrial safety and health assistance. Especially the latter, within the wider framework of individual care through real-time monitoring, represents a natural domain of application for neuromoprhic edge AI, thanks to wearable sensors nowadays capable of providing significant amount of data and information in a non-invasive way [16–18].

The efficiency of SNNs takes shape, from the hardware perspective, through various strategies for the development of dedicated platforms [19–24]. Among them, an intensively explored domain is that of Field-Programmable Gate Arrays (FPGAs), which allow custom SNN acceleration with high flexibility and low power consumption [25].

* Corresponding author.
E-mail addresses: benedetto.leto@polito.it (B. Leto), gianvito.urgese@polito.it (G. Urgese), enrico.macii@polito.it (E. Macii), vittorio.fra@polito.it (V. Fra).

https://doi.org/10.1016/j.future.2025.108193
Received 19 March 2025; Received in revised form 29 September 2025; Accepted 7 October 2025
Available online 11 October 2025
0167-739X/© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

FPGA-oriented designs can also be extended to Application-Specific Integrated Circuits (ASICs) [26] or complemented by them [27], as both strategies have proven to offer energy efficiency [28]. Highly promising results can then be achieved through RISC-V-based solutions, for either convolutional architectures [29] or recurrent networks [30]. Nonetheless, the difficult integration of dedicated hardware with components from other application domains [31,32] still hinders extensive application of neuromorphic computing for two main reasons: on the one hand, dedicated architectures able to perform bio-inspired asynchronous and sparse computation have limited availability and fragmented ecosystems, which make their integration and interoperability difficult [33]; on the other hand, interfacing conventional, real-valued, frame-based sensors with SNNs implies a spike-encoding step which typically results in non-trivial additional burdens [34–37].

The most promising answer to twofold obstruction outlined is the investigation and development of SNNs that can be fed with continuous data to also work on hardware platforms other than neurmorphic ones [38–41]. In this work, we adopt this pathway proposing a solution for neuromorphic edge AI relying on non-dedicated hardware. By means of a neuromorphic SSM built redesigning the Legendre Memory Unit (LMU) architecture [42], we tackle the Human Activity Recognition (HAR) task focusing on hand-oriented activities. The SNN-based model we show, entirely relies on Leaky Integrate-and-Fire (LIF) neuron populations to implement all the building blocks of the LMU, resulting in a natively neuromorphic architecture called LIF-based LMU (L^2MU). By means of a tailored encoding layer, it can operate directly with raw sensory data, and we report on its deployment on Raspberry Pi 4B in both full-precision (FP) and compressed versions, with the latter including 8-bit precision and model pruning. By selecting Raspberry Pi 4B as target edge device for the deployment of our L^2MU, we highlight how a general purpose, easily accessible and widely used platform [43] is suitable for applications of neuromorphic principles on non-dedicated hardware.

The successful implementation of the L^2MU to work with raw data on commercial-off-the-shelf (COTS) hardware carries a twofold promising demonstration: i) neuromorphic SSMs can be effectively adopted in smart environments for Internet of Things (IoT) tasks, and ii) neuromorphic edge AI can achieve competitive results with both full precision and quantized models.

2. Background

Smart environments are spaces where interconnected devices can offer improved functionalities through networked monitoring and sensing tools that collaborate in acquiring and processing data [44,45]. As a result, a consistent amount of information is collected in form of time-dependent data from a variety of sources [46–48]. In order to deal with signals produced in such scenarios, sequence modelling architectures are of primary importance, as they can provide unique capabilities to retrieve temporal information, and SSMs represent a preeminent example [49–51]. In the context of edge AI, an additional requirement for these architectures can be compression tolerance, which means they have to effectively operate with possibly reduced numerical precision [52].

2.1. Legendre memory unit (LMU)

In the realm of SSMs, the LMU is characterized by the distinguishing feature of employing the orthogonal basis of the shifted Legendre polynomials for high-dimensional projection of an input $u(t)$ [42]. By means of such basis, a delayed representation $u(t-\theta')$ is produced within a sliding window of duration θ, with $0 \leq \theta' \leq \theta$. The element of degree i of the polynomial basis is defined by Eq. (1) as

$$P_i(x) = (-1)^i \sum_{j=0}^{i} \binom{i}{j} \binom{i+j}{j} (-x)^i \tag{1}$$

and $u(t-\theta')$ is defined by Eq. (2) accordingly:

$$u(t-\theta') \approx \sum_{i=0}^{d-1} P_i\left(\frac{\theta'}{\theta}\right) m_i(t) \tag{2}$$

where the dimension d of the memory state $\mathbf{m}(t)$ identifies the highest order $(d-1)$ of the polynomials in the series expansion. Coherently with the definition of latent space in the SSM domain, the above mentioned $\mathbf{m}(t)$ is described through Eq. (3):

$$\theta \dot{\mathbf{m}}(t) = \mathbf{A}\mathbf{m}(t) + \mathbf{B}u(t) \tag{3}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the specific state space matrices computed, in the case of the LMU, through the Padé approximants following Eq. (4) and Eq. (5):

$$\mathbf{A} = [a]_{ij} \in \mathbb{R}^{d \times d}$$
$$a_{ij} = (2i+1) \begin{cases} -1 & i < j \\ (-1)^{i-j+1} & i \geq j \end{cases} \tag{4}$$

$$\mathbf{B} = [b]_i \in \mathbb{R}^d$$
$$b_i = (2i+1)(-1)^i \tag{5}$$

where $i, j \in [0, d-1]$.

2.2. Leaky integrate-and-fire (LIF) neuron model

At the foundation of SNNs are simplified models of biological neural dynamics [53], whose most common and widely employed example is the LIF neuron. Its simplicity and effectiveness in mimicking the event-based biological communication make this model particularly well-suited for computation [54].

The temporal dynamics of the membrane potential, ultimately responsible for spike emission and consequent information transmission, is described by Eq. (6):

$$\tau_m \frac{dv}{dt} = -v(t) + Ri(t) \tag{6}$$

where $v(t)$ is the membrane potential and τ_m is its decay time, while $i(t)$ is an incoming current and R models the resistance of the neuron membrane. When the membrane potential overcomes the threshold voltage V_{th}, a spike is emitted and $v(t)$ is reset to a value lower than V_{th} in order to satisfy the condition $v(t+dt) < V_{th}$ [53].

In simulation frameworks, a discretized form of the solution to Eq. (6) is typically used, leading, in the case of the `Leaky` model in `snnTorch`[1], to the following Eq. (7):

$$U_t = \beta U_{t-1} + W X_t - S_{t-1}\Theta \tag{7}$$

where X_t is the neuron's input at time t, W is the synaptic weights matrix, U_t is the membrane potential at time t, β is its decay factor, and $S_{t-1} \cdot \Theta$ models the reset mechanism by subtraction, with S_{t-1} being the spiking activity of the neuron at the previous time step and Θ representing the threshold voltage. Coherently with the basic principles of the LIF dynamics, the spiking activity S_t of such discretized models is defined through Eq. (8):

$$S_t = \begin{cases} 1, & if \quad U_t > \Theta \\ 0, & otherwise \end{cases} \tag{8}$$

For the implementation we describe here, `snnTorch` 0.7 was used.

2.3. Human activity recognition (HAR)

According to [55,56], HAR, namely the task of recognizing activities through movement analysis, can be formally defined as the problem of identifying the correct labels $Y_i \in Y = \{Y_1, Y_2, ..., Y_L\}$ for k-dimensional

[1] github.com/jeshraghian/snntorch/

time-series S_j having each dimension $s_k^j = \{x_{t_1}, x_{t_2}, ..., x_{t_T}\}$ made of a fixed number of values in the interval $I = [t_1, t_T]$.

Depending on what signals are monitored, and, accordingly, what sensors are used, different types of HAR can be identified for a variety of applications and domains, including healthcare, sports and smart environments [57–61]. The wide diffusion of wearable devices has made non-invasive monitoring of motion a relevant source of information in the latter, especially through Inertial Measurement Unit (IMU) sensors [60,62]. The Wireless Sensor Data Mining (WISDM) smartphone and smartwatch activity and biometrics dataset, hereinafter referred to as WISDM for simplicity, is a comprehensive collection of Activities of Daily Living (ADL) relying on signals collected through accelerometers and gyroscopes of smart devices [63,64]. It encompasses data from 51 subjects performing 18 different tasks, each of them lasting 3 minutes, divided into three categories: i) non-hand-oriented activities, ii) hand-oriented activities related to general tasks, and iii) hand-oriented activities tied to eating actions.

To address real-time applications in smart environments, samples shorter than the whole duration of 3 minutes are needed to successfully train a classifier, thus a windowing procedure has been applied to the signals from both accelerometer and gyroscope. Consistently with the perception of continuous human-machine interaction [65,66], we set a timescale of 2 s. Focusing on the hand-oriented general activities as in [67], we eventually produced a subset of 7 classes with 31,512 non-overlapping samples, each of them composed of 40 time steps and 6 channels corresponding to the three axes of the accelerometer and the three axes of the gyroscope. We then performed a 60:20:20 partition to define training, validation, and test set respectively.

Compared to the original WISDM dataset, the one we adopted[2] is the result of a reconstruction operation aimed at solving some issues related to non-uniform sampling of the data, absence of some classes, and temporal misalignment between accelerometer and gyroscope signals [68].

2.4. Model compression

Model compression refers to a set of techniques typically adopted in the domain of Deep Learning (DL) to reduce the memory footprint of a model and its computational demands [69–71]. Among them, pruning and quantization are often explored as strategies to, respectively, remove less important neural connections and convert weights and activations to lower-precision representations [72].

2.4.1. Pruning

Pruning is the technique that makes neural network models compact and effective by removing unnecessary connections. The underlying concept is to nullify weights that contribute very little to the model performance, resulting in a sparser version of the original model.

Structured and unstructured pruning are two primary approaches, with the former producing elimination of entire network channels or layers and the latter addressing individual synaptic weights in a more distributed way across the network [73]. Compared to structured pruning, the unstructured one preserves the weights matrices' original dimensions, thus implying a reduced impact on computing speedup unless hardware capable of effectively exploiting sparsity is used.

2.4.2. Quantization

Quantization is the process of converting numerical representations to data types with reduced precision, namely smaller bit widths, and it can involve both weights and activations within a neural network [74]. The typical procedure of transforming 32-bit floating point quantities to 8-bit counterparts leads to significant memory footprint reduction, and it is particularly beneficial to deploy models for edge AI resource-constrained devices.

[2] https://huggingface.co/datasets/neuromorphic-polito/siddha

In the domain of SNNs, quantization can also be used to reduce the precision of neuronal internal states such as the membrane potential and the threshold voltage [75,76].

3. Methodology

Building a neuromorphic model means taking inspiration from discrete and sparse computation. Nonetheless, just as the human brain must account for real-world signals, which can be considered continuous unless we enter the quantum realm, so has every SNN to interface with traditional sensors. In this section, we describe how we implemented our L^{2}MU, and the architecture depicted in Fig. 1, to work with raw non-spiking signals and how we employed it to solve the HAR task on COTS hardware.

3.1. LIF-based LMU (L^{2}MU)

The L^{2}MU is the redesign in terms of neuron populations of the LMU [42]. All the constituent elements, namely the memory, the hidden state, and the internal projection module, are translated into the form of LIF neuron populations. In this process, all the equations governing the interactions among the different components of the LMU have been adapted to handle spiking activities and synaptic currents established across the LIF neurons. The distinguishing inherent neuromorphic feature of the L^{2}MU, compared to the LMU, is indeed that, by definition, the connection of the different building blocks is based on the neural communication through spikes ruled by the LIF neuron equations.

The L^{2}MU architecture is shown in Fig. 1, where the conversion of each LMU constitutive block in neural populations is depicted in detail along with the functional dynamics of the LIF neuron. The latter is implemented through the above mentioned `Leaky` model.

As explained in the original paper [42], the LMU takes an input signal $\mathbf{x}_t$ and produces a hidden state $\mathbf{h}_t$, which is subsequently fed to the memory state $\mathbf{m}_t$. This progression is mediated by $\mathbf{u}_t$, which operates the transformation onto the Legendre polynomial basis through Eq. (9):

$$\mathbf{u}_t = \mathbf{e}_x^\top \mathbf{x}_t + \mathbf{e}_h^\top \mathbf{h}_{t-1} + \mathbf{e}_m^\top \mathbf{m}_{t-1} \tag{9}$$

where $\mathbf{e}_x$, $\mathbf{e}_h$ and $\mathbf{e}_m$ are encoding vectors [42]. In the L^{2}MU, such transformation is translated into Eq. (10) by relying on the spiking activity of the different populations of neurons:

$$\mathbf{u}_{t,curr} = \mathbf{e}_x^\top \mathbf{x}_{t,spk} + \mathbf{e}_h^\top \mathbf{h}_{t-1,spk} + \mathbf{e}_m^\top \mathbf{m}_{t-1,spk} \tag{10}$$

with the consequent spiking activity from the $\mathbf{u}$ population ruled by Eq. (11) as

$$\mathbf{u}_{t,spk} = \begin{cases} 1, & if \quad \mathbf{u}_{t,mem} > \mathbf{u}_{thr} \\ 0, & otherwise \end{cases} \tag{11}$$

being $\mathbf{u}_{thr}$ the threshold voltage to be overcome by the membrane potential $\mathbf{u}_{t,mem}$ for spike emission.

Starting from the input representation through the Legendre polynomial basis, the LMU's memory then evolves in time according to Eq. (12):

$$\mathbf{m}_t = \overline{\mathbf{A}}\mathbf{m}_{t-1} + \overline{\mathbf{B}}\mathbf{u}_t \tag{12}$$

where $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ represent the discretized form of the state space matrices introduced with Eq. (4) and Eq. (5), and they are defined as in Eq. (13):

$$\overline{\mathbf{A}} = (\Delta t/\theta)\mathbf{A} + \mathbf{I}\ ,\ \overline{\mathbf{B}} = (\Delta t/\theta)\mathbf{B} \tag{13}$$

with Δt representing a time step within the window of length θ introduced by Eq. (3). With similar arguments as for Eq. (10) and Eq. (11) with respect to Eq. (9), the fully spiking counterpart of Eq. (12) can be defined for the L^{2}MU as in Eq. (14):

$$\mathbf{m}_{t,curr} = \overline{\mathbf{A}}\mathbf{m}_{t-1,spk} + \overline{\mathbf{B}}\mathbf{u}_{t,spk} \tag{14}$$

with

$$\mathbf{m}_{t,spk} = \begin{cases} 1, & if \quad \mathbf{m}_{t,mem} > \mathbf{m}_{thr} \\ 0, & otherwise \end{cases} \tag{15}$$

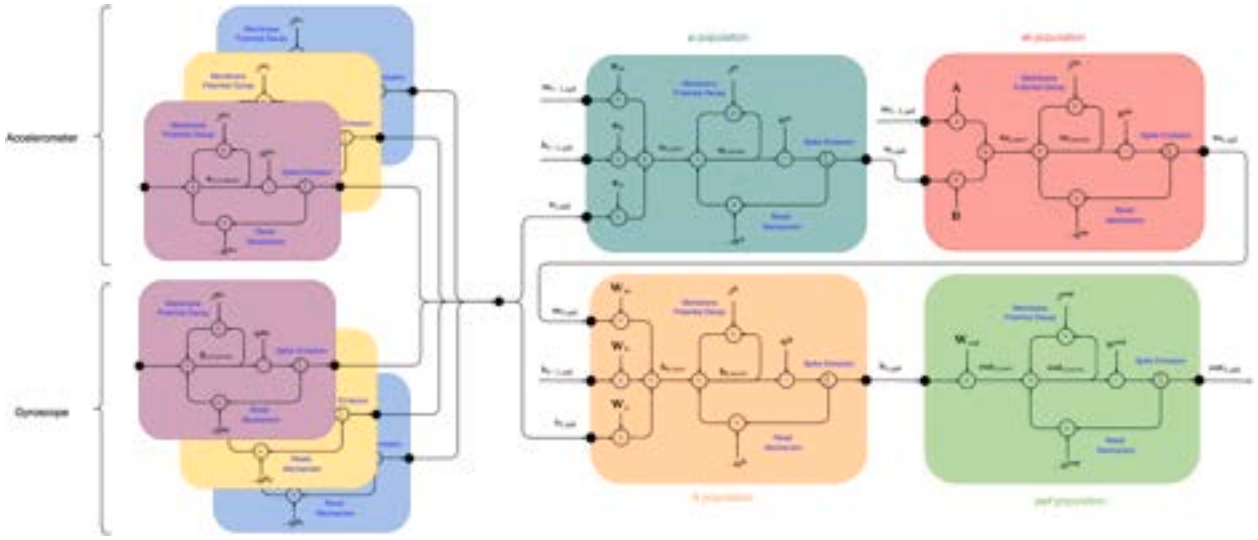


Fig. 1. Structure of the whole SSM including both the encoding module and the L^{2}MU along with the functional dynamics of the LIF neurons constituting each component. The input to the L^{2}MU is denoted as $\mathbf{x}_{t,spk}$, and it feeds the **u** and **h** populations at time step t. For ease of reading, the recurrent connections characterizing the L^{2}MU behaviour are depicted through their final end only, in order to do not overcomplicate the schematic while entirely preserving the fundamental information. The **out** population eventually gathers information from the **h** population to generate the final output.

where $\mathbf{m}_{t,mem}$ denotes the membrane potential and $\mathbf{m}_{thr}$ is the threshold voltage.

Further adopting the same approach, the hidden state originally defined for the LMU [42] by Eq. (16) as

$$\mathbf{h}_t = \tanh(\mathbf{W}_x\mathbf{x}_t + \mathbf{W}_h\mathbf{h}_{t-1} + \mathbf{W}_m\mathbf{m}_t) \tag{16}$$

is translated into a neuron population described by Eq. (17):

$$\mathbf{h}_{t,curr} = \mathbf{W}_x\mathbf{x}_{t,spk} + \mathbf{W}_h\mathbf{h}_{t-1,spk} + \mathbf{W}_m\mathbf{m}_{t,spk} \tag{17}$$

with

$$\mathbf{h}_{t,spk} = \begin{cases} 1, & if \quad \mathbf{h}_{t,mem} > \mathbf{h}_{thr} \\ 0, & otherwise \end{cases} \tag{18}$$

where, coherently with the previous cases, $\mathbf{h}_{t,mem}$ and $\mathbf{h}_{thr}$ represent the population-specific membrane potential and threshold voltage respectively.

3.2. Encoding module

Signal encoding is a crucial step for SNNs to interface with external sensory signals. In order to directly feed our L^{2}MU with raw data as input from conventional sensors, as opposed to event-based sensors, we equipped it with an encoding module. Its goal is to produces spike-encoded, i.e. binarized, signals suitable for the neuromorphic approach. Ultimately, this allows to avoid external, additional spike-encoding procedures that typically burden the development of end-to-end pipelines integrating neuromorphic models.

For our L^{2}MU, we designed an encoder based on channel-specific neurons. The rationale behind this choice is rooted in the fact that each dimension of the input, corresponding to the x-,y- or z-axis of either the accelerometer or the gyroscope, carries information that is distinctive and essential for the accurate interpretation of motion and orientation. Additionally, each of them embeds a different temporal behaviour that has to be taken into account in order to prevent the excessive or too low neural activity possibly arising from inaccurate definition of the neuron dynamics. Channel-specific neurons ensure that the network can effectively process input signals coherent with the unique characteristics of each axis, thus enhancing its sensitivity to the signal dynamics.

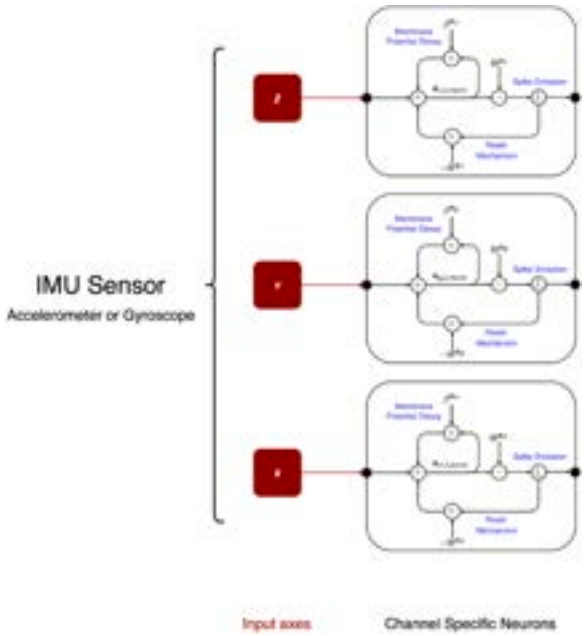


Fig. 2. Detailed schemtic of the basic component of the encoding module for an individual 3-axis IMU sensor. Each axis of the sensor is processed by specific neurons with tailored β and Θ.

The elemental structure of the encoding module we designed is shown in Fig. 2, where the β and Θ parameters specifically defining the neurons of each input channel are highlighted in a similar way as done in Fig. 1.

Referring to Eq. (6), the sensory signals are treated by the encoding module as input currents and converted into spikes, which are then transmitted to the L^{2}MU as $\mathbf{x}_{t,spk}$.

3.3. Hyperparameter optimization

To identify the optimal configuration for our neuromorphic SSM, we performed Hyperparameter Optimization (HPO) experiments through

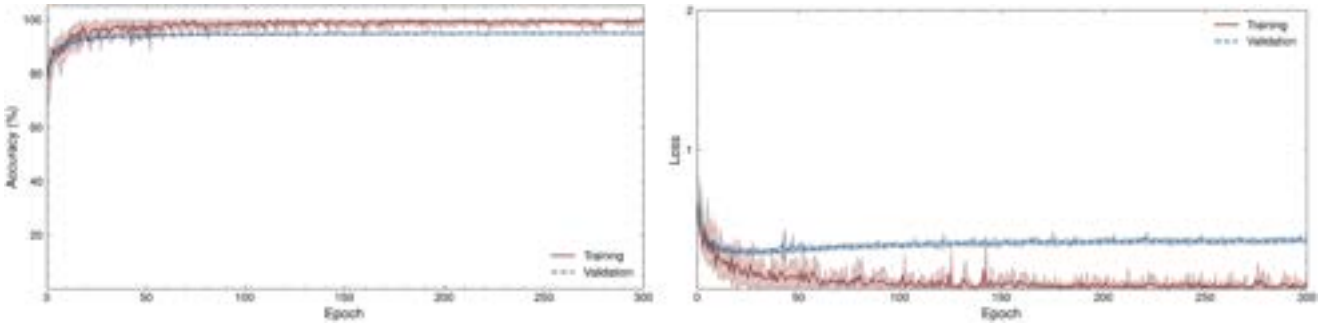


Fig. 3. Learning curves from the ten-time retraining of the optimal model identified through HPO. Solid lines represent mean values, and shaded areas account for standard deviation.

Table 1
Dataset dimensionality for training, validation and test.

Dataset	Samples	Time Steps	Features
Training	18,907	40	6
Validation	6,302	40	6
Test	6,303	40	6

the Neural Network Intelligence (NNI) toolkit[3] by relying on the methodology described in [77] to adapt the procedure presented in [67].

As tuner for the hyperparameter combinations exploration, the built-in *Anneal* algorithm was used, including random reinitialization after every 200 completed trials aimed at mitigating the impact of local minima [78].

The objective metrics to optimize was defined as the best validation accuracy across 300 epochs of training, with the latter performed including classification on the validation set at the end of each epoch and evaluation of the test accuracy after the last epoch. The optimal model was eventually selected at the end of HPO as the one providing the best test accuracy, and a ten-time retraining procedure was performed afterwards in order to statistically assess its classification performance through different seed values. In Fig. 3, the corresponding learning curves are shown for both loss and accuracy.

In Table 1, the dimensionality of the training, the validation, and the test set is reported.

3.4. Model compression

Starting from the retrained best model obtained with the HPO process, we applied pruning and quantization to compress its memory footprint and enhance its sparsity.

3.4.1. Gradual Magnitude Pruning (GMP)

Gradual Magnitude Pruning (GMP) is a model compression strategy that aims to reduce the number of parameters by iteratively removing weights with the smallest magnitudes [79,80]. This method takes advantage of the fact that a relevant number of weights of a trained network can be found to be close to zero and thus have a reduced impact on the network performance. Eliminating these weights by setting them to zero can significantly decrease the model complexity.

In the following, the procedure employed to apply GMP to our SSM is reported:

a) **Sensitivity scan**
By progressively increasing the sparsity of each layer's weights, the sensitivity scan calculates how responsive each layer is to pruning. Let $\mathbf{S}_l$ represent the relevant sparsity level and $\mathbf{W}_l$ represent the collection of weights in layer l:

1. Define a range of sparsity levels:
$\mathbf{S} = \{s_1, s_2, \dots, s_n\}$ with
$s_1 = 0.1,\ s_n = 1.0,\ \Delta s = 0.05$
where s_i represents the sparsity level (fraction of weights to prune) for layer l.
2. After obtaining to s_i sparsity, calculate the validation accuracy $\mathcal{A}_l(s_i)$ for each layer l by scanning over the sparsity values s_i. Select the s_l^* sparsity level that causes the smallest accuracy degradation:
$s_l^* = \arg\max_{s_i} \left(\mathcal{A}_l(s_i)\right)$ such that
$|\mathcal{A}_l(s_i) - \mathcal{A}_{\text{dense}}| \leq \delta$
where $\mathcal{A}_{\text{dense}}$ is the accuracy of the unpruned model and δ is the allowed accuracy degradation.

b) **Fine-grained pruning for each layer**
For each layer l, given the chosen sparsity level s_l^*, perform magnitude-based pruning:

1. Let $\mathbf{W}_l = \{w_1, w_2, \dots, w_m\}$ represent the weights in layer l.
2. Compute the number of weights to prune:
$n_{\text{zeros}} = \left\lfloor s_l^* \cdot m \right\rfloor$
where $m = |\mathbf{W}_l|$ is the total number of weights in layer l.
3. Define the importance of each weight as its absolute value:
$\text{importance}(w_i) = |w_i| \quad \forall w_i \in \mathbf{W}_l$
4. Determine the pruning threshold τ_l by selecting the n_{zeros}-th smallest weight in $\mathbf{W}_l$:
$\tau_l = \text{k-thvalue}(\mathbf{W}_l, n_{\text{zeros}})$
5. Create a binary mask $\mathbf{M}_l$ where weights below the threshold are pruned (i.e., set to zero):
$\mathbf{M}_l = \{m_i = 1 \text{ if } |w_i| > \tau_l,\ 0 \text{ otherwise }\}$
6. Apply the mask to the weights:
$\mathbf{W}_l' = \mathbf{W}_l \odot \mathbf{M}_l$
where $\odot$ denotes element-wise multiplication.

c) **Fine-tuning the pruned model**
After pruning, retrain the resulting model in order to fine-tune the pruned weights $\mathbf{W}_l'$ with the ultimate goal of compensate possible performance degradation. The fine-tuning step minimizes the loss function $\mathcal{L}$ over the remaining weights in the pruned model:

$$\min_{\mathbf{W}'} \mathcal{L}(\mathbf{W}', D) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i, \mathbf{W}'), y_i)$$

where $D = \{(x_i, y_i)\}$ is the dataset, $f(x_i, \mathbf{W}')$ is the model's prediction with the pruned weights, and ℓ is the loss function.

To fine-tune the pruned model, a learning rate equal to 10% of the initial one can be used to enable a more sensitive weights update.

3.4.2. Quantization

For a comprehensive analysis and evaluation of the impact of quantization on our neuromorphic SSM, different conditions have been

[3] https://github.com/microsoft/nni

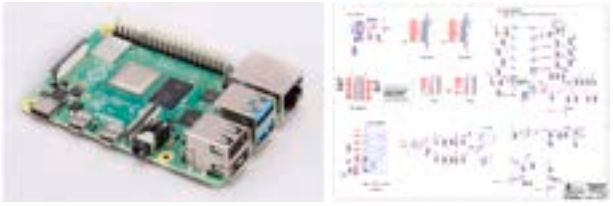


Fig. 4. The Raspberry Pi 4B[4] and its schematic diagram[5].

investigated, applying to the optimal FP model, both before and after pruning, two quantization strategies: Dynamic Quantization (DQ), that adapts only the model's weights at runtime by dynamically determining the scale factor based on the data range observed, and Quantization-Aware Training (QAT), that allows to perform the learning phase using both the model's weights and the neuronal states in their quantized form.

Similarly to the strategy adopted for the post-pruning fine-tuning, a reduced learning rate has been used for QAT, setting it to 1% of the original value. Additionally, it was observed that few epochs were sufficient to obtain low accuracy degradation, thus 10% of the original number of epochs has been used for QAT.

3.5. Model deployment

For the deployment of our neuromorphic SSM, TorchScript via `torch.jit.script` has been used in order to convert the model into an efficient format suitable for execution on embedded hardware. A scripted model in TorchScript is a serialized version of a PyTorch model, with the same operations but optimized performance. Such model can run independently of Python, making it highly suitable for production environments where the Python inference engines can be very disadvantageous in terms of performance. TorchScript indeed enables ahead-of-time compilation fusing operations and optimizing the execution graph.

We also developed a dedicated C++ program for efficient execution and better compatibility with embedded systems.

To further enhance performance, we additionally applied multiple optimization techniques when compiling the inference program. Specifically, we leveraged the following compiler flags:

- `-O3` to enable maximum compiler optimizations for performance;
- `-march=native` to generate optimized code for the specific processor architecture;
- `-ffast-math` to allow aggressive floating-point optimizations, improving execution speed;
- `-fopenmp` to enable parallel execution through OpenMP, utilizing multi-core processing capabilities.

3.6. Testbed

As edge device for the deployment of our L^{2}MU, we opted for a COTS hardware platform like Raspberry Pi 4B due to its adaptability to a multitude of domains [43]. The selected hardware, shown in Fig. 4,[4,5] is characterized by an embedded Broadcom BCM2711 with four 64-bit, 1.5 GHz ARM-A72 cores and 4 GB of DDR4 RAM, thus implying that our model deployment is grounded in the domain of edge AI between mobile AI and tiny AI [81], and validating our neuromorphic SSM for portable devices suitable for real-time, personalized and privacy-preserving applications.[6]

[4] https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[5] https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf

[6] https://mlsysbook.ai/

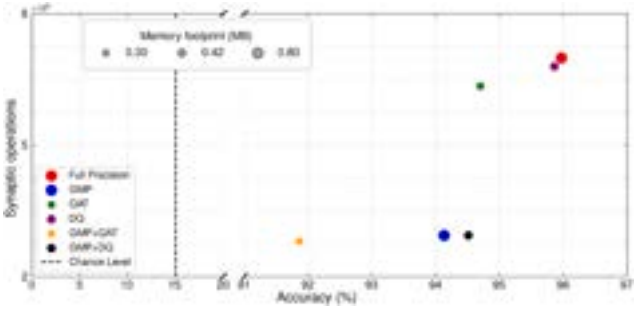


Fig. 5. Overview of the six different versions of the L^{2}MU deployed on Raspberry Pi 4B. Models quantized from FP are labelled with the adopted quantization technique only.

The practical setup employed for testing relies on the assumption that data transmission does not impact on the execution of the model: we considered as decoupled the communication efficiency with the board and the suitability of our neuromorphic SSM for non-dedicated hardware. Consequently, both model and data were pre-loaded onto the Raspberry Pi 4B, which is in turn connected through ethernet to a laptop acting as controller.

In future real-world scenarios, IMUs other than the ones we relied on through data from [63,64] can also be used as signal source, which implies specific conditions to be met in terms of data frequency and possibly requiring fine-tuning of the model. In the test conditions we defined, the input signals are acquired with a frequency of 20 Hz [63,64].

4. Results and discussion

Six different versions of the L^{2}MU have been deployed on Raspberry Pi 4B and extensively characterized, including specific metrics available in the NeuroBench framework [82] for neuromorphic models. Specifically, together with task-related and memory-related quantities, the following ones have been investigated: connection sparsity, activation sparsity, membrane updates, synaptic operations subdivided into effective multiply-accumulate operations (MACs) and effective accumulate operations (ACs), and dense operations. A summary of such characterization is reported in Table 2.

The highest accuracy achieved by the L^{2}MU on Raspberry Pi 4B is 95.97%, given by the FP model. At the other end, the lowest result is provided by the model obtained through QAT of the pruned model, and it is 91.86%. These two models are at the opposite ends also concerning the number of synaptic operations, which is about 59.8% lower for the latter.

In terms of sparsity, referring to either activation or connection sparsity, the most beneficial impact turns out to be given by GMP, both with and without quantization. Particularly looking at connection sparsity, combining it with the accuracy results, the change from 0.04 of the FP model to 0.63 of the pruned models highlights the potential for sparsity exploitation by our neuromoprhic SSM, as one of the pruned model, the GMP + DQ, reaches 94.51% in classification. Furthermore, this model provides such performance while reducing the number of effective MACs by 44.8% and the number of effective ACs by 58.1% with respect to the FP model.

By taking into account the memory footprint as well, a complete overview can be obtained, as reported in Fig. 5.

With an overall analysis, the impact of model compression can be highlighted from three perspectives: first, quantization clearly affects the memory footprint; second, pruning is crucial in improving sparsity; third, the combination of quantization and pruning has a strong impact on the number of synaptic operations. In Fig. 6, a radar plot is presented to summarize such analysis through a comparison of the six L^{2}MU versions with respect to the different metrics.

Table 2
Summary of the different L²MU versions deployed on Raspberry Pi 4B.

	Full Precision (FP)	Gradual Magnitude Pruning (GMP)	Quantization			
			From FP		From GMP	
			QAT	DQ	QAT	DQ
Accuracy (%)	95.97	94.13	94.70	95.86	91.86	94.51
Parameters ($\times 10^3$)	184.91					
Precision (bit)	32		8			
Footprint (MB)	0.80		0.30	0.42	0.30	0.42
Connection Sparsity	0.04	0.63	0.06		0.63	
Activation Sparsity	0.88	0.91	0.89	0.88	0.91	0.90
Membrane Upd. ($\times 10^3$)	25.4	23.3	18.8	25.3	17.5	23.3
Effective MACs ($\times 10^3$)	9.6	5.3	8.1	9.4	4.5	5.3
Effective ACs ($\times 10^3$)	689.3	288.6	626.7	670.1	276.2	288.6
Dense ($\times 10^6$)	7.4					

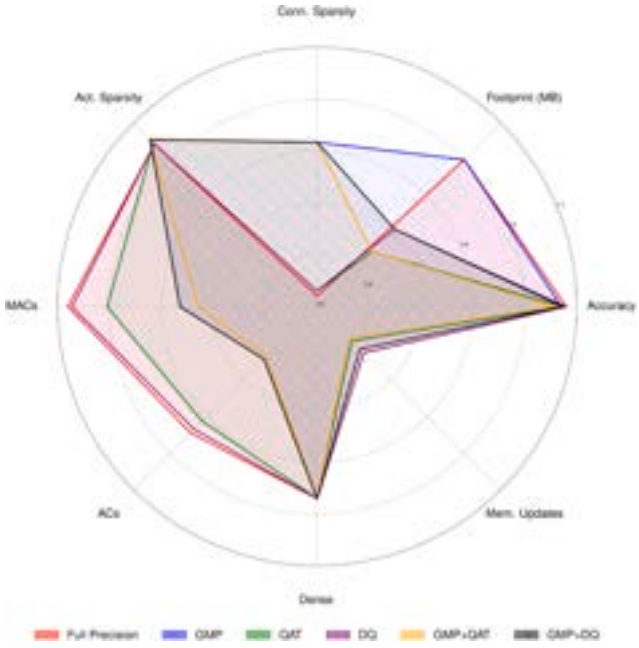


Fig. 6. Radar plot for direct comparison of the six L²MU versions. Models quantized from FP are labelled with the adopted quantization technique only.

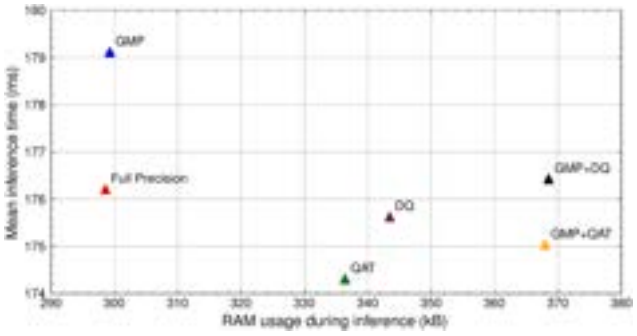


Fig. 7. Comparison of the different L²MU versions during inference in terms of latency and memory usage. Models quantized without pruning are labelled with the adopted quantization technique only.

As it is clear from both Table 2 and Fig. 6, the number of membrane updates is only slightly affected by model compression. The reason for this behaviour can be found in the fact that the adopted techniques do not impact on the neuronal dynamics in terms of temporal evolution and do not involve changes in the number of neurons within the model.

From the real-time, on-edge perspective, the additional level of characterization required to describe the deployed models is presented in Fig. 7, where inference by means of the different L²MU versions is investigated in terms of latency and memory usage.

Combining all the evaluated metrics, the GMP + DQ model provides the best trade-off in the search for an efficient model for edge AI applications. Compared to the FP model, the accuracy drop is indeed of 1.46%, while the memory footprint is reduced by 47.5%, the connection sparsity is increased by 14.75× and the number of synaptic operations is reduced by 57.9%.

4.1. Computational complexity

The neuromorphic nature of the L²MU, namely its being made of LIF neurons, offers an unusual lens through which to evaluate the computational complexity of the model. All the operations can indeed be interpreted in terms of synaptic connections, which in turn implies that the weight matrices are directly related to the neural populations dimensionality. Therefore, the whole analysis can be traced back to the number of neurons, which is, together with the update in time of the neuronal membrane potential, ultimately responsible for the computational complexity.

By means of the general formula $\text{FLOPs} = (2k-1)mn$ for a matrix product $\mathbf{P} = \mathbf{M}_1\mathbf{M}_2$ with $\mathbf{M}_1 \in \mathbb{R}^{m\times k}$ and $\mathbf{M}_2 \in \mathbb{R}^{k\times n}$, and given the all-to-all connection scheme adopted among the different populations along with the connectivity depicted in Fig. 1 according to Eqs. (10, 14 and 17), the number of floating-point operations (FLOPs) corresponding to the synaptic connections of the L²MU can be computed as:

$$
\begin{aligned}
\text{FLOPs}_{\text{syn}} \;= T \cdot [\,& 1 \cdot 1 \cdot \sum_{i=1}^{6} \mathcal{D}_{\text{enc},i} + \left(2 \cdot \sum_{i=1}^{6} \mathcal{D}_{\text{enc},i} - 1\right) \cdot 1 \cdot \mathcal{D}_{\text{u}} \;+ \\
& \left(2 \cdot \mathcal{D}_{\text{h}} - 1\right) \cdot 1 \cdot \mathcal{D}_{\text{u}} \;+ \left(2 \cdot \left(\mathcal{D}_{\text{m}} \cdot d\right) - 1\right) \cdot 1 \cdot \mathcal{D}_{\text{u}} \;+ \\
& + \left(2 \cdot d - 1\right) \cdot \mathcal{D}_{\text{m}} \cdot d \;+ 1 \cdot \mathcal{D}_{\text{u}} \cdot d \;+ \\
& + \left(2 \cdot \sum_{i=1}^{6} \mathcal{D}_{\text{enc},i} - 1\right) \cdot 1 \cdot \mathcal{D}_{\text{h}} \;+ \left(2 \cdot \mathcal{D}_{\text{h}} - 1\right) \cdot 1 \cdot \mathcal{D}_{\text{h}} \;+ \\
& + \left(2 \cdot \left(\mathcal{D}_{\text{m}} \cdot d\right) - 1\right) \cdot 1 \cdot \mathcal{D}_{\text{h}} \;+ \left(2 \cdot \mathcal{D}_{\text{h}} - 1\right) \cdot 1 \cdot \mathcal{D}_{\text{out}}]
\end{aligned}
$$

Table 3
Summary of the optimal hyperparameter values affecting the computational complexity of the L²MU.

		Dimension	Symbol
Encoding module	Population 1	40	$D_{\text{pop},1}$
	Population 2	40	$D_{\text{pop},2}$
	Population 3	40	$D_{\text{pop},3}$
	Population 4	40	$D_{\text{pop},4}$
	Population 5	40	$D_{\text{pop},5}$
	Population 6	40	$D_{\text{pop},6}$
u population		50	D_{u}
m population		$50 \cdot 3$	$D_{\text{m}} \cdot d$
h population		240	D_{h}
out population		7	D_{out}
d		3	d

where T represents the number of time steps of a single sample. In order to account for the operations required by Eq. (7) for the evolution in time of the membrane potentials, an additional term $\text{FLOPs}_{\text{mem}} = T \cdot 4D_p$ must be included for each neuron population p, thus leading to the number of FLOPs for a single-sample inference as defined by Eq. (19):

$$\begin{aligned}\text{FLOPs} &= \text{FLOPs}_{\text{syn}} + \text{FLOPs}_{\text{mem}} \\ &= \text{FLOPs}_{\text{syn}} + \sum_p \left(T \cdot 4D_p\right)\end{aligned} \tag{19}$$

In Table 3, the value for each term of Eq. (19) and Eq. (19) is reported as obtained through the HPO, summarizing the number of neurons for each population together with d, which determines the size of the state space matrices **A** and **B**. Substituting in Eq. (19) and Eq. (19), the resulting number of FLOPs is 14870840.

5. Conclusion

In this work, we reported on the L²MU, a neuromorphic SSM suitable for edge AI applications with COTS hardware and different numeric precisions. With a specific focus on the HAR task with inertial data from wearable devices, we investigated different compression techniques, exploring both pruning and quantization methodologies through various metrics for a comprehensive characterization. Real-time capabilities and performance have been assessed by deploying the L²MU on Raspberry Pi 4B.

Our analysis showed that, compared to the FP model, an almost negligible drop in accuracy, from 95.97% to 95.86%, can be achieved by adopting a DQ strategy that allows to reduce by 47.5% the memory footprint of the model. It also highlighted that the combination of pruning, through GMP, and quantization, by means of DQ, provides a significantly sparser and smaller model capable of achieving a classification accuracy comparable to the FP model with a more than halved number of synaptic operations.

L²MU has been shown to be a versatile architecture for neuromorphic SSMs deployed on COTS hardware for edge AI, and the advantages offered by the pruned and quantized versions presented in this work lay the groundwork for two pathways of future development. On the one hand, sparsity and reduced number of operations can be fully exploited on dedicated hardware; on the other hand, brain-inspired solutions can be further adopted on already commercially available edge devices.

CRediT authorship contribution statement

Benedetto Leto: Writing – review & editing, Writing – original draft, Validation, Investigation, Conceptualization; **Gianvito Urgese:** Writing – review & editing, Supervision, Project administration, Conceptualization; **Enrico Macii:** Resources, Project administration, Funding acquisition; **Vittorio Fra:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gianvito Urgese reports financial support was provided by European Union. Vittorio Fra reports financial support was provided by European Union. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This Research is funded by the European Union - NextGenerationEU Project 3A-ITALY MICS (PE0000004, CUP E13C22001900001, Spoke 6) and the Fluently project with Grant Agreement No. 101058680.
We acknowledge a contribution from the Italian National Recovery and Resilience Plan (NRRP), M4C2, funded by the European Union - NextGenerationEU (Project IR0000011, CUP B51E22000150006, “EBRAINS-Italy”).

References

[1] S.S. Gill, M. Golec, J. Hu, M. Xu, J. Du, H. Wu, G.K. Walia, S.S. Murugesan, B. Ali, M. Kumar, K. Ye, P. Verma, S. Kumar, F. Cuadrado, S. Uhlig, et al., Edge AI: A Taxonomy, Systematic Review and Future Directions, Cluster Comput. 28 (1) (2025) 18. https://doi.org/10.1007/s10586-024-04686-y
[2] M. Rahmati, Edge-AI based multi-criteria optimization framework for dynamic EV charging with real-time grid load, traffic, and user behavior integration, Computing 107 (9) (2025) 178.
[3] A. Rocha, M. Monteiro, C. Mattos, M. Dias, J. Soares, R. Magalhães, J. Macedo, Edge AI for Internet of Medical Things: A literature review, Comput. Elect. Eng. 116 (2024) 109202. https://doi.org/10.1016/j.compeleceng.2024.109202
[4] T.P. Fowdur, D.A. Milovanovic, Z.S. Bojkovic, Intelligent and Sustainable Engineering Systems for Industry 4.0 and Beyond, CRC Press, Boca Raton, 1 edition, Boca Raton, 2025. https://doi.org/10.1201/9781003511298
[5] D. Sharma, S. Sarkar, Enabling Inference and Training of Deep Learning Models for AI Applications on IoT Edge Devices, in: S. Pal, D. De, R. Buyya (Eds.), Artificial Intelligence-based Internet of Things Systems, Springer International Publishing, Cham, 2022, pp. 267–283. Series Title: Internet of Things, https://doi.org/10.1007/978-3-030-87059-1_10
[6] A. Hemmati, P. Raoufi, A.M. Rahmani, et al., Edge artificial intelligence for big data: a systematic review, Neural Comput. Appl. 36 (19) (2024) 11461–11494. https://doi.org/10.1007/s00521-024-09723-w
[7] H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, H. Vincent Poor, et al., Lead federated neuromorphic learning for wireless edge artificial intelligence, Nature Commun. 13 (1) (2022) 4269. https://doi.org/10.1038/s41467-022-32020-w
[8] L. Niedermeier, N. Dutt, J.L. Krichmar, et al., An integrated toolbox for creating neuromorphic edge applications, Neuromorp. Comput. Eng. 5 (1) (2025) 014003. https://doi.org/10.1088/2634-4386/adad0f
[9] W. Maass, Networks of spiking neurons: The third generation of neural network models, Neural Netw. 10 (1997). https://doi.org/10.1016/S0893-6080(97)00011-7
[10] S. Karilanova, S. Dey, A. Özçelikkale, State-Space Model Inspired Multiple-Input Multiple-Output Spiking Neurons, in: 2025 Neuro Inspired Computational Elements (NICE), 2025, pp. 1–9. https://doi.org/10.1109/NICE65350.2025.11065909
[11] S.F. Müller-Cleve, V. Fra, L. Khacef, A. Pequeño-Zurro, D. Klepatsch, E. Forno, D.G. Ivanovich, S. Rastogi, G. Urgese, F. Zenke, C. Bartolozzi, et al., Braille letter reading: A benchmark for spatio-temporal pattern recognition on neuromorphic hardware, Front. Neurosci. 16 (2022) 951164. https://doi.org/10.3389/fnins.2022.951164
[12] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G.A.F. Guerra, P. Joshi, P. Plank, S.R. Risbud, Advancing Neuromorphic Computing with Loihi: A Survey of Results and Outlook, Proc. IEEE 109 (5) (2021) 911–934. https://doi.org/10.1109/JPROC.2021.3067593
[13] C. Mayr, S. Hoeppner, S. Furber, et al., SpiNNaker 2: A 10 Million Core Processor System for Brain Simulation and Machine Learning, 2019, https://doi.org/10.48550/arXiv.1911.02385
[14] H. Bos, D. Muir, Sub-mW Neuromorphic SNN audio processing applications with Rockpool and Xylo, 2022, https://doi.org/10.48550/arXiv.2208.12991
[15] G. Orchard, E.P. Frady, D.B.D. Rubin, S. Sanborn, S.B. Shrestha, F.T. Sommer, M. Davies, Efficient Neuromorphic Signal Processing with Loihi 2, in: IEEE Workshop on Signal Processing Systems (SiPS), 2021-Octob, 2021. https://doi.org/10.1109/SiPS52927.2021.00053

[16] F. Tian, J. Yang, S. Zhao, M. Sawan, et al., NeuroCARE: A generic neuromorphic edge computing framework for healthcare applications, Front. Neurosci. 17 (2023) 1093865. https://doi.org/10.3389/fnins.2023.1093865

[17] K. Aboumerhi, A. Güemes, H. Liu, F. Tenore, R. Etienne-Cummings, et al., Neuromorphic applications in medicine, J. Neural Eng. 20 (4) (2023) 041004. https://doi.org/10.1088/1741-2552/aceca3

[18] S. Bian, M. Magno, Evaluating Spiking Neural Network on Neuromorphic Platform For Human Activity Recognition, in: Proceedings of the 2023 International Symposium on Wearable Computers, ACM, Cancun, Quintana Roo Mexico, 2023, pp. 82–86. https://doi.org/10.1145/3594738.3611369

[19] A. Basu, L. Deng, C. Frenkel, X. Zhang, Spiking Neural Network Integrated Circuits: A Review of Trends and Future Directions, in: 2022 IEEE Custom Integrated Circuits Conference (CICC), 2022, pp. 1–8. https://doi.org/10.1109/CICC53496.2022.9772783

[20] M. Barocci, V. Fra, E. Macii, G. Urgese, Review of open neuromorphic architectures and a first integration in the RISC-V PULP platform, in: 2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), IEEE, 2023, pp. 470–477.

[21] B. Vogginger, A. Rostami, V. Jain, S. Arfa, A. Hantsch, D. Kappel, M. Schäfer, U. Faltings, H.A. Gonzalez, C. Liu, et al., Neuromorphic hardware for sustainable AI data centers, arXivpreprintarXiv:2402.02521 (2024).

[22] D.R. Muir, S. Sheik, The road to commercial success for neuromorphic technologies, Nature communications 16 (1) (2025) 3586.

[23] P.K. Enuganti, B. Sen Bhattacharya, T. Serrano Gotarredona, O. Rhodes, Neuromorphic computing and applications: a topical review, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 15 (2) (2025) e70014.

[24] C. Silvano, D. Ielmini, F. Ferrandi, L. Fiorin, S. Curzel, L. Benini, F. Conti, A. Garofalo, C. Zambelli, E. Calore, et al., A survey on deep learning hardware accelerators for heterogeneous hpc platforms, ACM Comput. Surv. 57 (11) (2025) 1–39.

[25] M. Karamimanesh, E. Abiri, M. Shahsavari, K. Hassanli, A. van Schaik, J. Eshraghian, Spiking neural networks on FPGA: A survey of methodologies and recent advancements, Neural Netw. 186 (2025) 107256.

[26] L. Martis, G. Leone, L. Raffo, P. Meloni, SYNtzulA: Open-Source Hardware for Energy-Efficient Spiking Neural Network Inference, in: Proceedings of the 22nd ACM International Conference on Computing Frontiers: Workshops and Special Sessions, 2025, pp. 70–73.

[27] J. Saussereau, C. Jego, C. Leroux, J.-B. Begueret, Odatix: An open-source design automation toolbox for FPGA/ASIC implementation, SoftwareX 29 (2025) 101970.

[28] T. Zhang, J. Morris, K. Stewart, H.W. Lui, B. Khaleghi, A. Thomas, T. Goncalves-Marback, B. Aksanli, E.O. Neftci, T. Rosing, HyperSpikeASIC: Accelerating Event-Based Workloads With HyperDimensional Computing and Spiking Neural Networks, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 42 (11) (2023) 3997–4010.

[29] S. Manoni, P. Scheffler, A. Di Mauro, L. Zanatta, A. Acquaviva, L. Benini, A. Bartolini, NARS: Neuromorphic Acceleration through Register-Streaming Extensions on RISC-V Cores, in: Proceedings of the 21st ACM International Conference on Computing Frontiers: Workshops and Special Sessions, 2024, pp. 79–82.

[30] Z. Aizaz, J.C. Knight, T. Nowotny, FeNN: A RISC-V vector processor for Spiking Neural Network acceleration, in: 2025 Neuro Inspired Computational Elements (NICE), 2025, pp. 1–7. https://doi.org/10.1109/NICE65350.2025.11065891

[31] J. Clair, G. Eichler, L.P. Carloni, SpikeHard: Efficiency-driven neuromorphic hardware for heterogeneous systems-on-chip, ACM Transactions on Embedded Computing Systems 22 (5s) (2023) 1–22.

[32] D. Kudithipudi, C. Schuman, C.M. Vineyard, T. Pandit, C. Merkel, R. Kubendran, J.B. Aimone, G. Orchard, C. Mayr, R. Benosman, et al., Neuromorphic computing at scale, Nature 637 (8047) (2025) 801–812.

[33] J.E. Pedersen, S. Abreu, M. Jobst, G. Lenz, V. Fra, F.C. Bauer, D.R. Muir, P. Zhou, B. Vogginger, K. Heckel, G. Urgese, S. Shankar, T.C. Stewart, S. Sheik, J.K. Eshraghian, et al., Neuromorphic intermediate representation: A unified instruction set for interoperable brain-inspired computing, Nature Communications 15 (1) (2024) 8122. https://doi.org/10.1038/s41467-024-52259-9

[34] A. Vitale, A. Renner, C. Nauer, D. Scaramuzza, Y. Sandamirskaya, Event-driven Vision and Control for UAVs on a Neuromorphic Chip, in: IEEE International Conference on Robotics and Automation (ICRA), 2021. https://doi.org/10.1109/ICRA48506.2021.9560881

[35] G. Datta, S. Kundu, P.A. Beerel, Training Energy-Efficient Deep Spiking Neural Networks with Single-Spike Hybrid Input Encoding, in: International Joint Conference on Neural Networks (IJCNN), 2021. https://doi.org/10.1109/IJCNN52387.2021.9534306

[36] E. Forno, V. Fra, R. Pignari, E. Macii, G. Urgese, Spike encoding techniques for IoT time-varying signals benchmarked on a neuromorphic classification task, Front. Neurosci. 16 (2022). https://doi.org/10.3389/fnins.2022.999029

[37] S. Bian, E. Donati, M. Magno, et al., Evaluation of Encoding Schemes on Ubiquitous Sensor Signal for Spiking Neural Network, IEEE Sens. J. 24 (21) (2024) 35008–35018. https://doi.org/10.1109/JSEN.2024.3453927

[38] R.V.W. Putra, M. Shafique, SpikeDyn: A Framework for Energy-Efficient Spiking Neural Networks with Continual and Unsupervised Learning Capabilities in Dynamic Environments, in: 58th ACM/IEEE Design Automation Conference (DAC), 2021. https://doi.org/10.1109/DAC18074.2021.9586281

[39] F. Tian, J. Yang, S. Zhao, M. Sawan, NeuroCARE: A generic neuromorphic edge computing framework for healthcare applications, Front. Neurosci. 17 (2023). https://doi.org/10.3389/fnins.2023.1093865

[40] G. Urgese, A. Rios-Navarro, A. Linares-Barranco, T.C. Stewart, K. Michmizos, Editorial: Powering the next-generation IoT applications: new tools and emerging technologies for the development of Neuromorphic System of Systems, Front. Neurosci. 17 (2023). https://doi.org/10.3389/fnins.2023.1197918

[41] V. Fra, A. Pignata, R. Pignari, E. Macii, G. Urgese, Neu-BrAuER: A Neuromorphic Braille Letters Audio-Reader for Commercial Edge Devices, in: R. Meo, F. Silvestri (Eds.), Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2137, Springer Nature Switzerland, Cham, 2025, pp. 51–60. Series Title: Communications in Computer and Information Science, https://doi.org/10.1007/978-3-031-74643-7_5

[42] A.R. Voelker, I. Kajic, C. Eliasmith, Legendre memory units: Continuous-time representation in recurrent neural networks, Adv. Neural Inf. Process. Syst. 32 (NeurIPS) (2019) 0–5.

[43] S.E. Mathe, H.K. Kondaveeti, S. Vappangi, S.D. Vanambathina, N.K. Kumaravelu, A comprehensive review on applications of Raspberry Pi, Comput. Sci. Rev. 52 (2024) 100636.

[44] D.J. Cook, S.K. Das, Overview, John Wiley & Sons, Ltd, 2004, pp. 1–10. https://doi.org/10.1002/047168659X.ch1

[45] D.M. El-Din, A.E. Hassanein, E.E. Hassanien, Smart environments concepts, applications, and challenges, Mach. Learn. Big Data Anal. Paradigms: Anal. Appl. Challenges 77 (2021) 493–519.

[46] W. Gomaa, M.A. Khamis, A perspective on human activity recognition from inertial motion data, Neural Comput. Appl. 35 (28) (2023) 20463–20568. https://doi.org/10.1007/s00521-023-08863-9

[47] N.N. Alajlan, D.M. Ibrahim, Deep Smart Cities: A Review of Smart Cities Applications-Based an Inference of Deep Learning Upon IoT Devices, J. Adv. Res. Appl. Sci. Eng. Technol. 47 (2) (2024) 94–120. https://doi.org/10.37934/araset.47.2.94120

[48] Y.B. Dhiab, M.O.-E. Aoueileyine, M. Abdelkader, R. Bouallegue, Edge-Based Human Activity Recognition: A Novel Approach Using Spectral Analysis and Deep Learning, in: 2024 International Wireless Communications and Mobile Computing (IWCMC), IEEE, Ayia Napa, Cyprus, 2024, pp. 1734–1739. https://doi.org/10.1109/IWCMC61514.2024.10592539

[49] A. Klushyn, R. Kurle, M. Soelch, B. Cseke, P. van der Smagt, Latent matters: Learning deep state-space models, Adv. Neural Inf. Process. Syst. 34 (2021) 10234–10245.

[50] M.-I. Stan, O. Rhodes, Learning long sequences in spiking neural networks, Sci. Rep. 14 (1) (2024) 21957. https://doi.org/10.1038/s41598-024-71678-8

[51] Z. Wang, F. Kong, S. Feng, M. Wang, X. Yang, H. Zhao, D. Wang, Y. Zhang, Is mamba effective for time series forecasting?, Neurocomputing 619 (2025) 129178.

[52] S. Abreu, J.E. Pedersen, K.M. Heckel, A. Pierro, et al., Q-S5: Towards Quantized State Space Models, 2024, https://doi.org/10.48550/arXiv.2406.09477

[53] W. Gerstner, W.M. Kistler, Spiking neuron models: single neurons, populations, plasticity, Cambridge University Press, Cambridge, U.K. ; New York, Cambridge, U.K. ; New York, 2002.

[54] E.M. Izhikevich, Dynamical Systems in Neuroscience, The MIT Press, 2006. https://doi.org/10.7551/mitpress/2526.001.0001

[55] O.D. Lara, M.A. Labrador, A survey on human activity recognition using wearable sensors, IEEE Commun. Surv. Tut. 15 (3) (2012) 1192–1209.

[56] S.G. Dhekane, T. Ploetz, Transfer Learning in Sensor-Based Human Activity Recognition: A Survey, ACM Computing Surveys (2025).

[57] Z. Hussain, M. Sheng, W.E. Zhang, Different approaches for human activity recognition: A survey, arXiv preprint arXiv:1906.05074 (2019).

[58] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, Y. Liu, Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities, ACM Comput. Surv. (CSUR) 54 (4) (2021) 1–40.

[59] F. Gu, M.-H. Chung, M. Chignell, S. Valaee, B. Zhou, X. Liu, A survey on deep learning for human activity recognition, ACM Comput. Surv. (CSUR) 54 (8) (2021) 1–34.

[60] W. Gomaa, M.A. Khamis, A perspective on human activity recognition from inertial motion data, Neural Comput. Appl. 35 (28) (2023) 20463–20568.

[61] W. Qi, X. Xu, K. Qian, B.W. Schuller, G. Fortino, A. Aliverti, A Review of AIoT-based Human Activity Recognition: From Application to Technique, IEEE J. Biomed. Health Inf. 29 (2024).

[62] E. Ramanujam, T. Perumal, S. Padmavathi, Human Activity Recognition with Smartphone and Wearable Sensors using Deep Learning Techniques: A Review, IEEE Sens. J. 21 (2021). https://doi.org/10.1109/JSEN.2021.3069927

[63] G.M. Weiss, WISDM Smartphone and Smartwatch Activity and Biometrics Dataset, UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set 7 (2019).

[64] G.M. Weiss, K. Yoneda, T. Hayajneh, Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living, IEEE Access 7 (2019). https://doi.org/10.1109/ACCESS.2019.2940729

[65] R.B. Miller, Response time in man-computer conversational transactions, in: Proceedings of the December 9-11, 1968, fall joint computer conference, part I, 1968.

[66] P. Popovski, F. Chiariotti, K. Huang, A.E. Kalør, M. Kountouris, N. Pappas, B. Soret, A perspective on time toward wireless 6G, Proc. IEEE 110 (2022) 1116–1146.

[67] V. Fra, E. Forno, R. Pignari, T.C. Stewart, E. Macii, G. Urgese, Human activity recognition: suitability of a neuromorphic approach for on-edge AIoT applications, Neuromorp. Comput. Eng. 2 (1) (2022) 014006. Publisher: IOP Publishing, https://doi.org/10.1088/2634-4386/ac4c38

[68] M. Heydarian, T.E. Doyle, rWISDM: Repaired WISDM, a Public Dataset for Human Activity Recognition, arXiv preprint arXiv:2305.10222 (2023).

[69] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv preprint arXiv:1510.00149 (2015).

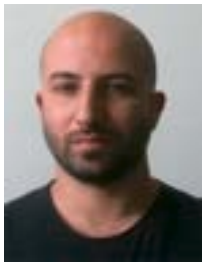
[70] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient DNNs, Adv. Neural Inf. Process. Syst. 29 (2016) 1387–1395.

[71] D. Liu, H. Kong, X. Luo, W. Liu, R. Subramaniam, Bringing AI to edge: From deep learning's perspective, Neurocomputing 485 (2022) 297–320.

[72] A. Klemetti, M. Raatikainen, L. Myllyaho, T. Mikkonen, J.K. Nurminen, Systematic literature review on cost-efficient deep learning, IEEE Access 11 (2023) 90158–90180.
[73] H. Cheng, M. Zhang, J.Q. Shi, A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations, IEEE Trans. Pattern Anal. Mach. Intell. 46 (2024) 10558–10578.
[74] T. Liang, J. Glossner, L. Wang, S. Shi, X. Zhang, Pruning and quantization for deep neural network acceleration: A survey, Neurocomputing 461 (2021) 370–403.
[75] R.V.W. Putra, M. Shafique, Q-spinn: A framework for quantizing spiking neural networks, in: 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, 2021, pp. 1–8.
[76] S. Venkatesh, R. Marinescu, J.K. Eshraghian, SQUAT: stateful quantization-aware training in recurrent spiking neural networks, in: 2024 Neuro Inspired Computational Elements Conference (NICE), IEEE, 2024, pp. 1–10.
[77] V. Fra, Application-oriented automatic hyperparameter optimization for spiking neural network prototyping, arXiv preprint arXiv:2502.12172 (2025).
[78] E. Forno, A. Acquaviva, Y. Kobayashi, E. Macii, G. Urgese, A Parallel Hardware Architecture For Quantum Annealing Algorithm Acceleration, in: 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), 2018-Octob, IEEE, 2018, pp. 31–36. ISSN: 23248440, https://doi.org/10.1109/VLSI-SoC.2018.8644777
[79] M. Zhu, S. Gupta, To prune, or not to prune: exploring the efficacy of pruning for model compression, arXiv preprint arXiv:1710.01878 (2017).
[80] D. Kuznedelev, E. Kurtic, E. Iofinova, E. Frantar, A. Peste, D. Alistarh, Accurate neural network pruning requires rethinking sparse optimization, arXiv preprint arXiv:2308.02060 (2023).
[81] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, S. Han, Tiny machine learning: Progress and futures [feature], IEEE Circuits Syst. Mag. 23 (3) (2023) 8–34.
[82] J. Yik, K. Van den Berghe, D. den Blanken, Y. Bouhadjar, M. Fabre, P. Hueber, W. Ke, M.A. Khoei, D. Kleyko, N. Pacik-Nelson, et al., The neurobench framework for benchmarking neuromorphic computing algorithms and systems, Nature Commun. 16 (1) (2025) 1545.



Benedetto Leto is a Research Fellow at Politecnico di Torino, where he received a M.Sc. in Computer Engineering in 2024 and a B.Sc. in Computer Engineering in 2021. In the Electronic Design Automation (EDA) Group, his main research interests are neuromorphic computing and AIoT applications.



Gianvito Urgese is an Assistant Professor at Politecnico di Torino, where he received a Ph.D. in Computer and Systems Engineering in 2016. His main research interests are neuromorphic computing and engineering, parallel and heterogeneous architectures, AIoT application development, and algorithm optimisation focused on Bioinformatics and embedded systems domains.



Enrico Macii is a Full Professor of Computer Engineering at Politecnico di Torino. He holds a Laurea degree in electrical engineering from the Politecnico di Torino, a Laurea degree in Computer Science from the Università di Torino, and a Ph.D. Degree in Computer Engineering from the Politecnico di Torino. His research interests are in the design of electronic digital circuits and systems, with a particular emphasis on low power consumption.



Vittorio Fra is a Researcher and Assistant Professor at Politecnico di Torino in the Interuniversity Department of Regional and Urban Studies and Planning (DIST). He holds a B.Sc. in Physical Engineering and a M.Sc. in Nanotechnologies for ICTs, received from Politecnico di Torino where he also obtained his Ph.D. in Physics. In the Electronic Design Automation (EDA) Group, he focuses his activity on neuromorphic computing.