LAST-PAIN: Learning Adaptive Spike Thresholds for Low Back Pain Biosignals Classification

Freek Hens, Mohammad Mahdi Dehshibi, *Senior Member, IEEE*, Leila Bagheriye, *Member, IEEE*, Ana Tajadura-Jiménez, and Mahyar Shahsavari, *Member, IEEE*

***Abstract*— Spiking neural networks (SNNs) present the potential for ultra-low-power computation, especially when implemented on dedicated neuromorphic hardware. However, a significant challenge is the efficient conversion of continuous real-world data into the discrete spike trains required by SNNs. In this paper, we introduce Learning Adaptive Spike Thresholds (LAST), a novel, trainable encoding strategy designed to address this challenge. The LAST encoder learns adaptive thresholds to transform continuous signals of varying dimensionality–ranging from time series data to high dimensional tensors–into sparse spike trains. Our proposed encoder effectively preserves temporal dynamics and adapts to the characteristics of the input. We validate the LAST approach in a demanding healthcare application using the EmoPain dataset. This dataset contains multimodal biosignal analysis for assessing chronic lower back pain (CLBP). Despite the dataset's small sample size and class imbalance, our LAST-driven SNN framework achieves a competitive Matthews Correlation Coefficient of 0.44 and an accuracy of 80.43% in CLBP classification. The experimental results also indicate that the same framework can achieve an F1-score of 0.65 in detecting protective behaviour. Furthermore, the LAST encoder outperforms conventional rate and latency-based encodings while maintaining sparse spike representations. This achievement shows promises for energy-efficient and real-time biosignal processing in resource-limited environments.**

***Index Terms*— Spike train encoder, spiking recurrent neural network, EmoPain, chronic pain, biosignal analysis.**

I. Introduction

SPIKING Neural Networks (SNNs) have gained increasing attention due to their potential for energy-efficient computation and their inherent ability to process temporal information. A fundamental challenge in realising the full potential of SNNs lies in converting continuous signals into discrete spike trains – the native language of these networks. While various encoding methods exist, including rate-based [1], time-to-first-spike [2], relative spike latency [3], and delta modulation [4], these approaches often struggle to effectively preserve temporal dynamics, adapt to varying signal characteristics, and maintain robustness against noise, particularly when dealing with signals that exhibit non-stationarity, high dimensionality, and complex temporal patterns. These limitations restrict the application of SNNs to complex real-world datasets.

To address these limitations, we introduce Learning Adaptive Spike Thresholds (LAST), a novel, trainable encoder designed for converting continuous signals into spike trains. LAST offers several key advantages: (i) it is applicable to signals of various dimensions, (ii) it learns signal-specific encoding strategies, (iii) it adapts to varying signal characteristics, (iv) it preserves temporal dynamics, and (v) it controls spike density through learnable parameters. Importantly, LAST is designed to be compatible with any SNN architecture, providing a versatile encoding solution for diverse applications.

To demonstrate the effectiveness and robustness of LAST, we chose a challenging real-world application, i.e., biosignal analysis using the EmoPain dataset [5]. This dataset, containing surface electromyography (sEMG) and inertial measurement unit (IMU) data, presents several significant challenges, including multimodal signals, a small sample size, class imbalance, and complex temporal patterns. By applying LAST to EmoPain, we seek to validate its functionalities within a complex scenario, wherein the combination of these challenges serves as a rigorous assessment of the encoder's robustness and adaptability.

In our experiments with the EmoPain dataset, we used LAST in conjunction with a Spiking Recurrent Neural Network (SRNN) and an ensemble method. This specific implementation serves as one example of how LAST can be integrated with SNNs; however, LAST is designed to be compatible with a wide range of SNN architectures.

In this paper, we present two key contributions:

1) The novel Learning Adaptive Spike Thresholds (LAST) encoder for converting continuous signals into spike trains, which is applicable to various signal dimensions and SNN architectures.

Received 5 August 2024; revised 4 February 2025; accepted 25 February 2025. Date of publication 28 February 2025; date of current version 10 March 2025. This work was supported by European Union's Horizon 2020 Program for the BODYinTRANSIT Project under Grant 101002711. *(Freek Hens and Mohammad Mahdi Dehshibi contributed equally to this work.) (Corresponding author: Mohammad Mahdi Dehshibi.)*

Freek Hens, Leila Bagheriye, and Mahyar Shahsavari are with the Donders Institute for Brain, Cognition and Behavior, Radboud University Nijmegen, 6525 XZ Nijmegen, The Netherlands (e-mail: freek.hens@outlook.com).

Mohammad Mahdi Dehshibi and Ana Tajadura-Jiménez are with the Department of Computer Science and Engineering, Universidad Carlos III de Madrid, 28911 Leganés, Spain (e-mail: mohammad.dehshibi@yahoo.com).

Digital Object Identifier 10.1109/TNSRE.2025.3546682

2) A demonstration of LAST's effectiveness and robustness in a challenging real-world application using the EmoPain dataset, which involves multimodal signals with small sample size and class imbalance.

The remainder of this paper is organised as follows: Section II reviews related work in continuous signal encoding, previous approaches to spike train generation, and their applications in biosignal processing. Section III details our proposed LAST encoder and its implementation within a complete classification framework. Section IV presents our experimental validation using the EmoPain dataset and compares our results with existing approaches. Finally, Section V concludes the paper and discusses future research directions.

II. Literature Review

A. Spike Encoding Methods and Challenges

Spiking Neural Networks (SNNs) have shown great promise, thanks to their biological plausibility and lower power consumption compared to conventional machine learning methods [6]. However, a critical challenge in leveraging SNNs lies in converting continuous signals into discrete spike trains, a process particularly relevant for applications involving complex real-world signals, such as biosignals. This process is particularly crucial for real-time applications and hardware implementations, where efficient and accurate signal encoding directly impacts system performance.

Rate-based encoding [1], which represents signals through the average firing rate of neurons, is known for its robustness and energy efficiency. However, it struggles to capture rapid changes in signals, potentially missing crucial information [7]. Temporal encoding methods focus on the precise timing of spikes. While latency coding [3] offers high temporal resolution, it can be susceptible to noise [8]. Phase coding boasts resilience to noise but requires complex synchronisation mechanisms [9], [10].

Similar trade-offs exist with Send-on-Delta, threshold-based representation, Ben's spiker algorithm, burst encoding, step-forward, and moving window approaches. While these methods offer unique advantages such as precise spike generation or direct encoding of analogue signals, they often come at the cost of increased computational complexity or implementation challenges [11]. The computational efficiency of these methods becomes particularly crucial when considering deployment on wearable devices or in real-time processing scenarios.

B. Challenges in Complex Signal Analysis

Complex real-world signals present unique challenges for spike encoding methods. Using biosignals as an example, surface electromyography (sEMG) and inertial measurement unit (IMU) data exhibit characteristics that make encoding particularly challenging: signal contamination from various sources, including motion artefacts and crosstalk from nearby physiological processes [12], inter- and intra-individual variability due to physiological differences or factors such as fatigue and stress [13].

Beyond these challenges, such signals are often high-dimensional and multi-channel, with a non-stationary nature. For instance, sEMG data varies due to differences in muscle anatomy between individuals, while IMU signals present challenges such as drift and integration errors, sensitivity to sensor placement, and the complexity of interpreting 3D motion data [14].

C. Existing Approaches and Limitations

In the analysis of complex signals using SNNs, various signal-to-spike conversion methods have been proposed. One prominent method is delta-encoding, which converts signals into spike trains by encoding changes in signal amplitude over time [15]. While this method is beneficial for its simplicity and efficiency in capturing temporal dynamics, it does not fully exploit the spatial information present in high-dimensional signals.

Another approach involves using variational mode decomposition (VMD) [16] to pre-process signals by decomposing them into band-limited modes before conversion into spike trains. While VMD can effectively handle controlled settings, it struggles with the complex dynamics observed in real-world scenarios, particularly when real-time processing is required. This is because factors such as network interactions and physiological variability can significantly influence signals in real-world applications [17].

Recently, there has been increasing interest in learnable encoders for multimodal signal analysis in SNNs [16], [18]. These methods offer the flexibility to adapt to specific spatial, temporal, and cross-modal characteristics of signals, potentially outperforming simpler encoding techniques. However, existing learnable encoders frequently fail to sufficiently tackle the interconnected challenges of high temporal resolution, noise robustness, and computational efficiency that are essential for the real-time analysis of complex biosignals. Additionally, many current approaches do not adequately address the requirements for real-time processing and hardware implementation, crucial factors for practical applications.

These requirements–retaining temporal dynamics, resisting noise, adapting to signal characteristics, and maintaining computational efficiency for real-time processing–motivate the development of our LAST encoder. LAST addresses these critical challenges by learning adaptive thresholds for spike generation, allowing it to adjust dynamically to the input signal characteristics.

III. Proposed Method

This section details the proposed two-stage approach for classifying chronic lower back pain using biosignals. In the first stage, a novel learnable encoder – Learning Adaptive Spike Thresholds (LAST) – converts biosignals (sEMG, Joint Angle, and Joint Energy) into spike trains. In the second stage, independent spiking recurrent neural networks are used within a multi-stream ensemble classification framework to analyse the encoded spike trains. This ensemble approach

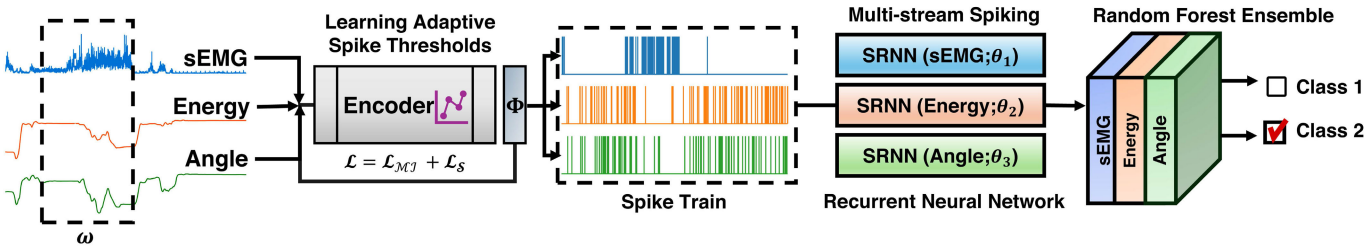


Fig. 1. A schematic of the proposed architecture, including the Learning Adaptive Spike Thresholds (LAST) encoder and Spiking Recurrent Neural Network (SRNN), which are responsible for classifying data from the EmoPain [5] database. The multimodal biosignals consist of sEMG sensors and derived features (i.e., Joint Energy and Joint Angle) from IMU sensors. These continuous signals convert into spike trains using the LAST encoder and are subsequently classified using an ensemble of SRNNs. The ensemble prediction is generated using a Random Forest meta classifier.

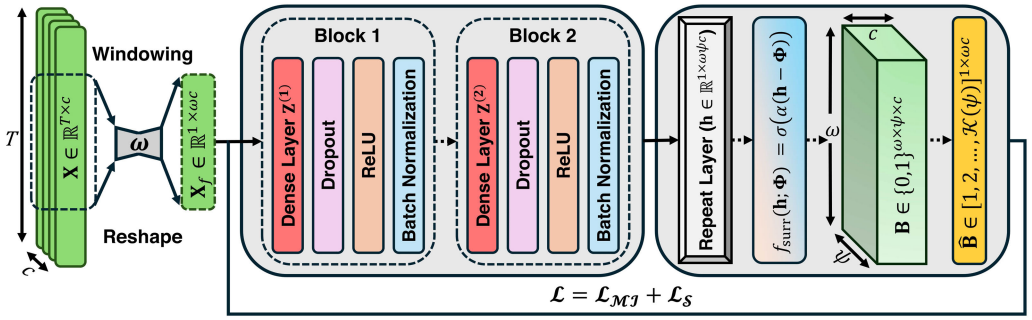


Fig. 2. The LAST architecture consists of feature extraction and feature-to-spike conversion modules. Here, $\mathcal{L}$ is the encoder loss function, formulated in Eq. (4).

combines the predictions from each modality-specific SRNN using a Random Forest classifier to perform classification. Figure 1 illustrates a schematic of the proposed architecture.

Let $\mathbf{X} \in \mathbb{R}^{T \times c}$ denote the input data, where T is the number of time steps and c is the number of channels. To maintain consistency in notation, we refer to the biosignal input dimensions as "channels," irrespective of whether they represent physical sensor channels or derived features. The data is segmented into windows of size ω time steps, resulting in a collection of windowed data points denoted by $\mathbf{X}^{(\omega)} \in \mathbb{R}^{\omega \times c}$ for each window ω.

A. LAST: Learning Adaptive Spike Thresholds

The LAST architecture, depicted in Fig. 2, consists of feature extraction and feature-to-spike conversion modules. The feature extraction module comprises two consecutive blocks, each containing a dense layer, dropout, ReLU activation, and batch normalisation. The ReLU activation function ($\text{ReLU}(\cdot) = \max(0, \cdot)$) introduces non-linearity, allowing the network to learn complex patterns. Dropout helps prevent overfitting by randomly setting a fraction of input units to 0 during training. Batch normalisation improves the stability and performance of the neural network by normalising the inputs to each layer.

To prepare for feature extraction, we reshape $\mathbf{X}^{(\omega)}$ into a feature vector $\mathbf{X}_f \in \mathbb{R}^{1 \times \omega c}$. The first block processes $\mathbf{X}_f$ through a dense layer $\mathbf{Z}^{(1)}$, resulting in $\mathbf{Z}^{(1)} \in \mathbb{R}^{1 \times \ell}$, where ℓ denotes the number of neurons in the layer. The second block applies similar operations to the output of the first block, producing $\mathbf{Z}^{(2)} \in \mathbb{R}^{1 \times \ell}$.

The feature-to-spike conversion module of the LAST converts the extracted features into surrogate spiking activity. This process begins with an element-wise repetition operation on the output of the second block using Eq. (1), expanding it to $\mathbf{h} \in \mathbb{R}^{1 \times \omega\psi c}$, where ψ represents the number of spikes allowed at each time step.

$$\mathbf{h} = \text{Repeat}(\mathbf{Z}^{(2)}, \psi) \tag{1}$$

A learnable threshold vector, denoted as $\Phi \triangleq [\phi_1, \phi_2, \cdots, \phi_{\omega\psi c}]^\top$, is incorporated into the encoding function $f : \mathbb{R} \to \{0, 1\}$. This function is designed as a differentiable surrogate in Eq. (2) to generate binary spike trains $\mathbf{B} \in \{0, 1\}^{\omega \times \psi \times c}$. Each threshold, denoted by $\phi_i \sim \mathcal{U}(a, b)$, $\forall i \in \{1, 2, \cdots, \omega\psi c\}$, is initialised from a uniform distribution with hyperparameters $a, b \in \mathbb{R}^+$.

$$f_{\text{surr}}(\mathbf{h}; \Phi) = \sigma(\alpha(\mathbf{h} - \Phi)) \tag{2}$$

where $\sigma(\cdot)$ is the sigmoid function and α controls the slope of the surrogate function. When $\alpha \to \infty$, the function approaches a true binary step function, mimicking the behaviour of real spiking neurons.

To align the dimensions of the spike train with the input for subsequent computations, we reshape $\mathbf{B}$ into $\hat{\mathbf{B}} \in \{1, 2, \cdots, \mathcal{K}(\psi)\}^{1 \times \omega c}$ using Eq. (3). Here, $\mathcal{K}(\psi)$ denotes the number of possible combinations arising from summing the activations across ψ spike positions at each time step and channel, capturing the temporal information about the order of spikes within $\hat{\mathbf{B}}$.

$$\hat{\mathbf{B}}_i = \sum_{j=1}^{\psi} p_j \mathbf{B}_{i,j,k}, \quad \forall i \in \{1, \cdots, \omega\}, k \in \{1, \cdots, c\} \tag{3}$$

where p_j represents the weight associated with the j-th spike position. This process incorporates the temporal information about the order of spikes into the final representation, as different spike orders will lead to different weighted sums and, consequently, different entries in $\hat{\mathbf{B}}$.

We introduce a customised loss function to update the parameters of the LAST encoder. This function consists of the Mutual Information term, inspired by [19], and the Sparsity term. The Mutual Information term ($\mathcal{L}_{\mathcal{MI}}$) measures the extent to which the encoder captures relevant information from the input data ($\mathbf{X}_f$) at multiple levels of representation, i.e., the encoded information in the hidden layers ($\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$) and encoded spike train ($\hat{\mathbf{B}}$). Meanwhile, the Sparsity term ($\mathcal{L}_{\mathcal{S}}$) serves as a regularisation term, encouraging the generation of sparse spike trains that efficiently represent the input data, promoting resource efficiency and robustness against overfitting. Formally, the loss function is formulated as in Eq. (4).

$$\begin{aligned} \mathcal{L}_{\mathcal{MI}} &= -\frac{1}{3}\left(\mathcal{I}\left[\mathbf{X}_f;\mathbf{Z}^{(1)}\right]+\mathcal{I}\left[\mathbf{X}_f;\mathbf{Z}^{(2)}\right]+\mathcal{I}\left[\mathbf{X}_f;\hat{\mathbf{B}}\right]\right), \\ \mathcal{L}_{\mathcal{S}} &= \lambda\cdot||\mathbf{X}_f-\hat{\mathbf{B}}||_1, \\ \mathcal{L} &= \mathcal{L}_{\mathcal{MI}}+\mathcal{L}_{\mathcal{S}} \end{aligned} \tag{4}$$

where $\mathcal{I}[\cdot;\cdot]$ computes the Mutual Information between tensors, $||\cdot||_1$ is the L1 norm, and λ is the strength coefficient controlling the level of sparseness imposed on the internal representations.

To calculate Mutual Information, the encoded information in the hidden layers and spike train need to have the same dimensions as the input data. We reshape the hidden layer outputs when their dimensions differ. If $\ell < \omega c$, the features will be repeated $\lfloor \frac{\ell}{\omega c} \rfloor$ times. Conversely, if $\ell > \omega c$, average pooling will be used with a pool size and stride of $\lfloor \frac{\omega c}{\ell} \rfloor$.

The role of $\mathcal{L}_{\mathcal{S}}$ in regulating the sparsity of the spike train is essential. It uses L1 norm regularisation to calculate the difference between the input data ($\mathbf{X}_f$) and the surrogate spike train ($\hat{\mathbf{B}}$). This difference serves as an approximation of the number of spikes in the spike train. To limit the number of spikes to align approximately with the *area* of the input signal, Eq. (5) defines the impact of the sparsity term on the training process. This control over sparsity is achieved by taking the derivative of the sparsity term with respect to $\hat{\mathbf{B}}$.

$$\frac{\partial \mathcal{L}_{\mathcal{S}}}{\partial \hat{\mathbf{B}}} = -\lambda\cdot\text{sign}(\mathbf{X}_f-\hat{\mathbf{B}}). \tag{5}$$

where sign$(\cdot)$ is the sign function. This derivative encourages the output values to be close to 0 or 1, leading to a sparse output:

1) When $\hat{\mathbf{B}} > \mathbf{X}_f$, the derivative is negative, pushing output values towards 0.
2) When $\hat{\mathbf{B}} < \mathbf{X}_f$, the derivative is positive, pushing output values towards 1.
3) When $\hat{\mathbf{B}} = \mathbf{X}_f$, the derivative is 0, leaving output values unchanged.

The computational complexity of LAST is $\mathbf{O}\big(\omega c\ell + \ell^2 + \omega\psi c\big)$ per window, where the first term ($\omega c\ell$) corresponds to the feature extraction through dense layers, the second term (ℓ^2) represents the operations in the second feature extraction block, and the final term ($\omega\psi c$) accounts for the spike generation process.

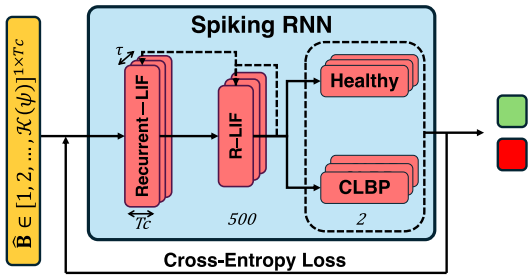


Fig. 3. The SRNN architecture. The spike trains $\hat{\mathbf{B}}$ of each data type (sEMG, Energy, Angle) are classified separately. T represent the total time steps, c are the channels, τ is the time dimension of the Recurrent Leaky Integrate-and-Fire (R-LIF) neurons.

This complexity scales linearly with the input dimensions (window size ω and number of channels c), making it suitable for real-time processing. The quadratic term (ℓ^2) depends on the number of hidden neurons ℓ, a fixed hyperparameter that does not scale with input size. The spike generation parameter ψ controls the maximum number of spikes per timestep, providing a tunable parameter to balance computational cost and encoding precision.

B. Spiking Recurrent Neural Network

The proposed spiking recurrent neural network, depicted in Fig. 3, leverages the temporal dynamics of spike trains for chronic lower back pain classification. The SRNN processes the output of the LAST encoder ($\hat{\mathbf{B}}$) through a network of Recurrent Leaky Integrate-and-Fire (R-LIF) neurons.

The network comprises L layers of R-LIF neurons, where $L = 2$ in our current implementation. The first layer (i.e., hidden layer) contains 500 neurons, and the second layer (i.e., readout layer) contains two neurons corresponding to the healthy and CLBP classes. All layers implement recurrent connections, allowing the network to capture complex temporal dependencies in the input spike trains.

Each R-LIF neuron is implemented using an exponentially decaying membrane potential, represented in discrete time steps. For each layer $l \in \{1, \cdots, L\}$, the membrane potential $\mathbf{U}^l[t]$ and output spikes $\mathbf{S}^l[t]$ at time step t are computed by Eq. (6).

$$\begin{aligned} \mathbf{U}^l[t+1] &= \beta\mathbf{U}^l[t] + \mathbf{I}^l[t+1] \\ &\quad + \mathbf{V}^l(\mathbf{S}^l[t]) - \mathbf{R}^l[t]\mathbf{U}_{\text{thr}}, \\ \mathbf{S}^l[t+1] &= \Theta(\mathbf{U}^l[t+1] - \mathbf{U}_{\text{thr}}), \end{aligned} \tag{6}$$

where $\beta = 0.99$ is the membrane potential decay rate chosen to enhance the LIF neurons' capacity to capture temporal dependencies in input spike trains by promoting longer memory. $\mathbf{I}^l[t]$ is the input current for layer l. The recurrent connections, $\mathbf{V}^l(\mathbf{S}^l[t])$, capture the impact of spikes from all interconnected neurons within layer l. The binary vector $\mathbf{S}^l[t]$ represents the output spikes of all neurons in layer l at time t. We set the membrane threshold $\mathbf{U}_{\text{thr}}$ to 1 in this study. The binary reset-by-subtraction mechanism $\mathbf{R}^l[t]$ is set to 1 when the threshold is reached and 0 otherwise, effectively

subtracting $\mathbf{U}_{\text{thr}}$ (which is 1 in this case) from the membrane potential. Lastly, $\Theta(\cdot)$ refers to the Heaviside step function.

All layers implement all-to-all connectivity. The input currents and recurrent connections for each layer are computed using Eq. (7).

$$\begin{aligned} \mathbf{I}^1[t] &= \mathbf{W}^{\{0,1\}}\hat{\mathbf{B}}[t], \\ \mathbf{I}^l[t] &= \mathbf{W}^{\{l-1,l\}}\mathbf{S}^{l-1}[t], \quad \text{for } l > 1, \\ \mathbf{V}^l(\mathbf{S}^l[t]) &= \mathbf{W}^l_{\text{rec}}\mathbf{S}^l[t], \quad \text{for } l \in 1, \cdots, L, \end{aligned} \tag{7}$$

where $\mathbf{W}^{\{0,1\}} \in \mathbb{R}^{n_1 \times c}$, $\mathbf{W}^{\{l-1,l\}} \in \mathbb{R}^{n_l \times n_{l-1}}$, and $\mathbf{W}^l_{\text{rec}} \in \mathbb{R}^{n_l \times n_l}$ are learnable weight matrices. Here, n_l represents the number of neurons in layer l, with $n_1 = 500$ and $n_2 = 2$ in our architecture, and $\hat{\mathbf{B}}[t]$ represents the input at time step t.

The SRNN processes the input over τ time steps, where $\tau = 5$ is chosen to capture the temporal dynamics of the R-LIF neurons. The network's output is computed using Eq. (8), which accumulates the spikes and membrane potentials of the output layer over all time steps.

$$\begin{aligned} \mathbf{S}_{\text{out}} &= \left[\mathbf{S}^L[1], \mathbf{S}^L[2], \cdots, \mathbf{S}^L[\tau]\right], \\ \mathbf{U}_{\text{out}} &= \left[\mathbf{U}^L[1], \mathbf{U}^L[2], \cdots, \mathbf{U}^L[\tau]\right], \end{aligned} \tag{8}$$

where the final classification is determined by the neuron with the highest cumulative activity across these τ time steps in the output layer.

To train the SRNN, we use the cross-entropy loss function, which is particularly suitable for our binary classification task between healthy and CLBP classes. The cross-entropy loss $\mathcal{L}$ is defined as in Eq. (9).

$$\mathcal{L}_{\text{SRNN}} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{9}$$

where C is the number of classes (in our case, $C = 2$), y_i is the true label (0 or 1) for class i, and $\hat{y}_i$ is the predicted probability for class i. The predicted probabilities are obtained by applying a softmax function to the accumulated output $\mathbf{S}_{\text{out}}$ and $\mathbf{U}_{\text{out}}$. To handle the non-differentiable nature of spikes during backpropagation, we employ a surrogate gradient function. Specifically, we use the gradient of the shifted arc-tan function in Eq. (10).

$$\frac{\partial S}{\partial U} = \frac{1}{\pi} \frac{1}{\left(1 + \left(\pi U \frac{\alpha}{2}\right)^2\right)}. \tag{10}$$

This surrogate gradient allows us to backpropagate the loss through the network and update the weights of the SRNN, enabling effective training despite the discrete nature of spike events.

C. Multi-Stream Ensemble Classification

We have implemented a multi-stream ensemble classification approach using separate LAST-SRNN pipelines to leverage the diverse information captured by various biosignal modalities. This tailored approach provides customised encoding and classification strategies for each type of data to compensate for the information loss during continuous-to-spike conversion and leverages cross-modal interactions. For each data modality $m \in \{\text{sEMG, Angle, Energy}\}$, we train an independent LAST-SRNN pipeline formulated in Eq. (11).

$$\mathbf{Y}_m = \mathcal{P}_m(\hat{\mathbf{B}}_m; \boldsymbol{\theta}_m), \tag{11}$$

where $\hat{\mathbf{B}}_m$ is the output of the LAST encoder for modality m, as defined in Eq. (3). Here, $\mathcal{P}_m(\cdot)$ denotes the LAST-SRNN pipeline for that modality, $\boldsymbol{\theta}_m$ represents the learnable parameters of the pipeline, and $\mathbf{Y}_m \in \mathbb{R}^2$ is the output prediction (i.e., probabilities for Healthy and CLBP classes) for modality m. Each LAST-SRNN pipeline is trained independently using the cross-entropy loss function, as described in Section III-B.

The outputs from these individual pipelines serve as features for the meta-classifier, which is implemented using a Random Forest classifier [20]. This classifier is comprised of multiple decision trees, each trained on a bootstrap sample of the training data using the Gini impurity function to measure the quality of a split. The final prediction is determined by majority voting among the trees using Eq. (12).

$$\hat{y} = \underset{C}{\text{mode}} \left\{ \text{RF}_i \left(\mathbf{Y}_{\text{sEMG}}, \mathbf{Y}_{\text{Angle}}, \mathbf{Y}_{\text{Energy}} \right) \right\}_{i=1}^{N_t}, \tag{12}$$

where $\text{RF}_i(\cdot)$ represents the prediction of the i-th decision tree in the Random Forest. The final prediction $\hat{y}$ is determined by majority voting among the N_t trees.

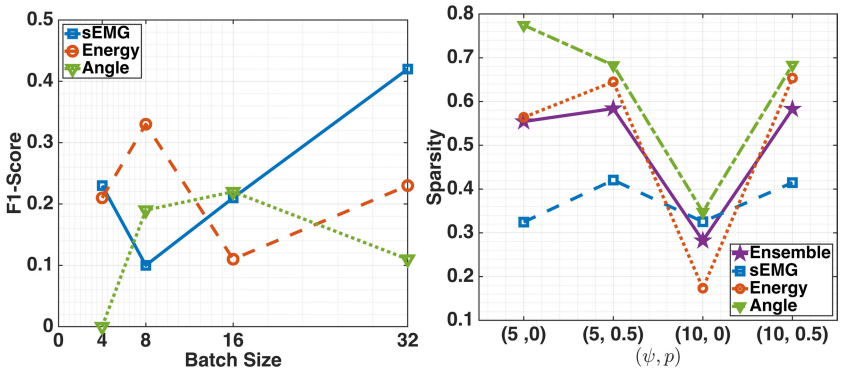
IV. Experiments and Results

A. Dataset and Preparation

The EmoPain dataset [5] is a multimodal collection of data from 46 participants, encompassing healthy individuals and those with CLBP. Participants performed various exercises (one-leg-stand, reach-forward, stand-to-sit, sit-to-stand, and bend-down) at normal and difficult levels, with transitions included to simulate real-life movements.

Data were collected using 18 wearable IMU sensors and four sEMG sensors. The IMU sensors recorded full-body 3D motion, from which 13 joint angle data (Joint Angle) and 13 angular energy data (Joint Energy) were derived. The sEMG sensors captured muscle activity data from the right and left upper and lower back muscle groups. The dataset includes self-reported pain intensity ratings (0-(10) from individuals with chronic lower back pain and labels for pain-related behaviours (protective vs. non-protective) provided by physiotherapists. For the CLBP classification task, we assigned a positive label to individuals reporting a pain intensity greater than 5. For the pain-related behaviour classification task, we determined the ground truth by taking the majority vote across raters. Frames with tied votes were assigned a positive label.

To prepare the data for analysis, we addressed the variation in recording lengths by trimming recordings that exceeded 18,000 frames by 6,000 frames from both ends. This method allowed us to balance standardisation with data preservation, reducing information loss and the impact of stress and fatigue [12]. We used normalisation techniques for all data modalities (sEMG, Angles, and Energies) to ensure consistent scaling and facilitate model training. Specifically, we normalised values to a 0-1 range by subtracting the minimum



(a) (b)

Fig. 4. (a) The influence of batch size on the F1 score of the proposed LAST-SRNN for each modality. (b) The spike density for the proposed architecture as a function of (ψ, p). While the difference in spike density between $(\psi, p) = (5, 0.5)$ and $(\psi, p) = (10, 0.5)$ is less than 1% (0.549 vs 0.540), the former achieves better performance across other classification metrics.

value and dividing by the maximum value within each sample. Additionally, we zero-padded the data to match the longest recording length (17,995 time steps). Lastly, we employed a sliding window approach with a 3,000 time-step window (50 seconds at 60 Hz), allowing the LAST to capture longer-term temporal dependencies and enhance robustness to noise.

B. Implementation Details

The experiments were conducted on a high-performance computing system equipped with an AMD EPYC 7302 16-core processor, 504GB of RAM, and an NVIDIA RTX A6000 GPU. The deep learning framework used was PyTorch version 2.3.1 [21]. The AdamW optimiser [22] was employed for both the LAST encoder and the SRNN classifier, with fixed learning rates of 5.0×10^{-3} and 7.5×10^{-4}, respectively, due to its effectiveness in handling sparse gradients, a common characteristic in SNN training.

The number of neurons in the hidden layers was chosen based on the size of the sliding window. Therefore, both $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$ were designed with 3000 neurons each to effectively process the temporal information within the data. The SRNN classifier's hidden layer was composed of 500 recurrent LIF neurons. This decision was deliberate to find a balance between computational efficiency, model performance, and generalisability. Additionally, to ensure that the model learns from all classes in the context of this imbalanced dataset with a small sample size and to mitigate any biasing effects, we applied minority over-sampling with weighted sample replacement during batch creation.

An empirical study identified optimal hyperparameters for our LAST-SRNN architecture. The best configuration (see Fig. 4) included a batch size of 32 for sEMG, 8 for Energy, and 16 for Angle, along with a dropout rate of 0.5 and $\psi = 5$.

The training process consisted of 30 epochs for the LAST encoder and 25 epochs for the SRNN classifier, with early stopping implemented to prevent overfitting. This training regime was carefully chosen to balance model convergence and computational efficiency. All the hyperparameters and settings determined through this empirical search were subsequently used consistently across all experiments in this study. This approach ensures the reproducibility of our results and allows for a fair comparison across different experimental conditions.[1]

C. Evaluation Metrics

We used both standard classification metrics and a sparsity measure tailored to SNNs to evaluate the effectiveness of the LAST-SRNN framework for CLBP classification on the EmoPain dataset. All metrics were assessed using leave-one-subject-out cross-validation to ensure a rigorous analysis, especially given the relatively small size of the EmoPain dataset.

The standard classification metrics employed in this study encompassed accuracy, Macro F1 score, Area Under the Curve (AUC), and Matthews Correlation Coefficient (MCC). These metrics were suggested in the EmoPain challenge [23] for comprehensive insights into the model's performance across various aspects and are formulated in Eq. (13) and Eq.(14).

$$\begin{aligned}
\text{Accuracy} &= \frac{TP+TN}{TP+TN+FP+FN}, \\
\text{Precision} &= \frac{TP}{TP+FP}, \\
\text{Recall} &= \frac{TP}{TP+FN}, \\
\text{F1} &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}},
\end{aligned} \tag{13}$$

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}, \tag{14}$$

where TP, FP, TN, and FN represent the number of true positive, false positive, true negative, and false negative predictions by the classifier, respectively.

[1]The implementation is accessible for reproducibility purposes on https://github.com/freek1/emopain-stl

TABLE I
PERFORMANCE OF ENCODERS ON THE EMOPAIN DATABASE FOR CLBP CLASSIFICATION

Encoder	Classifier	Accuracy ↑	AUC ↑	F1 ↑	MCC ↑	Spike density ↓			
						Ensemble	sEMG	Energy	Angle
Baseline Methods									
Rate-based [1]	SRNN	73.91%	0.50	0.00	0.00	0.17	0.07	0.58	0.70
Latency-based [3]	SRNN	71.74%	0.51	0.13	0.04	0.20	0.20	**0.20**	**0.20**
Proposed Method									
LAST–Stacked	SRNN	**80.43%**	**0.68**	**0.53**	**0.44**	0.55	0.36	0.70	0.79
LAST–Vanilla	SRNN	76.09%	0.60	0.35	0.27	**0.02**	**0.01**	0.44	0.53

In addition to these standard metrics, we introduced a spike density measure to assess the efficiency of our encoded spike trains. This measure provides insight into the sparsity of our neural representations across our multimodal approach. The overall spike density is computed as the harmonic mean of the individual densities for each data modality, as shown in Eq. (15).

$$\text{SpikeDensity} = \frac{3}{\sum_m \frac{1}{D_m}},$$
$$D_m = \frac{\sum_{i=1}^{T}\sum_{j=1}^{\psi}\sum_{k=1}^{c_m}\mathbf{B}_{m,i,j,k}}{T\psi c_m}, \quad (15)$$

where, $\mathbf{B}_m \in \{0,1\}^{T\times\psi\times c_m}$ represents the spike train for modality m, summed over T time steps, c_m channels, and ψ spikes per time step. The denominator represents the maximum possible number of spikes for that modality.

D. Experimental Results

Our analysis focuses on three main aspects: (1) the performance of different spike train encoding strategies with the SRNN classifier for both CLBP and protective behaviour classification, (2) an ablation study to understand the impact of architectural choices and hyperparameters, and (3) a comparison with state-of-the-art deep learning models. All experiments were conducted using leave-one-subject-out cross-validation on the EmoPain dataset.

1) CLBP Classification: We assessed four spike train encoding strategies using the SRNN classifier for chronic lower back pain classification. LAST–Stacked represents the proposed encoder explained in Section III-A. LAST–Vanilla represents a simplified version of LAST–Stacked, wherein the input is directly connected to the Repeat layer, excluding the two feature extraction blocks. Rate-based encoding [1] and Latency-based encoding [3] serve as baseline methods. Table I presents the experimental results, and Fig. 5 shows the corresponding confusion matrices.

The results indicate that LAST–Stacked achieved the best performance across all metrics. This highlights the effectiveness of our proposed method's architecture, particularly its feature extraction blocks, which enhance classification accuracy and discriminative power. While the F1 score does not match the levels achieved by highly optimised deep learning methods in our previous work [14], [24], this gap should be considered within the broader context of neuromorphic computing. Our primary objective extends beyond competing directly with deep learning in raw performance metrics.

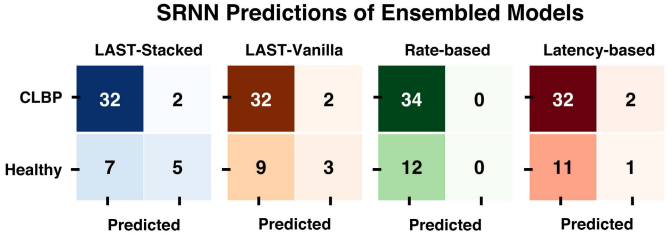


Fig. 5. Confusion Matrices of the Encoders with the SRNN classifier for CLBP Classification.

Instead, we focus on demonstrating the potential of Spiking Neural Networks (SNNs) for energy-efficient biosignal processing. This highlights the unique advantages of neuromorphic computing beyond mere performance metrics.

The distinction between LAST–Stacked and LAST–Vanilla highlights the importance of incorporating feature extraction blocks. These elements are important for developing more discriminative representations, which significantly enhance the performance of LAST–Stacked. Additionally, the performance of LAST–Vanilla, even with minimal spike density, suggests its potential utility in resource-constrained environments.

Baseline methods demonstrated notable limitations. Rate-based coding demonstrated reasonable accuracy but struggled to distinguish between classes effectively, as indicated by the F1 score and MCC. Latency-based encoding showed lower overall accuracy, highlighting the importance of combining spatial and temporal encoding strategies for optimal results.

2) Protective Behaviour Classification: To illustrate the adaptability of our framework, we evaluated its performance on protective behaviour classification using physiotherapist-provided labels in the EmoPain dataset. The results presented in Table II and Fig. 6 reinforce our previous findings, showcasing LAST–Stacked's robust performance in this additional classification task.

The F1 score achieved by LAST–Stacked in protective behaviour classification indicates its effectiveness in identifying instances of protective behaviour when they occur. This characteristic is valuable in clinical settings, where accurately detecting protective behaviours is critical, even if it comes at the expense of some false positives. Such a trade-off can be advantageous in scenarios where missing instances of protective behaviour could have more serious consequences than false alarms.

Analysing the performance patterns across encoders provides valuable insights into the nature of protective behaviour classification. LAST–Stacked's higher spike density compared

TABLE II
PERFORMANCE OF ENCODERS ON THE EmoPain DATABASE FOR PROTECTIVE BEHAVIOUR CLASSIFICATION

Encoder	Classifier	Accuracy ↑	AUC ↑	F1 ↑	MCC ↑	Spike density ↓			
						Ensemble	sEMG	Energy	Angle
Baseline Methods									
Rate-based [1]	SRNN	41%	0.41	0.37	-0.17	0.17	0.06	0.58	0.70
Latency-based [3]	SRNN	50%	0.52	0.51	0.01	0.20	0.20	**0.20**	**0.20**
Proposed Method									
LAST–Stacked	SRNN	**61%**	**0.55**	**0.65**	**0.21**	0.65	0.66	0.59	0.69
LAST–Vanilla	SRNN	54%	0.52	0.57	0.08	**0.02**	**0.01**	0.44	0.53

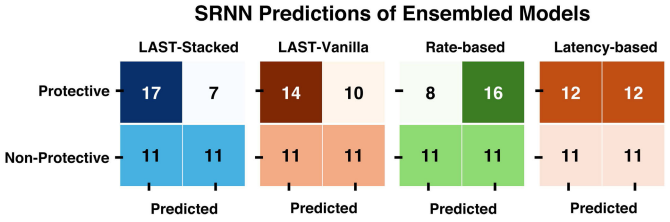


Fig. 6. Confusion matrix for protective behaviour classification.

TABLE III
COMPARISON OF THE PROPOSED SPIKING ARCHITECTURE WITH DEEP LEARNING MODELS ON THE EmoPain DATASET

Methods	CLBP			Pain-related Behaviour		
	AUC	F1	MCC	AUC	F1	MCC
Deep Learning Methods						
MiMT [25]	0.65	0.72	0.33	0.66	0.41	0.33
LSTM + GCN [26]	0.69	0.73	0.36	0.44	0.66	0.35
GRU-RNN [14]	0.85	0.76	0.41	0.78	0.83	0.37
L-SFAN [24]	0.84	0.77	0.51	-	-	-
Proposed Spiking Method						
LAST-SRNN	0.68	0.53	0.48	0.55	0.65	0.21

to CLBP classification indicates that identifying protective behaviours necessitates richer temporal information. LAST–Vanilla's performance with minimal spike density further supports this observation, underscoring the importance of detailed spike representations for this specific task.

The baseline methods showed considerable limitations in this task. Rate-based coding performed poorly, achieving only modest accuracy with a negative MCC, indicating performance worse than random chance. Although latency coding demonstrated some improvement, its performance remains below that of the LAST method, highlighting the critical role of adaptive threshold learning in capturing complex movement patterns.

3) Ablation Study: The performance differences between LAST–Stacked and LAST–Vanilla across both classification tasks warranted further investigation of our architectural choices. We focused particularly on the impact of batch size and the relationship between spike generation parameters (ψ, p).

Figure 4(a) reveals distinct optimal batch sizes for different modalities. These variations likely stem from each modality's unique temporal characteristics and information density. Smaller batch sizes for certain modalities (e.g., Energy) suggest these data types benefit from more frequent parameter updates, possibly due to their more variable temporal patterns.

The relationship between spike generation parameters (see Fig. 4(b)) provides crucial insights into the efficiency-performance trade-off in our architecture. While configurations of $(\psi, p) = (5, 0.5)$ and $(10, 0.5)$ produced similar spike densities, the former achieved better performance across classification metrics. This finding suggests that increasing the number of potential spike positions does not necessarily improve classification performance. Instead, the balance between sufficient temporal resolution and efficient spike generation appears more crucial for effective biosignal encoding.

These ablation results support our architectural choices in LAST–Stacked. The feature extraction blocks, despite increasing computational complexity, are vital for capturing nuanced temporal dynamics in both CLBP and protective behaviour classifications. The ability to maintain good performance with carefully chosen parameters demonstrates the framework's potential for optimization in resource-constrained applications.

4) Comparison With Deep Learning Models: Our comparison with state-of-the-art deep learning approaches provides valuable insights into the current capabilities and limitations of spike-based computation for biosignal analysis. Table III summarizes this comparison for both CLBP and protective behaviour classification tasks.

For CLBP classification, our spike-based approach demonstrates competitive performance in balanced classification, particularly evident in the MCC scores. While deep learning approaches achieve higher overall accuracy and AUC, these gains come at the cost of substantially greater computational resources. This narrow performance gap suggests that our spike-based approach effectively captures the essential class discrimination patterns.

The protective behaviour classification results reveal interesting trade-offs. Our method shows particular strength in positive class detection, which is especially relevant for clinical applications. However, the lower AUC and MCC scores compared to deep learning methods indicate that maintaining consistent performance across all classification scenarios remains challenging within the constraints of spike-based computation.

A key advantage of our approach lies in its potential for energy efficiency, particularly when implemented on neuromorphic hardware. Traditional deep learning approaches, while achieving higher accuracy in some metrics, typically require substantially greater computational resources. Our work explores this fundamental trade-off between computational efficiency and classification performance, establishing a foundation for low-power biosignal analysis systems.

These results demonstrate both the promise and current limitations of spike-based computation for complex biosignal analysis. We achieve competitive performance in key metrics, validating the potential of our approach for efficient neuromorphic implementation. Sparse representations translate directly to reduced energy consumption on neuromorphic hardware, a key advantage for resource-constrained applications. The varying performance across tasks provides valuable insights for future development, suggesting areas where architectural improvements might bridge the remaining performance gap with deep learning while preserving the inherent efficiency advantages of neuromorphic computation.

V. Conclusion and Future Work

Neuromorphic computing has great potential for ultra-low-power signal processing. However, converting continuous signals into spike trains is still challenging. In this paper, we introduced Learning Adaptive Spike Thresholds (LAST), a method designed to address this challenge. The LAST encoder learns adaptive thresholds for generating spikes, preserving spatio-temporal dynamics while keeping representations sparse across various signal dimensions. This sparsity helps make neuromorphic systems more efficient, reducing power use as seen in recent advances [27]. To demonstrate how well LAST works, we tested it on biosignal analysis with the EmoPain dataset, which includes challenges like small sample sizes, class imbalance, and complex temporal patterns.

Our experiments demonstrated that LAST can handle complex classification tasks within the constraints of spike-based computation. We achieved competitive results in classifying CLBP and promising outcomes in detecting protective behaviours. These results were reached using sparse spike representations (i.e., 54.9% and 65.0% spike density) for CLBP and protective behaviour classification, respectively. This suggests that LAST fits well with energy-constrained neuromorphic systems.

LAST's success in managing multimodal biosignals suggests its potential for applications beyond healthcare. As a signal-agnostic encoder, it handles various signal types, from one-dimensional time series to high-dimensional tensors. This versatility makes LAST valuable for fields such as audio processing, image analysis, and environmental monitoring [28]. Additionally, LAST's capability to manage small datasets and address class imbalance renders it particularly practical when large-scale data collection is challenging or costly.

Several research directions are worth exploring next. First, running LAST on dedicated neuromorphic hardware platforms, like Intel's Loihi [29], could help fully realise its potential for low-power computation. Recent progress in these platforms, reaching femtojoule-level power consumption per synaptic operation [30], [31]. Also, looking at more advanced architectures, such as spiking transformers [32], [33], along with new learning methods that combine supervised global learning with local mechanisms [34], could improve LAST's ability to capture complex temporal dependencies. These developments could extend our framework's capabilities beyond classification to include predictive functions.

In healthcare, future work could investigate pain progression patterns through longitudinal studies, integrate additional physiological markers (e.g., heart rate variability, skin conductance) for developing early warning systems, and explore closed-loop control systems for automated interventions. These extensions would need careful consideration of clinical requirements and additional data collection protocols, but they have the potential to greatly improve pain management strategies.

Overall, the LAST encoder shows promise in connecting conventional signal processing with energy-efficient neuromorphic computing. By providing a robust and adaptable solution for continuous signal encoding into sparse spike trains, LAST establishes a foundation for developing energy-efficient intelligent systems across diverse application domains.

References

[1] E. D. Adrian and Y. Zotterman, "The impulses produced by sensory nerve endings," *J. Physiol.*, vol. 61, no. 4, pp. 465–483, Aug. 1926.

[2] S. Oh et al., "Neuron circuits for low-power spiking neural networks using time-to-first-spike encoding," *IEEE Access*, vol. 10, pp. 24444–24455, 2022.

[3] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, no. 5866, pp. 1108–1111, 2008.

[4] S. S. Radhakrishnan, A. Sebastian, A. Oberoi, S. Das, and S. Das, "A biomimetic neural encoder for spiking neural network," *Nature Commun.*, vol. 12, no. 1, p. 2143, Apr. 2021.

[5] M. S. H. Aung et al., "The automatic detection of chronic pain-related expression: Requirements, challenges and the multimodal EmoPain dataset," *IEEE Trans. Affect. Comput.*, vol. 7, no. 4, pp. 435–451, Oct. 2016.

[6] M. Shahsavari, D. Thomas, A. Brown, and W. Luk, "Neuromorphic design using reward-based STDP learning on event-based reconfigurable cluster architecture," in *Proc. Int. Conf. Neuromorphic Syst.*, Jul. 2021, pp. 1–8.

[7] J. M. Fabian, D. C. O'Carrol, and S. D. Wiederman, "Sparse spike trains and the limitation of rate codes underlying rapid behaviours," *Biol. Lett.*, vol. 19, no. 5, May 2023, Art. no. 20230099.

[8] W. Ma, L. Chen, C. Du, and W. D. Lu, "Temporal information encoding in dynamic memristive devices," *Appl. Phys. Lett.*, vol. 107, no. 19, Nov. 2015, Art. no. 193101.

[9] I. Uysal, H. Sathyendra, and J. G. Harris, "Spike-based feature extraction for noise robust speech recognition using phase synchrony coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 1529–1532.

[10] M. Shahsavari, D. Thomas, M. van Gerven, A. Brown, and W. Luk, "Advancements in spiking neural network communication and synchronization techniques for event-driven neuromorphic systems," *Array*, vol. 20, Dec. 2023, Art. no. 100323.

[11] J. K. Eshraghian et al., "Training spiking neural networks using lessons from deep learning," *Proc. IEEE*, vol. 111, no. 9, pp. 1016–1054, Sep. 2023.

[12] R. F. Rojas, N. Brown, G. Waddington, and R. Goecke, "A systematic review of neurophysiological sensing for the assessment of acute pain," *Npj Digit. Med.*, vol. 6, no. 1, p. 76, 2023.

[13] N. Makaram, P. A. Karthick, and R. Swaminathan, "Analysis of dynamics of EMG signal variations in fatiguing contractions of muscles using transition network approach," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–8, 2021.

[14] M. M. Dehshibi, T. Olugbade, F. Diaz-de-Maria, N. Bianchi-Berthouze, and A. Tajadura-Jiménez, "Pain level and pain-related behaviour classification using GRU-based sparsely-connected RNNs," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 3, pp. 677–688, May 2023.

[15] J. Plank, C. Zheng, C. Schuman, and C. Dean, "Spiking neuromorphic networks for binary tasks," in *Proc. Int. Conf. Neuromorphic Syst.*, Jul. 2021, pp. 1–9.

[16] E. Donati, M. Payvand, N. Risi, R. Krause, and G. Indiveri, "Discrimination of EMG signals using a neuromorphic implementation of a spiking neural network," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 5, pp. 795–803, Oct. 2019.

[17] M. Elhamdaoui, F. O. Rziga, K. Mbarek, and K. Besbes, "Spike-time-dependent plasticity rule in memristor models for circuit design," *J. Comput. Electron.*, vol. 21, no. 4, pp. 1038–1047, Aug. 2022.

[18] P. Gong, P. Wang, Y. Zhou, and D. Zhang, "A spiking neural network with adaptive graph convolution and LSTM for EEG-based brain-computer interfaces," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 1440–1450, 2023.

[19] A. El Ferdaoussi, J. Rouat, and E. Plourde, "Efficiency metrics for auditory neuromorphic spike encoding techniques using information theory," *Neuromorphic Comput. Eng.*, vol. 3, no. 2, Jun. 2023, Art. no. 024007.

[20] T. K. Ho, "Random decision forests," in *Proc. IEEE 3rd Int. Conf. Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282.

[21] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. NIPS Autodiff Workshop*, no. 4, 2017, pp. 1–4.

[22] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–18.

[23] J. O. Egede et al., "EMOPAIN challenge 2020: Multimodal pain evaluation from facial and bodily expressions," in *Proc. 15th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, Nov. 2020, pp. 849–856.

[24] J. Ortigoso-Narro, F. Diaz-De-Maria, M. M. Dehshibi, and A. Tajadura-Jiménez, "L-SFAN: Lightweight spatially-focused attention network for pain behavior detection," *IEEE Sensors J.*, early access, Jan. 17, 2025, doi: 10.1109/JSEN.2025.3540415.

[25] T. Olugbade, N. Gold, A. C. D. C. Williams, and N. Bianchi-Berthouze, "A movement in multiple time neural network for automatic detection of pain behaviour," in *Proc. Companion Publication Int. Conf. Multimodal Interact.*, Oct. 2020, pp. 442–445.

[26] C. Wang, Y. Gao, A. Mathur, A. C. D. C. Williams, N. D. Lane, and N. Bianchi-Berthouze, "Leveraging activity recognition to enable protective behavior detection in continuous data," *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 5, no. 2, pp. 1–27, Jun. 2021.

[27] S. Ward-Foxton. (Jan. 2025). *Innatera Unveils Neuromorphic AI Chip to Accelerate Spiking Networks*. [Online]. Available: https://www.eetimes.com/innatera-unveils-neuromorphic-ai-chip-to-accelerate-spiking-networks/

[28] M. M. Dehshibi, M. S. Fahimi, and M. Mashhadi, "Vision-based site selection for emergency landing of UAVs," in *Recent Advances in Information and Communication Technology*. Berlin, Germany: Springer, 2015, pp. 133–142.

[29] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[30] H. Bos and D. Muir, "Sub-mW neuromorphic SNN audio processing applications with rockpool and Xylo," in *Embedded Artificial Intelligence*. Aalborg, Denmark: River, 2023, pp. 69–78, doi: 10.1201/9781003394440.

[31] M. Yao et al., "Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip," *Nature Commun.*, vol. 15, no. 1, p. 4464, May 2024.

[32] Z. Zhou et al., "Spikformer: When spiking neural network meets transformer," in *Proc. 11th Int. Conf. Learn. Representations*, 2023, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=frE4fUwz_h

[33] M. Yao et al., "Spike-driven transformer," in *Proc. 37th Conf. Neural Inf. Process. Syst.*, 2023, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=9FmolyOHi5

[34] Y. Wu et al., "Brain-inspired global-local learning incorporated with neuromorphic computing," *Nature Commun.*, vol. 13, no. 1, p. 65, Jan. 2022.