Spiking neural network with time-varying weights for rail squat detection

Phusakulkajorn, Wassamon; Hendriks, Jurjen; Li, Zili; Núñez, Alfredo

DOI
10.1016/j.asoc.2025.113689

Publication date
2025

Document Version
Final published version

Published in
Applied Soft Computing

Applied Soft Computing Journal 184 (2025) 113689

Contents lists available at ScienceDirect

Applied Soft Computing

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

Spiking neural network with time-varying weights for rail squat detection

Wassamon Phusakulkajorn, Jurjen Hendriks, Zili Li, Alfredo Núñez *

Section of Railway Engineering, Delft University of Technology, Delft, 2628CN, The Netherlands

ARTICLE INFO

Keywords:
Spiking neural network
Rail monitoring
Axle-box acceleration
Rail squats
Explainable artificial intelligence
Intelligent railway infrastructure

ABSTRACT

Axle box acceleration (ABA) measurements can be used for continuously monitoring rail infrastructure and detecting rail surface defects such as squats. However, accurately detecting squats is challenging due to their short-duration responses and low occurrence in ABA signals, particularly for light squats that exhibit subtle ABA responses. To address this challenge, we propose using a spiking neural network (SNN) with time-varying weights to enhance the detection performance of rail squats based on ABA measurements. Our approach employs a simple SNN architecture without hidden layers, trained using a method that combines genetic algorithms, k-fold cross-validation, and multi-start gradient-based approach to optimise hyperparameters and weights. The proposed methodology demonstrates competitive accuracy compared to other state-of-the-art SNN-based methods on UCI benchmarks for both binary and multi-class nonlinear problems. Part of its advantages include higher efficiency with a simpler architecture and training approach that reduces computational times while achieving effective spatiotemporal pattern detection. As shown by real-field measurements from Dutch and Swedish railways in anomaly detection, it effectively captures subtle changes in light squat defect responses in ABA signals and achieves a detection performance of 100% for severe squat defects and over 93% for light squat defects. Furthermore, we show that the spike responses, postsynaptic potentials, and membrane potentials can be used as a new way to explain and analyse the ABA signals. The proposed method using time-varying weights highlights a correspondence with the physical problem and offers an ability to capture sudden and subtle changes in the responses, which is crucial, particularly for detecting defects in their early stages.

1. Introduction

Squats are short-wave surface defects and one type of rolling contact fatigue on railway rails [1]. Fig. 1 illustrates examples of surface defects from a railway line in Sweden and the Netherlands. The severity of squats can be classified into light, moderate, and severe [2]. Early detection and assessment of all types of squats are crucial for planning maintenance operations [3] (see Fig. 2).

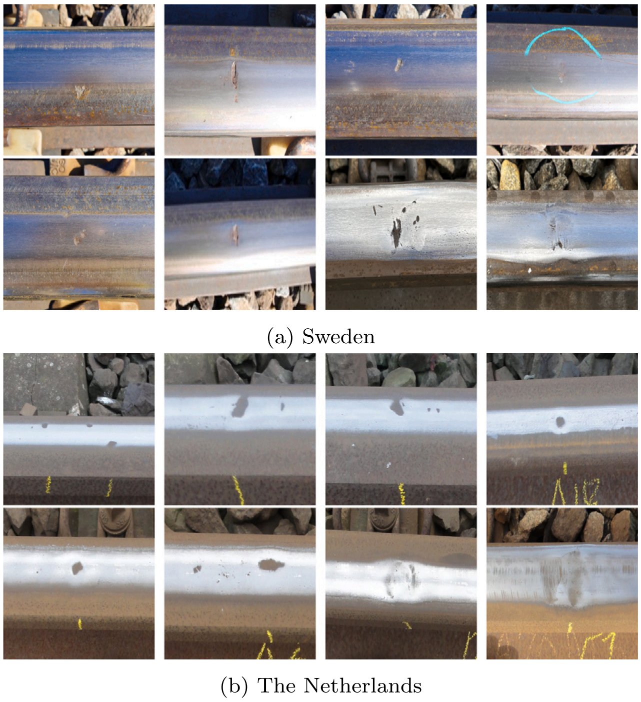
Rail maintenance typically considers grinding and replacement. In the case of light squats with minor cracks, grinding has the potential to remove them completely from the rail surface [4]. When squats become severe, a degradation of the track structure is experienced, and rail breaks can eventually occur. For severe squats, rail replacement is more suitable as multiple grinding passages might not entirely remove them, leading to the re-appearing of the defect due to residual damages. As grinding is more cost- and performance-effective than rail replacement, the early and accurate detection and management of squats are needed.

Various measurement technologies have been used for the detection of squats, for example, ultrasonic [5], eddy current [6], guided waves [7], image processing [8], axle box acceleration (ABA) [2], among others. In the literature, light squats can be detected using ABA measurements with performance between 78% to 85% [2,9]. Using vertical and longitudinal ABA signals in conjunction with noise-reduction techniques in [9], 100% of severe squats were detected, while 85% of small rail surface defects were found. For the automatic detection method proposed in [2], the detection performance was 100% for severe squats but 78% for light squats. As approximately 15%–22% of light squats are not detected using traditional methods, we consider using neural networks to increase the detection performance in this work.

In the literature, deep learning neural networks have been utilised to detect squats [8,10,11]. In [8], the performance of small, medium, and large deep convolutional neural network (DCNNs) for detecting squats using video images were compared. The results demonstrated that a large DCNN model was needed to detect light squats with 64.4% accuracy. In [10], a DCNN was proposed to detect squats using video images correlated to the corresponding ABA signals. This approach detected 96.9% of visible squats. In [11], the proposed unsupervised method based on convolutional variational auto-encoder achieved a

[*] Corresponding author.
E-mail address: A.A.NunezVicencio@tudelft.nl (A. Núñez).

https://doi.org/10.1016/j.asoc.2025.113689
Received 20 March 2023; Received in revised form 16 July 2025; Accepted 24 July 2025
Available online 8 August 2025



(a) Sweden

(b) The Netherlands

Fig. 1. Rail surface defects from different countries.

hit rate of 100% for light squats using ABA measurements under a simulated vehicle–track environment. However, in practice, the detection performance of light squats is not 100% due to their complex spatiotemporal patterns, which are challenging for current detection methods to capture.

Spiking neural networks (SNNs), considered the third generation of neural networks [12], offer the ability to handle temporal and spatiotemporal patterns through their unique mechanism of processing data as discrete events or spikes. The potential applications of SNNs span various fields such as biomedical science and mechanical engineering [13–17]. SNNs are emerging as powerful tools for energy-efficient, real-time processing. In object detection, SNNs achieve high accuracy with ultralow latency, making them well-suited for neuromorphic hardware deployment [18–20]. They have also shown promise in smart environments, for example, enabling face recognition in IoT-based office automation through multi-dimensional attention mechanisms [21]. Additionally, SNNs have been successfully integrated with self-attention to improve sequence modelling in recommendation systems, effectively capturing temporal dependencies [22]. Furthermore, fMRI-based SNNs have demonstrated robust anti-disturbance capabilities in speech recognition, underscoring their resilience to noise [23]. These developments highlight the applicability of SNNs in addressing real-world challenges where power efficiency and temporal precision are critical.

To explore their capability in railway application further, this paper proposes using SNNs for detecting rail squats. The selection is motivated by their ability to process sparse, event-based data with spatiotemporal patterns that correspond to encoded features derived from abrupt changes in the time–frequency domain of ABA signals, such as those caused by squats. While CNNs are widely used for detection tasks, they rely on continuous, dense input processing with artificial neurons and typically require substantial computational resources on conventional hardware (e.g., GPUs, TPUs). In contrast, SNNs use biologically inspired spiking neurons with temporal dynamics, enabling event-driven, sparse activity that leads to low energy consumption and efficient temporal processing without additional modules like LSTMs or GRUs. SNNs also handle sparse sensor input well, require fewer labelled examples, and are optimised for neuromorphic hardware (e.g., Loihi, TrueNorth). Although the SNN ecosystem is still maturing, with fewer libraries and benchmarks available compared to CNNs, these advantages make SNNs particularly appropriate for detecting rail squats in practical deployment scenarios.

SNNs have been designed with many network parameters [24–29]. On the one hand, having many hidden layers and nodes allows biological plausibility and higher accuracy. On the other hand, having many parameters poses the issue that SNNs are more difficult to train and their evaluation can be computationally costly. Railway infrastructures are inherently large-scale, with variations that depend on time and track location and stochastic systems. Additionally, rail squat detection technologies rely on high-frequency monitoring data. Thus, a less computationally expensive model that meets online detection requirements and facilitates decision-making is preferred.

Employing SNNs without hidden layers or hidden nodes has been considered in the literature to reduce computational efforts. In [30,31], and [32], efficient algorithms for networks with no hidden layer were presented. In both works, the algorithms outperformed the existing SNNs with more complex network architecture in terms of both accuracy and computational cost when testing with benchmarks from the UCI machine learning repository for both binary and multiple classes. The contribution from [30,31], and [32] has opened up an opportunity to find a well-balanced trade-off between computational effort and accuracy for SNNs. Therefore, instead of using large network architecture, this work uses simple network architecture with no hidden layers to solve a complex spatiotemporal problem presented in early squat detection. The main contributions of this paper are:

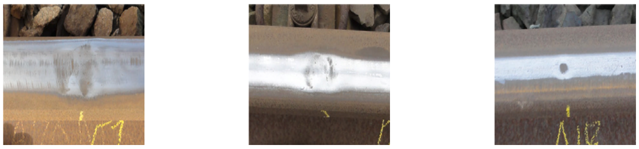
1. An SNN-based methodology with time-varying weights is proposed to detect rail surface defects, e.g., squats, of varying severity levels, using ABA measurements. This method aims to improve the detection performance of light squats, which present challenges due to their subtle, short-duration responses and typically a low percentage of appearance in ABA signals compared to healthy rails. Instead of using large network architecture, this work uses simple network architecture with no hidden layers to solve a complex spatiotemporal problem presented in early squat detection.
2. A global optimisation approach is considered for the training process, incorporating a genetic algorithm to search for hyper-parameters based on cross-validation and gradient-based approach to adjust time-varying weights with multiple starts.
3. A utilisation of spike responses, postsynaptic potentials, and membrane potentials is presented to provide an explainable way for squat detection that relies on ABA signals. Visual explanations from these internal spike behaviours are presented to identify a correspondence with the physical problem.

The rest of this paper is outlined as follows. Section 2 provides background knowledge of rail squats and SNNs. Section 3 introduces the problem of squat detection and the proposed framework, while the feature engineering is elaborated in Section 4. Section 5 presents the proposed SNN-based methodology. Section 6 presents a sensitivity analysis of hyper-parameters of our methodology and the comparative study with the state-of-the-art SNNs using UCI benchmarks. In Section 7, the capability of our SNN-based methodology to solve a complex spatiotemporal problem presented in squat detection based on ABA measurements is compared with other methods. The explainability of the methodology for squat detection is also elaborated in this section. This paper is concluded in Section 8.

2. Background knowledge

2.1. Rail squats

A rail squat is a type of rail surface defect that arises from rolling contact fatigue (RCF). It is characterised by local plastic deformation



(a) Severe squat. (b) Moderate squat. (c) Light squat.

Fig. 2. Illustration of rail squats with different severity levels.

on the rail top surface. The direct cause is excessive dynamic wheel–rail contact force. Contributing factors to their occurrence include increased axle loads and traffic density, different maintenance policies, deteriorating track quality, new materials for wheels and rails, and the use of stiffer concrete sleepers [33].

Severe, moderate, and light squats are classifications of rail squats distinguished by their wavelengths and the frequency characteristics of wheel–rail dynamic interactions, which can be measured by ABA systems. Fig. 2 illustrates squat classifications in which Figs. 2(a), 2(b), and 2(c) show an example of a severe, moderate, and light squat, respectively. Severe squats are the largest and deepest defects among the three categories, with lengths typically ranging more than 50 millimetres [34]. Severe squats generate pronounced, high-amplitude vibrations with distinct frequency characteristics between 200–400 Hz [2]. The larger the severe squats, the greater the risk of derailment. Moderate squats are smaller than severe squats, with lengths ranging between 30–50 millimetres [34]. They cause rail vibrations with the same frequency band as the severe squats, though less pronounced amplitudes. Light squats are the smallest and shallowest defects among the three categories, often representing the initial or early stages of rail squats. Their lengths range between 8–30 millimetres [34]. The characteristic frequency bands for light squats are between 200–400 Hz and 1000–2000 Hz [2]. If left untreated, light squats can develop into more severe defects. However, some small defects can be worn away through natural wear.

Rail squats are critical to detect and manage because they can lead to increased maintenance costs, reduced rail life, and potential safety hazards. However, detecting squats, particularly for light squats, with high accuracy is challenging. Some possible underlying reasons are the following. First, light squats are anomalies with a low percentage of appearance in monitoring data. Some light squats are difficult to find via visual inspections (or even impossible when these are still not visible to human eyes), making the labelling process difficult. Second, the response of light squats in ABA signals appears suddenly and has a very short duration. For example, at a light squat of 8 mm in wavelength with a measurement speed of 110 km/h, the duration of its response can be 0.26 ms. Third, the responses of light squats in ABA signals are affected by the variability of the railway track parameters and measurement conditions. For example, different dynamic responses occur at squats on top or in between sleepers, thermite welds, flash welds, joints, crossings, transition zones, etc. Last and most importantly, the frequency components of light squats are slightly different from those of healthy rails, with subtle characteristics occurring dominantly at high frequencies, specifically in the range of 1000–2000 Hz. Therefore, advanced detection methods are essential for identifying these defects early and accurately.

2.2. Spiking neural networks

A spiking neural network (SNN) is brain-inspired and is a class of neural networks that more closely mimic the functioning of biological brains compared to traditional neural networks [12]. Unlike the other two generations, inputs of SNNs are encoded into temporal information represented as trains of spiking events rather than numeric values.

2.2.1. Spiking neuron models

The core component of an SNN is spiking neurons. They communicate with each other by generating and propagating electrical pulses known as action potentials or spikes. The likelihood of a spike generation depends on the types of synaptic inputs, i.e., excitatory or inhibitory. Excitatory inputs enhance the likelihood of a neuron firing a spike, whereas inhibitory inputs decrease this likelihood. The dynamics of a spiking neuron are characterised by its membrane potential, which evolves according to the excitatory and inhibitory inputs it receives from presynaptic neurons. When the membrane potential reaches a certain threshold, the neuron generates one or more spikes. As a result, the internal potential of each neuron has to be computed for a continuous duration of time to obtain the precisely timed patterns of spikes [27]. Producing a continuous time-encoded output and connecting to other neurons, SNNs offer an ability to deal with temporal and spatiotemporal patterns.

The most widely used model to describe the dynamics of the spiking neurons is the leaky integrate-and-fire (LIF) neuron model. The LIF model integrates incoming spikes until the membrane potential $v(t)$ reaches a threshold v_{th}, at which point the neuron fires (emits a spike) and resets its potential. The dynamics of the LIF model are described by the following formula [35]:

$$\tau_{\text{m}} \frac{dv(t)}{dt} = -(v(t) - v_{\text{rest}}) + I_{\text{exc}}(t) - I_{\text{inh}}(t), \tag{1}$$

if $v(t) \geq v_{\text{th}}$, then $v(t) \leftarrow v_{\text{rest}}$ and emit a spike,

where τ_{m} is the membrane time constant, v_{rest} is the resting potential, $I_{\text{exc}}(t)$ and $I_{\text{inh}}(t)$ are the excitatory and inhibitory currents, respectively. More spiking neuron models can be found in [36].

2.3. Information transmission and processing

Synapses in SNNs are the connections between neurons, where the presynaptic neuron sends signals and the postsynaptic neuron receives them. When a presynaptic neuron fires, a spike is transmitted to another neuron across synapses, in which synaptic weights play a crucial role as they determine the strength and impact of these transmitted spikes on the postsynaptic neuron. The postsynaptic neuron then responds to the incoming spike from multiple presynaptic neurons by integrating the signal into its membrane potential and generating spikes when the potential exceeds a certain threshold. In SNNs, information processing relies on the precise timing of these spikes, enabling efficient computation. The event-driven nature of SNNs allows for asynchronous processing [37], where neurons fire only when necessary, potentially reducing power consumption and enhancing response times compared to traditional neural networks [38]. These properties make SNNs particularly well-suited for applications requiring real-time processing and energy efficiency [37,38].

2.4. Learning

Synaptic plasticity plays an important role in reaching the high-level performance of SNNs. It is the process of learning and adjusting the synaptic weights to perform a given task. Various approaches have been proposed to adjust the synaptic weights. The spike-timing-dependent plasticity (STDP) is an approach based on a bio-inspired formulation of synaptic plasticity. The STDP updates synaptic weights in an unsupervised manner based on dependencies between presynaptic and postsynaptic spikes. Learning algorithms developed based on the STDP are, for instance, Tempotron [26], SWAT [28], ReSuMe [29], TMM-SNN [39], SEFRON [30], and reward-modulation spike-timing-dependent plasticity (R-STDP) [40]. The SWAT, TMM-SNN, and R-STDP algorithms successfully implemented the classification of multi-layer feed-forward SNNs, while the Tempotron, ReSuMe, and SEFRON algorithms were successful for a single-layer SNN. Another approach for training SNNs considers the backpropagation algorithm [41]. In the case of SNNs, the discontinuity mechanism between the internal state potential and the response of spiking neurons prevents the direct use of backpropagation. In [24], the SpikeProp deployed backpropagation with the assumption that a piece-wise linear function can approximate the internal potential at an infinitesimal time around the instant of neuronal firing. The learning rule of SpikeProp has been successfully extended from single to multiple spikes to enhance the training performance of SNNs [25,27].

Inspired by [30], the SNN in this paper is designed with time-varying weights and contains no hidden layers. We use the SpikeProp to deal with the discontinuity when a spike is fired. Although backpropagation is traditionally associated with multi-layer networks, this work adapts it as an efficient framework for computing and updating the weights in a single-layer SNN based on the gradient of the loss function. A gradient-based approach is then used to adjust time-varying weights by including an additional term that captures the variations over time in the update rule of the weights. Multi-starts are considered, and a genetic algorithm is employed to search for the optimal hyper-parameters based on cross-validation.

3. Squat detection problem

The detection of squats using ABA measurements can be considered a classification problem. This problem assumes availability of training samples $\mathcal{H}_D = \left\{\left(\mathbf{x}^{(d)}, c^{(d)}\right),\ \ d = 1, \dots, D\right\}$ in which an input $\mathbf{x}^{(d)} \in \mathbb{R}^M$. and its class label $c^{(d)} \in \mathbb{N}$ are required. Assigning a respective class label for ABA measurements at rails is a tedious and time-consuming process. Additionally, when dealing with squats at an early stage of their development, multiple data sources are required to confirm their existence and it is common to rely on field observation which is prone to human error. Moreover, it is difficult to obtain class information for defective and healthy samples because early squats can be invisible to the human eye or video cameras. Also, rails are affected by local infrastructure conditions, local railway track dynamics, and different stochastic variables. In this paper, labelling was carried out by human domain experts and was verified by fieldwork.

To obtain an estimator of the mapping between measured ABA signals at rails and their class label, an SNN-based methodology is proposed. Fig. 3 illustrates the framework of early detection of rail squats considered in this paper. It comprises two main parts: the feature engineering to obtain the representations of the measured ABA signals at rails (see Section 4) and the spiking neural network-based methodology that classifies whether or not a given rail segment contains squats. The latter part is detailed in Section 5.

4. Feature engineering

The data used in our work were collected using the ABA system operating at a sampling frequency of 25.6 kHz, which produces time-equidistant samples. The original acceleration signals for each rail segment are recorded in the time domain. GPS data is subsequently used to map the time domain signals to their corresponding spatial locations along the railway track. This mapping facilitates the association of spectral content with specific track locations, which is crucial for understanding where particular defects occur.

Fig. 4 presents the processes of feature engineering considered in this work. The original acceleration signals of each rail sample are first pre-processed by means of wavelet analysis with the Morlet function to realise the frequency content of the rail dynamics. A moving standard deviation is proposed to extract representative features for ABA measurements in the frequency domain. This selection is motivated by its ability to effectively capture local variations in the ABA signal intensity, making it particularly suitable for identifying abrupt changes that occur due to defects. Then, data are represented by concatenating representative features from vertical and longitudinal ABAs.

4.1. Wavelet analysis

To analyse the frequency content of the signal, we employ wavelet analysis due to its independence between the window size and the time–frequency representation. In this work, the CWT is selected because it provides a continuous representation of the signal, offering higher resolution and detail in both the time and frequency domains. This is particularly advantageous for detecting small rail defects such as squats. The CWT is a time–frequency analysis tool in which the observed function is multiplied by a group of shifted and scaled wavelet functions. To achieve computational feasibility, we discretised the CWT and the wavelet coefficient $W_n(s)$ at a discrete wavelet scale $s > 0$ and time index n are defined as [2]:

$$W_n(s) = \sum_{n'=0}^{T_w-1} a_{n'} \psi^{\star} \left(\frac{\left(n' - n\right) \delta_t}{s} \right), \tag{2}$$

where a_n is a time series with a time step of δ_t; $n' = 0, \dots, T_w - 1$ is the time shift operator where T_w is time window of the signal; $\psi^{\star}$ is a family of wavelets deduced from the mother wavelet by different translations and scaling; $\star$ indicates a complex conjugate. In this paper, we consider the Morlet function for the mother wavelet. The Morlet function is defined as:

$$\psi_0(\eta) = \pi^{-1/4} e^{i\omega_0 \eta} e^{-\eta^2/2}, \tag{3}$$

where ψ_0 is a nondimensional frequency. The power spectrum of a wavelet transform is defined as the square of the wavelet coefficients, i.e.,

$$\left| W_n^2(s) \right|. \tag{4}$$

Note that the frequency scale s obtained from our wavelet analysis has logarithmic spacing. This is due to the adjustable length of each wavelet. At higher frequencies, the length is shorter. Thus, higher frequencies have a larger bandwidth and are spaced further apart than lower frequencies. In this work, we select 113 wavelet scales as they provide a good trade-off between computational time for the wavelet analysis and the SNN and representation of the responses of rail dynamics at defects.

It is worth noting that CWT can be computationally intensive when processing long track segments and large datasets. The time complexity of the CWT is influenced by both the signal length and the number of scales, whereas the DWT's time complexity depends solely on the signal length. When a smaller number of scales is used, the computational demands of CWT can be competitive with DWT. Nevertheless, the enhanced resolution and continuity offered by the CWT can make this additional computational cost worthwhile for defect detection applications. The DWT, while more computationally efficient, produces a discrete representation that may lose some of the continuity and detail needed for accurate defect detection. For applications where computational efficiency is a priority, the DWT offers a practical balance between resolution and computational efficiency. Future research can include a comparison between the capabilities of CWT and DWT in extracting features from ABA signals for rail defect detection.

To address the computational feasibility of CWT, particularly for long track segments, our implementation processes the wavelet transform into overlapping batches. This strategy eliminates edge effects while allowing seamless stitching of transformed outputs. Although CWT is computationally intensive, we adopt GPU acceleration and batch optimisation techniques, enabling near real-time performance. In our application, the transformed data is not stored but processed on-the-fly to reduce memory usage (which would otherwise exceed 1 TB/hour).

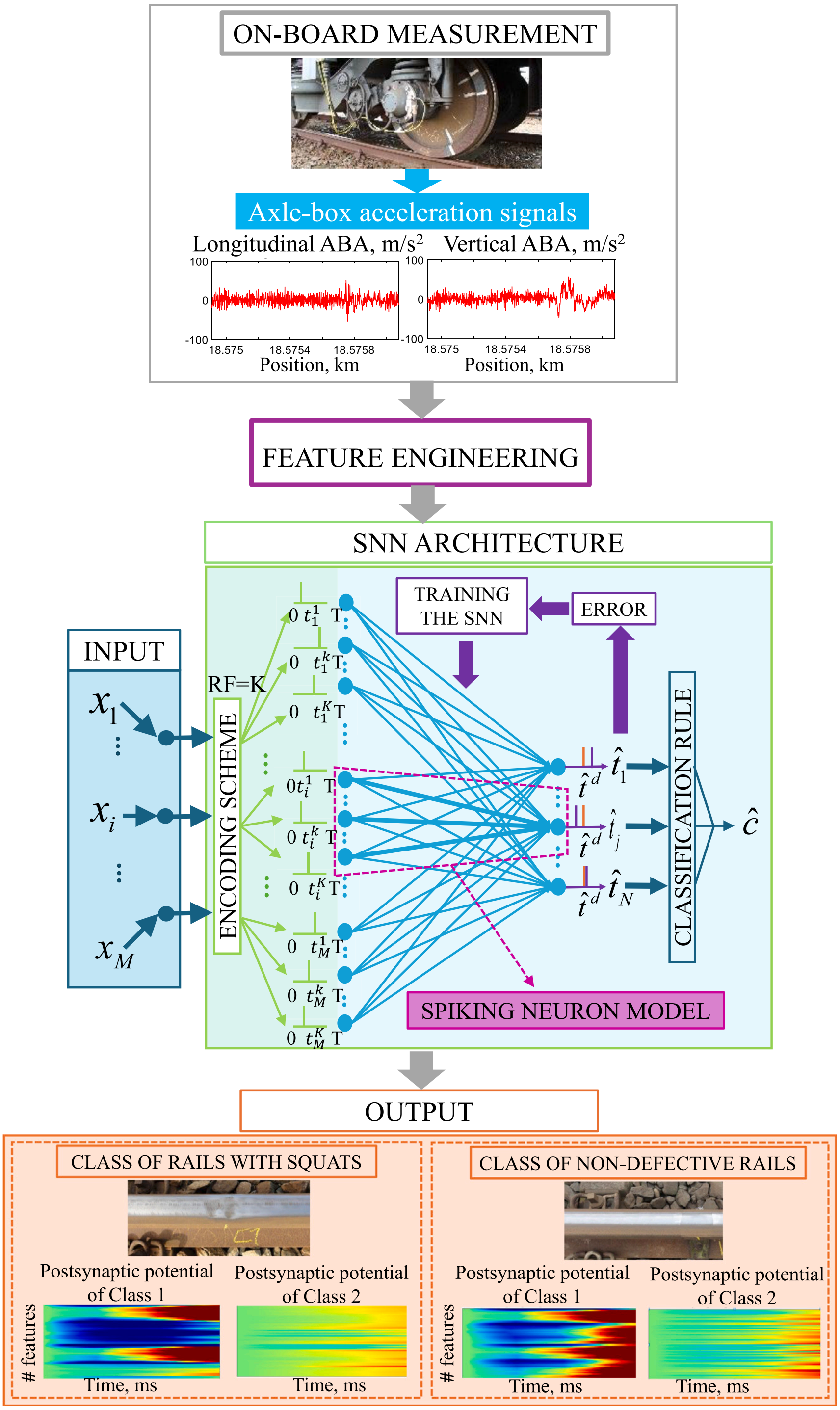


Fig. 3. The framework of spiking neural network with time-varying weights for detecting rail squats.

4.2. Feature extraction using moving standard deviation

For a vector of the wavelet power spectrum at a wavelet scale s, $\mathbf{W}(s) = \left[\left|W_1^2(s)\right|, \ldots, \left|W_{T_w}^2(s)\right|\right] \in \mathbb{R}^{1\times T_w}$, the moving standard deviation $\mathbf{Z}(s) = \left[z_1(s), \ldots, z_q(s), \ldots, z_Q(s)\right] \in \mathbb{R}^{1\times Q}$ of $\mathbf{W}(s)$ is obtained by calculating a standard deviation over a sliding window of length k_w across the T_w neighbouring elements of $\mathbf{W}(s)$. When k_w is odd, the window is centred about the element in the current position. When k_w is even, the window is centred on the current and previous elements. The window size is automatically truncated at the endpoints when there are not enough elements to fill the window, i.e., $Q = \left(T_w - k_w + 1\right)$. When the window is truncated, the standard deviation is taken over only the elements that fill the window. Mathematically, the moving standard deviation $\mathbf{Z}(s)$ of the sliding window of length k_w is expressed as:

$$z_q(s) = \sqrt{\frac{\sum_{i=q}^{q+k_w-1}\left(\left|W_i^2(s)\right| - \bar{W}(s)\right)^2}{k_w - 1}}; q = k_w, \ldots, Q, \tag{5}$$

where $\bar{W}(s)$ is the mean of $\left[\left|W_q^2(s)\right|, \ldots, |W_{q+k_w-1}^2(s)|\right]$. The representative feature at each frequency scale s, $x(s)$, is then obtained by:

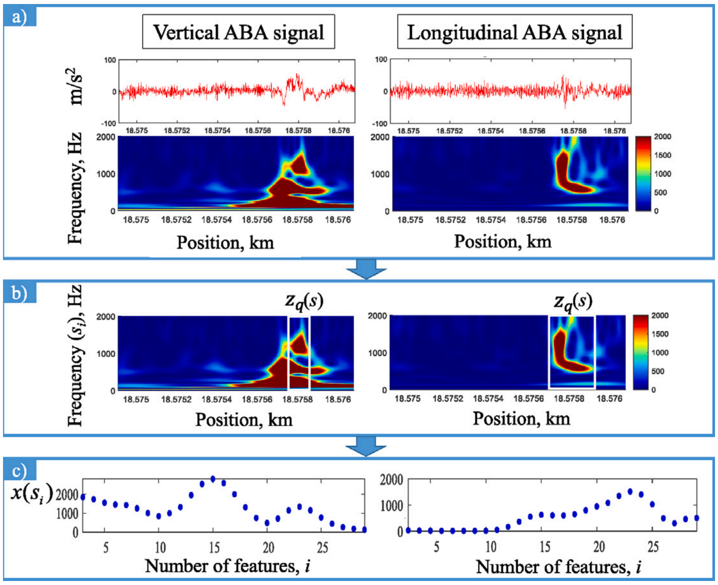


Fig. 4. Steps in feature engineering: (a) Wavelet transform (CWT) applied to the ABA signals, showing the frequency content at different spatial positions. (b) Feature extraction by moving standard deviation, in which the white lines in the CWT plots indicate the window used for computing the moving standard deviation. (c) Selected features after the application of the proposed feature selection method.

$$x(s) = \max_{q} z_q(s). \tag{6}$$

In this paper, a sensitivity analysis is performed to obtain the appropriate length of sliding windows, k_w, that is used to extract features from ABA signals in the vertical and longitudinal directions. With the length window k_w moving along a rail line as shown in Fig. 4, variation of the energy contents can be realised. In this work, the energy contents of the signal are analysed in two dimensions: along the time domain, which can be mapped to distances along the track, and across different frequencies. The energy contents along distances are used to represent the temporal characteristics of the rail, reflecting how dynamic responses of rail change over short timescales (e.g., seconds to minutes) as the train moves. This analysis captures short-term variations rather than long-term temporal changes over months or years. Conversely, the energy contents across different frequencies provide insight into the spatial characteristics, representing variations in the rail's condition at different spatial wavelengths. Using (6), these temporal pieces of information are embedded per frequency. When considering a range of frequencies (wavelet scales), the corresponding temporal features $x(s)$ form one-dimensional frequency-based inputs. These inputs preserve critical spatiotemporal characteristics, as the temporal information is represented by the peak variations derived from the moving standard deviation, and the spatial information is encoded in the frequency (wavelet scale) corresponding to each feature. This approach simplifies yet retains the essential spatiotemporal characteristics of rail defects, which are then used as inputs to the SNN.

It is worth noting that, due to the principle of frequency-dependent resolution inherent in the CWT, using a constant window length for the moving standard deviation may result in sensitivity to higher frequencies, as the CWT provides higher temporal resolution at those frequencies. Exploring a varying window length that adapts to the CWT's frequency-dependent resolution could be a valuable direction for future work to refine the feature extraction process further.

4.3. Feature selection

Our feature selection process is guided by the wavelength–frequency relationship and prior studies [2]. Specifically, the range of wavelet scales between 200–2000 Hz was chosen because these frequencies capture the most relevant frequency components of ABA responses associated with squats [2]. These scales emphasise localised variations in signal intensity, which are significant for capturing abrupt changes caused by squats without introducing noise from unrelated low- and high-frequency components. The selected wavelet scales are described as:

$$\mathbf{x} = \left\{ x_i = x\left(s_i\right); s_i \in [200, 2000] \right\}. \tag{7}$$

Considering ABA signals from both vertical and longitudinal directions [9], particularly longitudinal ABA increases the detection of light squats, a total of 54 frequency-based features are used to represent the dynamic of rails in this paper. For further research, it is interesting to explore a combination of data-based and physics-based approaches to determine the optimal resolution for the methodology. While the current study uses bands chosen in logarithmic ranges, alternative partitioning or other frequency-based features could be of interest for effective detection.

5. Spiking neural network with time-varying weights methodology

The SNN-based methodology consists of five main aspects: the temporal spike encoding scheme, the spiking neuron model, the network architecture, the methodology used to train the SNN, and the classification rule.

5.1. Encoding scheme

This work considers scaling the input into the interval [0,1]. Specifically, for a given input $\mathbf{x} = \left[x_1, \ldots, x_i, \ldots, x_M\right]^{\mathrm{T}} \in [0,1]^M$, each feature x_i must be converted into spike events. In this paper, we consider the most used population rank encoding scheme to convert information into spike events [42]. Unlike time-to-first spike encoding, which uses the precise timing of the first spike, and rate encoding, which uses the firing rate, population rank encoding represents information based on the pattern of neuron activation rather than the timing order or firing rate alone.

In this paper, the scheme uses overlapping Gaussian receptive field neurons to encode each input feature x_i into spike times. The firing strength of the ith input feature $r_k\left(x_i\right)$ emitted by the Gaussian

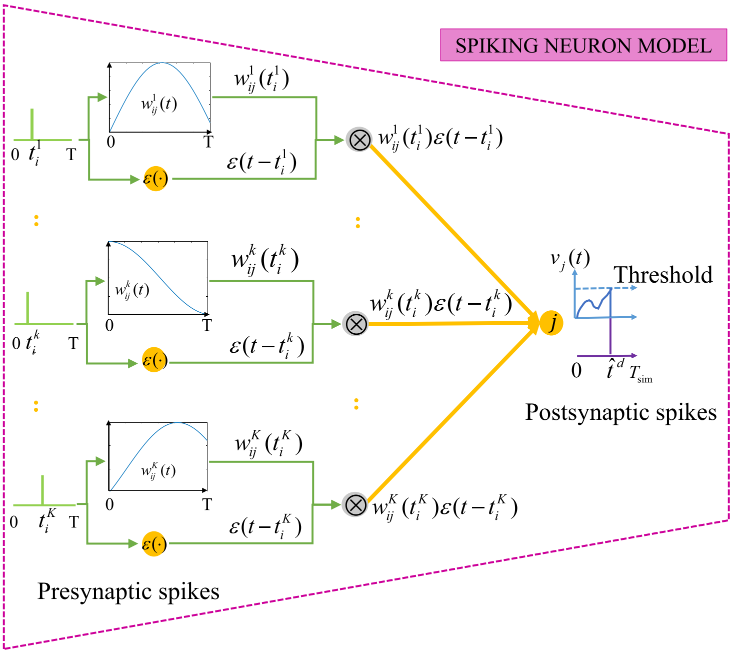


Fig. 5. The spiking neuron model.

receptive field neuron k is defined as [31]:

$$r_k\left(x_i\right) = \exp\frac{-\left(x_i-\delta_k\right)^2}{2\sigma^2}, \tag{8}$$

$$\delta_k = \frac{2k-3}{2(K-2)}, \tag{9}$$

and

$$\sigma = \frac{1}{\gamma}\cdot\frac{1}{K-2}, \tag{10}$$

where K represents the number of receptive field neurons (given by the number of populations with different Gaussian receptive fields), and γ the overlap constant, the centre δ_k and the width σ of the Gaussian function are defined according to the range of the input features.

Therefore, each feature x_i is converted into K presynaptic spikes of firing strength $\left[r_1\left(x_i\right),\ldots,r_K\left(x_i\right)\right]^{\mathrm{T}}$ using (8). The associated firing times $\mathbf{t}_i = \left[t_i^1,\ \ldots,t_i^K\right]^{\mathrm{T}} \in [0,T]^K$ of the feature x_i are obtained by rescaling $r_k\left(x_i\right)$, $k=1,\ldots,K$ into the presynaptic spike time interval of $[0,T]$ ms. It is worth mentioning that the unit for T is milliseconds (ms) because neuronal firing and synaptic processes typically occur over millisecond timescales, consistent with biological neuron behaviour [43]. This choice aligns with conventions in other SNN research, where temporal dynamics are modelled on biologically inspired time scales. Using seconds (s) would lack the precision needed to represent the precise timing critical for spike-based computations.

5.2. Spiking neuron model

In this paper, a spiking neuron model based on the leaky-integrate-and-fire model [44] is adopted, as illustrated in Fig. 5. Our SNN is designed with time-varying synaptic weights, $w_{ij}^k(t)\in[0,T]$, to connect between the presynaptic neurons associated with the kth spike time, t_i^k, of the input feature x_i and the postsynaptic neuron j.

A postsynaptic potential $S_{ij}^k(t)$ of the output neuron j at time t is determined as the product of the spike response of the presynaptic spike t_i^k, $\epsilon\left(t-t_i^k\right)$, and the time-varying synaptic weight $w_{ij}^k(t)$ evaluated at $t=t_i^k$, $w_{ij}^k\left(t_i^k\right)$, which is mathematically expressed as [30]:

$$S_{ij}^k(t) = w_{ij}^k\left(t_i^k\right)\cdot\epsilon\left(t-t_i^k\right), \tag{11}$$

$$\epsilon(t) = \begin{cases} 0 & \text{if } t\le 0,\\ \dfrac{t}{\tau}\exp\left(1-\dfrac{t}{\tau}\right) & \text{if } t>0, \end{cases} \tag{12}$$

where τ is the time constant of the spiking neuron. A membrane potential $v_j(t)$ of the output neuron j is defined as the summation of a postsynaptic potential $S_{ij}^k(t)$ over the input spikes of x_i, $\mathbf{t}_i = \left[t_i^1,\ldots,t_i^K\right]^{\mathrm{T}}$. A membrane potential $v_j(t)$ is expressed as [30]:

$$v_j(t) = \sum_{i=1}^{M}\sum_{k=1}^{K} S_{ij}^k(t) \tag{13}$$

$$= \sum_{i=1}^{M}\sum_{k=1}^{K} w_{ij}^k\left(t_i^k\right)\cdot\epsilon\left(t-t_i^k\right). \tag{14}$$

The neuron fires a postsynaptic spike when the membrane potential reaches the firing threshold Λ. The postsynaptic firing time $\hat{t}_j$ is defined as:

$$\hat{t}_j = \left\{t|v_j\left(t\right)\ge\Lambda\right\}. \tag{15}$$

At the postsynaptic firing time $\hat{t}_j$, the membrane potential of the neuron j is defined as:

$$v_j\left(\hat{t}_j\right) = \sum_{i=1}^{M}\sum_{k=1}^{K} w_{ij}^k\left(t_i^k\right)\cdot\epsilon\left(\hat{t}_j-t_i^k\right), \tag{16}$$

where $\hat{t}_j$ lies in the postsynaptic spike time interval of $[0,T_{\mathrm{sim}}]$ ms. The parameter T_{sim} represents the simulation time used for the internal potential to reach the threshold of the neuron. Its value can be set differently [30], and it is a problem-dependent parameter. Note that our paper handles the postsynaptic firing time that does not reach the threshold by setting it to the upper limit of the time window.

5.3. Spiking neural network architecture

This paper considers a two-layered fully connected feedforward SNN with no hidden layers and no hidden nodes, as illustrated in Fig. 3. For K receptive field neurons, an input $\mathbf{x}$ is encoded into the presynaptic input spike time $\mathbf{t} = [\mathbf{t}_1,\ldots,\mathbf{t}_M]^T \in [0,T]^{M\times K}$, in which the number $K\times M$ determines the number of input neurons for our SNN architecture. An output neuron is designed to associate with one of the N classes. Therefore, the network architecture comprises N output

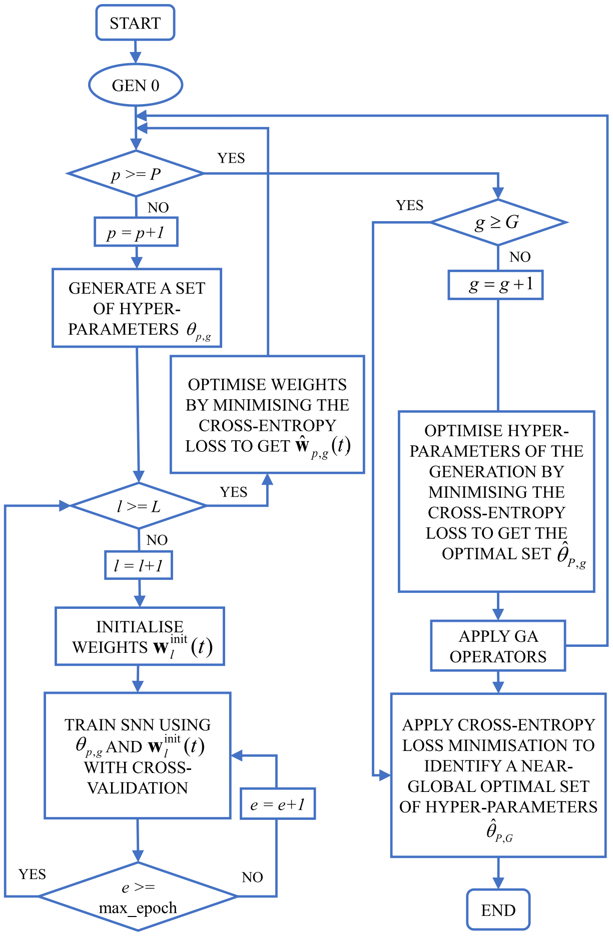

START
GEN 0
$p >= P$
YES
NO
$p = p+1$
GENERATE A SET OF HYPER-PARAMETERS $\theta_{p,g}$
OPTIMISE WEIGHTS BY MINIMISING THE CROSS-ENTROPY LOSS TO GET $\hat{\mathbf{w}}_{p,g}(t)$
$l >= L$
YES
NO
$l = l+1$
INITIALISE WEIGHTS $\mathbf{w}_l^{\text{init}}(t)$
TRAIN SNN USING $\theta_{p,g}$ AND $\mathbf{w}_l^{\text{init}}(t)$ WITH CROSS-VALIDATION
$e = e+1$
YES
$e >=$ max_epoch
NO
YES
$g \geq G$
NO
$g = g+1$
OPTIMISE HYPER-PARAMETERS OF THE GENERATION BY MINIMISING THE CROSS-ENTROPY LOSS TO GET THE OPTIMAL SET $\hat{\theta}_{P,g}$
APPLY GA OPERATORS
APPLY CROSS-ENTROPY LOSS MINIMISATION TO IDENTIFY A NEAR-GLOBAL OPTIMAL SET OF HYPER-PARAMETERS $\hat{\theta}_{P,G}$
END

Fig. 6. Schematic diagram of the SNN training.

neurons. This architecture is referred to as $K \times M$: N and constitutes a total of $K \times M \times N$ time-varying synaptic weights to determine.

5.4. Training the SNN

For the proposed SNN, three groups of parameters are defined: a group of given parameters, a group of hyper-parameters, and a group of initial synaptic weights and synaptic weights. The first group contains parameters that are assumed given as in [30]. The parameters and their associated value are encoding-related parameters with $K = 6$ and $\gamma = 0.7$, the presynaptic spike interval $T = 3$ ms, and the postsynaptic spike interval $T_{\text{sim}} = 4$ ms. A time resolution of 0.01 ms is considered for all synapses in the network.

To obtain the value of parameters of the second and the third group, an SNN-based methodology is proposed as illustrated in Fig. 6. To obtain parameters as close to optimal global ones as possible, three main steps considered in the methodology include (1) hyper-parameter tuning with a genetic algorithm (GA), (2) weight initialisation with multi-starts and (3) weight updating with a gradient-based approach including the effects of the time dependency of the weights in the update rule.

5.4.1. Hyper-parameter tuning with genetic algorithm

To achieve a near-global optimal set of hyper-parameters, $\hat{\theta}$, we perform a genetic algorithm considering the number of individuals, $p \in \{1, \dots, P\}$ and the number of generations, $g \in \{1, \dots, G\}$. The hyper-parameters tuned by a GA include the time-varying weight kernel ($\sigma \in [0.05, 0.55]$) [30], the desired postsynaptic firing time ($\hat{t}^d \in [0, T_{\text{sim}}]$), the time constant of spike response function ($\tau \geq T$), and learning rate of weight update (η). The time constant of the spike response function is set to be greater than T as only a single spike is allowed from the output neuron. The maximum value of τ is considered to be 4 ms as a higher value of τ results in difficulty in raising the potential towards the threshold. The range of each hyper-parameter shown in Table 1 is determined to search for a good classification of all benchmarks.

Table 1
A range of the hyper-parameters considered for classification.

Hyper-parameter	Range	Search step size
$\hat{t}^d$	[0.05,3.5]	0.05
σ	[0.05,0.55]	0.05
η	[0.0001,0.001]	0.00005
τ	[3.0,4.0]	0.01

Throughout the evolution process of the GA, individuals of each generation, $\theta_{p,g}$, are obtained via the process of crossover and mutation. The two parents are chosen randomly from the whole population of the present generation. Then, two off-springs are bred by swapping the tails of their two parents at a random crossover point. The tuning parameters of GA are selected by default in the toolbox. A sensitivity analysis can be conducted to determine a reasonable number of generations and individuals.

Using a given set of hyper-parameter values $\theta_{p,g}$, the SNN is trained using the multi-start and cross-validation strategy to obtain the optimal set of time-varying weights $\hat{\mathbf{w}}_{p,g}(t)$. Then, the different P combinations of hyper-parameter values are computed and evaluated to find the near-global optimal set of hyper-parameters of the generation g, $\hat{\theta}_{P,g}$. Finally, the set of hyper-parameter values $\hat{\theta}_{P,g}, g = 1, \dots, G$ that achieves the best solution is selected as the set of hyper-parameters, $\hat{\theta}_{P,G}$.

5.4.2. Weight initialisation with multi-starts

For a given $K \times M$ input spike times and the corresponding N output synaptic neurons, which is equal to the number of classes in the classification task, the total number of $K \times M \times N$ synaptic weights are initialised by the uniformly distributed random numbers between 0 and 1. We limit the range of the initial synaptic weights to make sure that all neurons fire within the simulation time in the first epoch of network training. These initial weights are then distributed over the time interval by employing (23) to obtain time-varying weights, $\mathbf{w}^{\text{init}}(t) = \left\{ w_{ij}^{k,\text{init}}(t); k = 1, \dots, K, i = 1, \dots, M, \quad j = 1, \dots, N \right\}$. In this paper, the training process of the synaptic weights are repeated according to the multiple sets of initial weights $\{\mathbf{w}_l^{\text{init}}(t); l = 1, \dots, L\}$, in which L is the number of multi-starts.

5.4.3. Time-varying weights updating with a gradient-based approach

Using encoding, each input $\mathbf{x}$ gets a unique spike representation $\mathbf{t}$. The true class labels are also transformed into desired spike times by mapping each class to a predefined unique spike time within a specific time window. The desired spike times should be spaced apart to create a clear temporal distinction between the classes during training. This time-based encoding ensures that the SNN can distinguish between different classes based on when spikes occur. However, the desired spike time can be considered a hyperparameter, as its value directly impacts the model's performance. For a given set of hyper-parameter values $\theta_{p,g}$ and an initial set of time-varying weights $\mathbf{w}_l^{\text{init}}(t)$, the predicted postsynaptic output spike times, $\hat{\mathbf{t}}^a$, are obtained. Then, the error ($\mathcal{L}$) obtained at the output neurons between the predicted and the desired spike time is used to update the synaptic weights between the ith presynaptic and the jth postsynaptic neurons.

In our approach, the time-varying weights are updated by using SpikeProp as proposed in [24] and a gradient-based approach update rule that includes the effect of time in the weights. For epoch e, the synaptic weights, $w_{ij}^{k,e}(t)$ at a single time instance $t = t_i^k$ is adjusted by:

$$\Delta w_{ij}^{k,e}\left(t_i^k\right) = -\eta \frac{\partial \mathcal{L}}{\partial w_{ij}^{k,e}\left(t_i^k\right)}. \tag{17}$$

where η is the learning rate and the sum squared error ($\mathcal{L}$) is used as a measure of the discrepancy between the desired and the predicted postsynaptic spike times. As $\hat{t}_j$ is a function of v_j, which depends on the weights w_{ij}^k at time instance t_i^k, the derivative on the right-hand part of (17) can be expanded to:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{k,e}\left(t_i^k\right)} = \frac{\partial \mathcal{L}}{\partial \hat{t}_j}\left(\hat{t}_j^a\right) \frac{\partial \hat{t}_j}{\partial v_j\left(\hat{t}_j\right)}\left(\hat{t}_j^a\right) \frac{\partial v_j\left(\hat{t}_j\right)}{\partial w_{ij}^{k,e}\left(t_i^k\right)}\left(\hat{t}_j^a\right). \tag{18}$$

As given in [24], the first and the third derivative terms can be expressed coordinatewise for $j = 1, \dots, N$ as:

$$\frac{\partial \mathcal{L}}{\partial \hat{t}_j} = \hat{t}_j^a - \hat{t}^d, \tag{19}$$

$$\frac{\partial v_j\left(\hat{t}_j^a\right)}{\partial w_{ij}^{k,e}\left(t_i^k\right)} = \epsilon\left(\hat{t}_j^a - t_i^k\right), \tag{20}$$

and, the second derivative term can be obtained by following [24] as:

$$\frac{\partial \hat{t}_j}{\partial v_j\left(\hat{t}_j\right)}\left(\hat{t}_j^a\right) = \frac{-1}{\partial v_j\left(\hat{t}_j\right)/\partial \hat{t}_j}\left(\hat{t}_j^a\right) = \frac{-1}{\partial v_j\left(\hat{t}_j^a\right)/\partial \hat{t}_j^a}. \tag{21}$$

Considering the denominator, we have

$$\begin{aligned} \frac{\partial v_j\left(\hat{t}_j^a\right)}{\partial \hat{t}_j^a} &= \frac{\partial}{\partial \hat{t}_j^a}\left(\sum_{i=1}^{M}\sum_{k=1}^{K} w_{ij}^{k,e}\left(t_i^k\right)\cdot \epsilon\left(\hat{t}_j^a - t_i^k\right)\right) \\ &= \sum_{i=1}^{M}\sum_{k=1}^{K} w_{ij}^{k,e}\left(t_i^k\right) \frac{\partial \epsilon\left(\hat{t}_j^a - t_i^k\right)}{\partial \hat{t}_j^a} \\ &= \sum_{i=1}^{M}\sum_{k=1}^{K} w_{ij}^{k,e}\left(t_i^k\right)\epsilon\left(\hat{t}_j^a - t_i^k\right)\left(\frac{1}{\hat{t}_j^a - t_i^k} - \frac{1}{\tau}\right). \end{aligned} \tag{22}$$

Substituting (22) into (21), the second derivative term is obtained. Hence, the error gradient at the postsynaptic spike time of an output neuron j at $t = \hat{t}_j$ is obtained using (19), (20), and (21).

Next, we include short-term plasticity by distributing the long-term plasticity over a specific time interval. Following [30], a Gaussian distribution is chosen as the modulating function. Therefore, $\Delta w_{ij}^{k,e}\left(t_i^k\right)$ at single time instance t_i^k is embedded into a Gaussian distribution such that a time-varying function $\Gamma_{ij}^k(t)$ is described as:

$$\Gamma_{ij}^{k,e}(t) = \Delta w_{ij}^{k,e}\left(t_i^k\right)\cdot \exp\left(\frac{-\left(t - t_i^k\right)^2}{2\sigma^2}\right), \tag{23}$$

where σ is the efficacy update range. The use of the exponential function to describe time-varying features of synaptic weights is followed from [30] to achieve biological plausibility. It is described in [45,46] that neural behaviour and learning processes in the human brain involve changes in synaptic potential that follow exponential decay or growth patterns. While exponential functions are often employed to model these temporal dynamics, an evaluation of the impact of different time-varying functions can be interesting.

Hence, the update rule of time-varying synaptic weights for the synapse connected between the presynaptic neuron i and the postsynaptic neuron j is:

$$w_{ij}^{k,e+1}(t) = w_{ij}^{k,e}(t) + \Gamma_{ij}^{k,e}(t), e = 1, \dots, E, \tag{24}$$

in which E denotes the maximum number of training epochs used for a simulation.

Note that (24) is used to obtain the weight parameters for each cross-validation fold. To obtain the optimal set of time-varying weights, k-fold cross-validation is considered to train and test the SNN with k different portions of the dataset. Then, the optimal set of time-varying weights is obtained by minimising the cross-entropy loss. This is applied after determining the predicted class from the spike times.

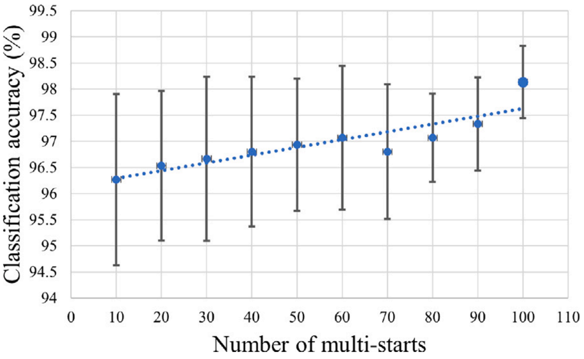


Fig. 7. The effect of the number of multi-starts.

5.5. Classification rule

In this paper, the transmission types are ruled by the time-to-first spike coding scheme. Our output neuron can emit only once and its output associated with class j is the first postsynaptic spike time $\hat{t}_j$. For a problem with N classes, the classification rule to predict the class label from the postsynaptic spike times $\hat{\mathbf{t}} = \left[\hat{t}_1, \dots, \hat{t}_j, \dots, \hat{t}_N\right]^T$ for an input $\mathbf{x}$ is defined as follow:

$$\hat{c} = \operatorname*{argmin}_j \hat{t}_j. \tag{25}$$

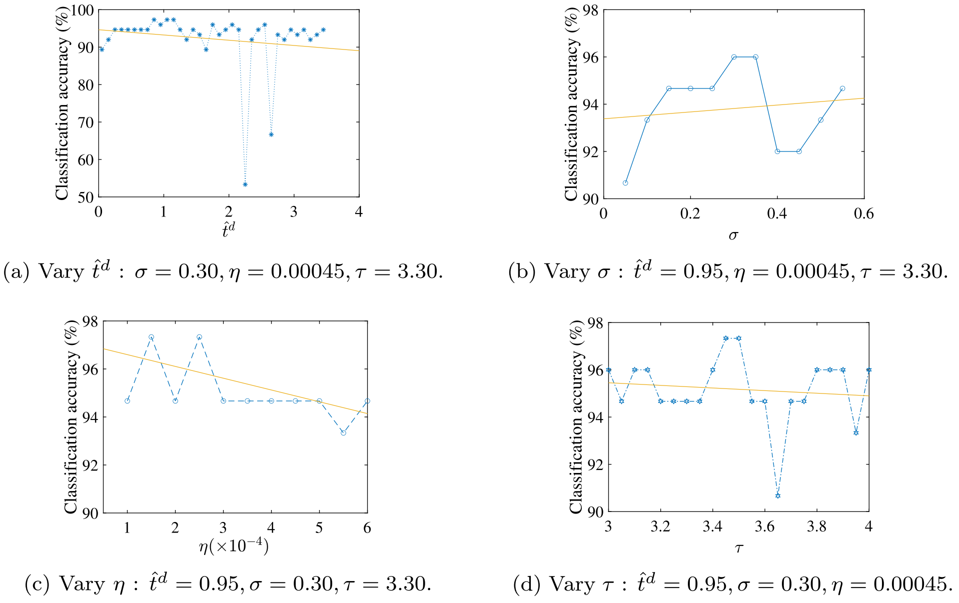
6. Comparative study using UCI benchmarks

In this section, we perform a sensitivity analysis of hyper-parameters and evaluate the classification performance of the proposed SNN-based methodology using UCI benchmarks. To address the challenge of training SNNs, we employ a global optimisation framework that integrates a genetic algorithm for hyperparameter tuning, multi-start initialisation for robustness, and a gradient-based learning rule adapted from SpikeProp. This combination allows us to effectively train a single-layer SNN with time-varying synaptic weights, despite the discontinuities inherent in spike-based computation. The presynaptic spike interval T is set to 3 ms, and the postsynaptic spike interval T_{sim} is 4 ms, with a time resolution of 0.01 ms, which ensures high temporal precision consistent with biologically inspired spike timing.

6.1. Sensitivity analysis

The Iris dataset is considered as a showcase for sensitivity analyses of the number of individuals and generations used in GA and a number of multi-starts. The Iris dataset contains 3 classes of 50 instances each [47]. Each class refers to a type of iris plant which are Iris Setosa (class 1), Iris Versicolour (class 2), and Iris Virginica (class 3). There are 4 input features for each instance: sepal length, sepal width, petal length, and petal width. In total, 75 instances are used for training and 75 instances are used for testing.

After the normalisation, each input feature is encoded by an array of six different one-dimensional Gaussian receptive fields. The network architecture for the Iris dataset, therefore, consists of 24 input neurons and 3 output neurons, referred to as 24:3. We followed the procedure presented in Fig. 6 to obtain the parameters that better fit the Iris dataset. The number of epochs considered in the analysis is 100 epochs which shows to be sufficient for the training.



(a) Vary $\hat{t}^d$: $\sigma = 0.30, \eta = 0.00045, \tau = 3.30$.

(b) Vary σ : $\hat{t}^d = 0.95, \eta = 0.00045, \tau = 3.30$.

(c) Vary η : $\hat{t}^d = 0.95, \sigma = 0.30, \tau = 3.30$.

(d) Vary τ : $\hat{t}^d = 0.95, \sigma = 0.30, \eta = 0.00045$.

Fig. 8. Effect of hyper-parameter variation for the SNN-based methodology — the Iris dataset.

6.1.1. Number of multi-starts

For the network architecture of 24:3, a total number of 72 synaptic weights are initialised and distributed over the time interval $[0, 3]$ ms. Given a set of hyper-parameters $\theta = \{\hat{t}^d = 0.95, \sigma = 0.30, \eta = 0.00045, \tau = 3.30\}$, we investigate the proposed SNN-based methodology in terms of classification accuracy when considering different numbers of multi-starts. Considering $I = 10, 20, \ldots, 90, 100$ and repeating the experiments ten times, Fig. 7 shows that the classification accuracy is distributed between 94.54% and 98.67%. It is observed that the higher the number of multi-starts, the lower the variance and the higher the average classification accuracy. However, there is a trade-off between the computational time and the use of a higher number of multi-starts.

6.1.2. Hyper-parameter variation

To understand the effect of the hyper-parameters in the SNN architecture, classification accuracy as a function of some hyper-parameters is presented in Fig. 8. It is observed that when the value of $\hat{t}^d$ increases, the obtained classification accuracy tends to decrease. The same trend is also observed for the variation of η and τ, whereas the obtained classification accuracy is enhanced by decreasing their value. For the value of σ, higher classification accuracy can be achieved by increasing its value. Furthermore, it is observed that the proposed methodology is sensitive to the variations of $\hat{t}^d$ more than the variation of other hyper-parameters.

6.1.3. Number of individuals and generations in genetic algorithm

Following the procedure in Fig. 6 with the number of multi-starts set to 60, we perform a sensitivity analysis for the number of individuals and generations in GA to obtain a set of hyper-parameters, $\hat{\theta}$, for the Iris dataset. We consider the numbers of individuals, $p \in \{1, \ldots, P\}, P = \{10, 30, 50, 70, 100\}$ and the number of generations, $g \in \{1, \ldots, G\}, G = 100$. Fig. 9 illustrates that the classification accuracy increases as the number of generations increases. Moreover, it is demonstrated that a larger population size can significantly improve accuracy using fewer generations. This observation allows understanding the effect of the GA parameters to obtain the hyper-parameter setting considering the Iris dataset. However, a GA expends more memory and more computational time in finding a solution for a large population size. Therefore, a compromise between accuracy and computational effort is to be considered for a reasonable trade-off.

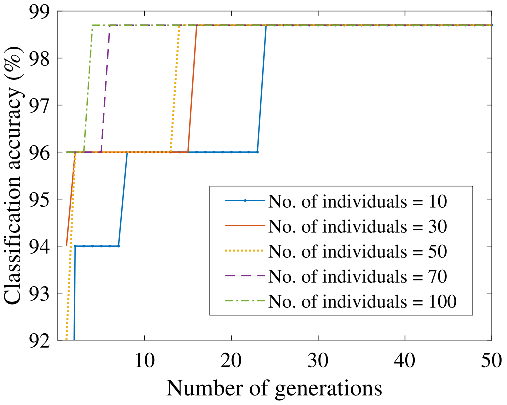


Fig. 9. The evolution of the classification accuracy for the best individual in a generation. The plot shows the classification accuracy at the last epoch (=100).

Table 2
A description of benchmark datasets from the UCI machine learning repository.

Dataset	Feature	No. of	No. of instances	
		class	Training	Testing
Liver disorders	6	2	170	175
Breast cancer	9	2	350	333
Ionosphere	33	2	175	176
Iris	4	3	75	75

6.2. Performance

First, the performance of the SNN-based methodology is examined using four benchmarks from the UCI machine learning repository [47]. Each UCI benchmark dataset presents unique challenges such as imbalanced classes, small sample sizes, high dimensionality, nonlinear separability, and noisy data [47]. These datasets have been widely used to benchmark proposed methods in several studies [28,30,31]. The datasets are described in Table 2. To solve the classification problems, our SNN is trained by following the procedure in Fig. 6. The tuned hyper-parameters are the time-varying weight kernel (σ), the desired postsynaptic firing time ($\hat{t}^d$), the time constant of spike response (τ), and the learning rate of weight update (η).

Table 3
The optimal set of hyper-parameter values and the corresponding classification accuracy for UCI benchmarks obtained from the proposed SNN-based methodology.

Benchmark	Hyper-parameter				Accuracy (%)				No. of
					Training		Testing		epoch
	$\hat{t}^d$	σ	η	τ	Max	Avg	Max	Avg	
Liver disorders	3.48	0.52	0.00015	3.09	95.4	91.8	75.6	70.2	100
Breast cancer	0.20	0.50	0.00080	3.45	96.5	96.3	98.8	98.7	100
Ionosphere	1.90	0.25	0.00010	3.00	100.0	99.5	97.7	94.3	100
Iris	0.95	0.30	0.00045	3.30	100.0	100.0	98.7	96.5	100

Table 4
Performance comparison with the other SNN methods. NB: the computational time shown in our SNN does not include the time required for multiple-start initialisation and GA optimisation.

Dataset	Method	Architecture	Avg. accuracy (%)		No. of	Training
			Training	Testing	epoch	time (s)
Liver disorders	SpikeProp [24]	37:15:2	71.5	65.1	3000	11.36
	SWAT [28]	36:468:2	74.8	60.9	500	93.04
	TMM-SNN [39]	36:5:8:2	74.2	70.4	442	4.16
	WOLIF [31]	19:1	81.9	**80.3**	500	–
	SEFRON [30]	37:1	91.5	67.7	100	1.1637
	Mc-SEFRON [32]	36:2	77.3	69.6	100	1.5784
	SNN with time-varying weights without GA	36:2	89.5	67.8	100	0.4401
	SNN with time-varying weights with GA	36:2	**91.8**	70.2	100	0.4401
Breast cancer	SpikeProp [24]	64:15:2	97.6	97.0	1500	16.07
	SWAT [28]	54:702:2	96.5	95.8	500	278.87
	TMM-SNN [39]	54:2:8:2	97.4	97.2	70	17.75
	WOLIF [31]	28:1	97.8	97.0	500	–
	SEFRON [30]	55:1	98.3	96.4	100	1.3202
	Mc-SEFRON [32]	54:2	98.4	97.4	100	1.7999
	SNN with time-varying weights without GA	54:2	**98.9**	97.2	100	0.7225
	SNN with time-varying weights with GA	54:2	96.3	**98.7**	100	0.7225
Ionosphere	SpikeProp [24]	199:25:2	89.0	86.5	3000	27.30
	SWAT [28]	198:2574:2	86.5	90.0	500	503.79
	TMM-SNN [39]	204:23:34:2	98.7	92.4	246	25.17
	WOLIF [31]	100:1	94.4	90.6	500	–
	SEFRON [30]	199:1	97.0	88.9	100	2.1686
	Mc-SEFRON [32]	204:2	94.2	89.7	100	3.5643
	SNN with time-varying weights without GA	198:2	97.1	91.5	100	0.8147
	SNN with time-varying weights with GA	198:2	**99.5**	**94.3**	100	0.8147
Iris	SpikeProp [24]	50:10:3	97.4	96.1	1000	9.01
	SWAT [28]	16:208:3	95.5	95.3	500	34.77
	TMM-SNN [39]	24:4:7:3	97.5	**97.2**	94	1.41
	WOLIF [31]	13:1	94.1	95.1	500	–
	Mc-SEFRON [32]	24:3	98.4	97.1	100	0.2097
	SNN with time-varying weights without GA	24:3	99.6	95.3	100	0.2626
	SNN with time-varying weights with GA	24:3	**100**	96.5	100	0.2626

Table 3 presents the optimal set of hyper-parameters and the corresponding classification accuracy for each benchmark obtained from our proposed SNN-based methodology. By setting the hyper-parameters as described, Table 3 shows that the classification accuracy for both training and test sets achieves more than 90% for Breast cancer, Ionosphere, and Iris datasets. The highest averaged accuracy is 100% and 98.7% for training and test sets, respectively. The obtained performance demonstrates the effectiveness of the proposed methodology in addressing the mentioned challenges in each dataset. However, it initially encountered poor performance on the liver disorder dataset, achieving only 67% accuracy. After the proposed global optimisation approach is employed to fine-tune network parameters, the liver disorder dataset accuracy is improved by 3.5% and overall performance is enhanced. Consequently, our methodology outperformed other methods like SpikeProp, SWAT, and SEFRON, as shown in Table 4.

6.3. Comparison with other SNN methods

Our methodology is compared with state-of-the-art SNNs. Three learning algorithms with larger SNN architecture, e.g. SpikeProp [24], SWAT [28] and TMM-SNN [39], and three learning algorithms with a simple SNN architecture (without hidden layers and nodes), e.g. SEFRON [30], Mc-SEFRON [32], and WOLIF [31] are selected. To quantify the advantages of the proposed global optimisation approach, a comparison of our methodology both with and without the use of GA is also included. For our methodology without the GA, the hyper-parameter values as suggested in [30] are used. However, our learning rate is set to 0.001 in order to increase the probability of convergence [24,27]. The comparative analysis considers the evaluation of averaged classification accuracy, the number of epochs used in each method, and the average training time used per epoch. Note that the performance results for SpikeProp, SWAT, TMM-SNN, and SEFRON, including both accuracy and training time, are reproduced from [30,48]. The computational time for our methodology with and without GA is conducted in MATLAB 2021b using a 64-bit operating system with Windows 10 OS in a CPU with 4 cores, 16 GB memory, and 3.6 GHz speed. For a fair comparison, we reproduce the computational time of SEFRON on our hardware and calculate the performance ratio of the computational time for SEFRON between our hardware and the one used in [30,48]. Then, we use the SEFRON as an intermediary to estimate computational time for other methods on our hardware through the performance ratio. It is important to note that the training times reported in Table 4 exclude hyperparameter tuning for all methods, including the multi-start initialisation and GA optimisation used in our approach. This ensures a fair comparison, as the hyperparameter tuning times for the other methods are not explicitly reported in their respective work.

It is shown in Table 4 that the SpikeProp, SWAT, and TMM-SNN require many hidden neurons to achieve a testing accuracy above 85% on Ionosphere, Breast cancer, and Iris benchmarks, whereas our method achieves comparable performance with fewer neurons. Consequently, the larger SNN architectures of those methods result in significantly higher computational time used per training epoch compared to our method. In the case of Ionosphere dataset, the SpikeProp algorithm achieves 89% training accuracy and 86.5% testing accuracy using 25 hidden nodes and 3000 epochs, taking 27.3 s per epoch. In contrast, our methodology achieves a significantly higher training accuracy of 99.5% and testing accuracy of 94.3%, using no hidden nodes, significantly fewer epochs, and lower computational time per epoch. For the difficult binary classification problem of Liver disorders, our method requires fewer epochs, fewer neurons, and less computational time to achieve similar accuracy to the TMM-SNN. This indicates that the time-varying weight SNN, trained using our proposed methodology, achieves performance comparable to that of an SNN with a larger network architecture while requiring less computational time. Notably, when the GA is not utilised, there is a noticeable decline in performance. This underscores the importance of our proposed global optimisation approach in determining the network parameters and highlights its significant contribution to the effectiveness of our method.

Compared with the WOLIF in which its network is designed with constant weights and without hidden layers, both of our method configurations (with and without the GA) use a lower number of epochs to achieve higher accuracy for Ionosphere, Breast cancer and Iris benchmarks. With the global optimisation using GA, our method demonstrates approximately 2% improved accuracy on average from the WOLIF. However, the WOLIF demonstrates the highest accuracy in Liver disorders dataset among the methods. The training time for the WOLIF is not available and, therefore, is not included in the table.

Compared to SEFRON on benchmarks for Liver disorders, Breast cancer, Ionosphere, our proposed SNN-based methodology achieves higher classification accuracy regardless of the use of a GA. With the same number of epochs and a similar architecture, our method also consumes less training time per epoch. While (23) and (24) introduce a learning rule inspired by STDP, the gradient-based weight update in our method, detailed in (17)–(22), directly minimises the error between predicted and desired spike times. This distinction from SEFRON contributes to the improved classification accuracy with lower computational cost. Additionally, when our method employs global optimisation using GA, it achieves an average accuracy improvement of approximately 5% over SEFRON. Note that the SEFRON cannot directly be tested using the Iris benchmark as the network is applicable only for binary classification due to the use of a single output neuron, whereas we employed more output neurons.

The Mc-SEFRON is an extension of SEFRON designed to handle multi-class classification through modified training algorithms and population encoding schemes. While our proposed methodology demonstrates higher classification accuracy with less computational efficiency across several benchmarks, the Mc-SEFRON outperforms in specific aspects. Notably, it attains higher average accuracy on the Iris dataset and demonstrates superior computational efficiency, being approximately 20% faster per epoch compared to our method.

Notably, the use of GA in our methodology demonstrates performance improvement by 2.35% on average. However, this does not imply that the proposed SNN is overly dependent on hyperparameters or unsuitable for online inference. The GA is a one-time process and is used only during training to optimise hyperparameters. Once these parameters are determined, the model operates efficiently with fixed parameters. However, future research could explore adaptive techniques, such as incremental learning or online adaptation, to adjust to new data without full retraining, enhancing long-term performance.

7. Spiking neural network-based methodology with squat detection

With its promising performance on classification, this section presents the applicability of an SNN with time-varying weights having no hidden layers and our proposed methodology to solve a complex spatiotemporal problem presented in squat detection.

7.1. Measurements

Data used in this paper were obtained from multiple ABA measurements mounted on a measuring train in a track section measured at almost constant speed. The sampling frequency of the ABA measurements was 25.6 kHz. To alleviate potential issues that might arise from speed variation, although slightly in this work, we preprocess the data by first transforming the time-domain signals into frequency responses using a fixed number of time-domain data points. These frequency responses are then resampled to provide a fixed number of inputs to the SNN. For future research, spatial frequency analysis could be explored to mitigate the effects of speed variations on the number of samples. However, nonlinearities introduced by higher velocities and dynamic impacts remain an open challenge. New AI methods capable of adapting to varying speeds and segment lengths could be investigated.

The ABA measurements and field validation campaigns were conducted in (1) the track between Zwolle and Meppel, the Netherlands and (2) the track between Luleå and Narvik, Sweden. The information was acquired from accelerometers installed in both longitudinal and vertical directions and from the left and the right wheels of the leading and trailing wheelsets [9]. We repeated the measurements and both are used in the analysis. The measurements captured responses from healthy rails, welds, insulated joints, switches, and squats (light and severe). Our measurements are labelled according to observations and fieldwork. The locations of insulated joints, welds, and switches are known. The responses of insulated joints and switches are removed from the analysis as their locations are known and their degradation mechanisms have particular characteristics that are possible to analyse. As squats can initiate from welds, our analysis includes the responses of welds for both welds with squats and welds without visible squats.

7.2. Implementation details

From a practical perspective aimed at enabling timely and actionable maintenance decisions, the choice of segment length in our analysis is driven by the need to accurately detect and localise local defects. Longer segments (e.g., 1 km or 10 km) tend to aggregate multiple defect types and locations, resulting in ambiguous outputs where a defect is flagged but its precise location remains unknown. This significantly reduces the usefulness of such results for maintenance planning. In contrast, shorter segments enable more accurate localisation and prioritisation. Moreover, the availability of validated labels obtained via detailed manual inspections is inherently limited to only a few kilometres per day. Therefore, segmenting the data into smaller lengths increases the number of labelled samples and improves the generalisation of model training. Therefore, this work focuses on the adoption of shorter segments (less than 1 metre) to support both technical and practical needs of infrastructure monitoring.

A sensitivity analysis on the length of the ABA signals used is performed for both measurements. The process begins with an analysis of defect characteristics, utilising field data or expert knowledge to estimate the typical response durations associated with defect types and severity levels. Once the response duration is estimated, the signal is segmented into varying durations. Each segment is then evaluated to assess its effectiveness in capturing the relevant characteristics of the defects. This evaluation identifies the segment duration that provides the most accurate representation of the defect characteristics, which is subsequently selected for further analysis. For the measurements from the Netherlands, the analysis suggests that using the ABA signals covering a distance of 78 cm yields the best predictive performance in terms of detection performance for the particular measurements of light squats considered in this case study. This performance is obtained with the sliding window of length equals to the distance of 8 and 4 centimetres for the vertical and longitudinal ABA signals, respectively. For the measurements from Sweden, the analysis suggests using the ABA signals covering a distance of 100 cm with the sliding window of length equal to the distance of 10 and 5 centimetres for the vertical and longitudinal ABA signals, respectively.

For each case study, the SNN is trained to classify whether or not a given rail segment contains squats. Therefore, our problem consists of a class of non-defective rails, referred to as Class 1, and a class of rails with squats, referred to as Class 2. For Class 1, the samples include rails at welds and healthy rails, whereas Class 2 includes rail samples with light and severe squats. For the Dutch case study, the dataset contains a total of 944 rail samples, of which 824 samples are labelled as non-defective rails and 120 samples are labelled as rails with squats. For the Swedish case study, the dataset contains a total of 1222 rail samples, of which 1068 samples are labelled as non-defective rails and 154 samples are labelled as rails with squats. In railway applications, labelled data are limited due to the complexity of the labelling process. For instance, the correct label of one defect, particularly for early-stage defects, requires an analysis of locations and a confirmation from multiple data sources because of their subtle development and these early-stage defects are frequently overlooked by human observers or video cameras. Additionally, railway tracks involve hundreds of kilometres, and fieldwork has limited access. Moreover, diverse local infrastructure conditions, track dynamics, and stochastic variables, e.g., track misalignments or roughness, train speed, axle load, wheel conditions, environmental conditions, measurement noise, and the hunting effect, make the labelling process more difficult. As a result, assigning class labels is a laborious and time-consuming process. For the model training, each of the datasets is divided into the training and test sets with the ratio of 70:30, and 10-fold cross-validation is performed, in which 90% of the training set is used to train the SNN and the other 10% are used for validating the trained model. The trained model is then tested with the holdout test set to evaluate its performance for squat detection.

By working with short segments of less than 1 metre, the system can process each segment in under one minute, including feature extraction and SNN-based classification, using a standard laptop equipped with an Intel i9 CPU, 128 GB RAM, and an NVIDIA RTX 3080 GPU. This capability enables near real-time operation, allowing defect alerts to be generated shortly after the train passes a given track section. When more complex models are required, processing can be performed offline using high-performance computing resources. This supports both onboard deployment for real-time monitoring and office-based post-processing for advanced analytics.

7.3. Imbalanced dataset

Conventional supervised methods rely extensively on balanced datasets between classes to guarantee their performance. In some railway infrastructures, it is extremely difficult to obtain a wide variety of class information for defects. That is the case when the inframanager conducts preventive grinding campaigns. In the cases reported in this paper, the dataset used for training our model is imbalanced; that is, examples of healthy rails are abundant. Moreover, not only defects but healthy data are also difficult to label. This is because of different rail behaviours at different locations, the time-varying characteristic of railway infrastructure conditions and various different stochastic variables.

To deal with the situation of imbalanced data, clustering can be exploited to construct the balanced data set by reducing the number of the majority classes [49] or oversampling the minority classes [50]. Transfer learning is another approach that researchers use to transfer learned parameters from a domain of balanced datasets to another one whose datasets are imbalanced [51]. To show the effect of the imbalanced dataset on the performance of our method, a sensitivity analysis is done by considering different percentages of faulty data in the training set. By reducing the number of the healthy class, we consider five different percentages of faulty samples: (1) 13%, (2) 17%, (3) 25%, (4) 50%, and (5) 67%. To obtain model performance, we follow the feature engineering described in Section 4 and obtain the 54 frequency-based representations of the measured ABAs at rails. After encoding these features into spikes, the SNN architecture for this problem constitutes $54 \times 6 = 324$ input nodes and 2 output nodes. Following our SNN-based methodology (see Fig. 6), Fig. 10 reports an averaged classification performance obtained from the training set of each case. It is observed that detection performance for defects improves when a higher percentage of defective samples is included in the training set. However, for this experiment, total classification accuracy and detection performance for healthy samples degrade as the model is trained with an insufficient amount of data.

7.4. Results

Firstly, the capability of our SNN-based methodology to solve the problem of squat detection is compared with other methods using ABA measurements from the Netherlands in terms of detection performance and computational cost. Then, we show its applicability in a railway line from Sweden. For both case studies, we follow the datasets described in Section 7.2.

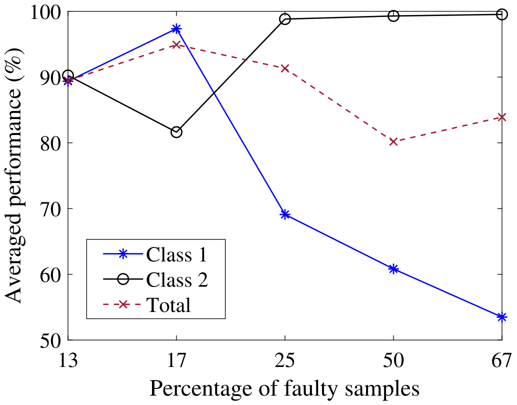


Fig. 10. Effect of the percentage of faulty samples in the training set.

7.4.1. Evaluation metrics

The effectiveness of the proposed methodology is assessed based on the following metrics:

$$\text{Specificity} = \frac{TN}{TN + FP}, \tag{26}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{27}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{28}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{29}$$

$$\text{FA} = \frac{FP}{TP + FP}, \tag{30}$$

where true positives (TP) are the number of correctly classified defects, false positives (FP) are the number of normal samples classified as defects, and False Negatives (FN) are the number of misclassified defects. Specificity is a metric for the detection performance of non-defective rails (Class 1), while Recall is a metric for the detection performance of defective rails (Class 2). FA denotes false alarm or false positive rate.

7.4.2. Classification performance and computational cost

Following our SNN-based methodology (see Fig. 6), the optimal set of hyper-parameter values for the Dutch railway line is: $T = 3$, $T_{\text{sim}} = 4$, $\hat{t}^d = 2.46$, $\sigma = 0.21$, $\eta = 0.000248$, and $\tau = 3.44$. By setting the hyper-parameters as described, the experimental results show that our SNN-based methodology yields an accuracy of 93.12% for non-defective rails (Specificity) and 97.14% for squat detection (Recall). The methodology can detect 100% of severe squats and 95.83% of light squats. It achieves the highest F1-score at 79.07% with an FA of 33.33%. Fig. 11(a) shows an example of predictions for the Dutch railway line. The length showcased here is 1.03 km, with 30 locations of squats reported. Our method predicts 42 locations of squats, of which 29 locations are correct, one squat location is missed, and 13 locations are false alarms. The field validation has indicated that these false alarms are at welds. With the high magnitude of ABA responses, the method classified them as defects. Further research is needed to differentiate welds from defects directly from ABA responses. Still, the location of welds can be registered in advance by inframanagers and verified with other measurement sources. Thus, welds can be excluded from the analysis of the detection of isolated defects. An example of a false alarm is given and discussed in Section 7.6.

In order to justify its applicability of squat detection, the performance of our method is compared with a wavelet-based method [2], support vector machine (SVM), and neural network-based methods. SVM is included as it has a simple architecture. For the NN-based methods, Gated Recurrent Unit (GRU) and artificial NN (ANN) are

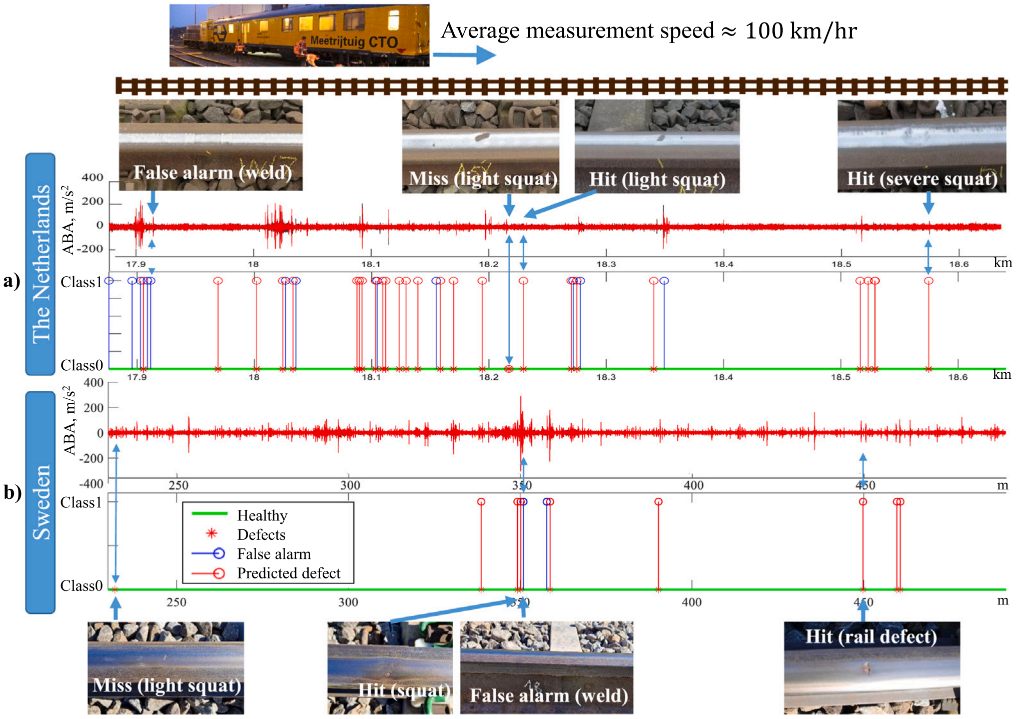


Fig. 11. Detection of rail squats.

Table 5
Comparative results of our SNN-based methodology with other methods using the ABA measurements from the Netherlands.

Method	Architecture	Detection performance (%)				Precision	F1-score	FA
		Class 1 (Spec.)	Class 2 (Recall)	Light squats	Severe squats	(%)	(%)	(%)
- Wavelet-based method [2]	–	87.87	85.71	79.17	100.0	50.0	63.16	50.0
- SVM with non-linear RBF kernel	–	98.38	60.00	45.83	90.91	84.0	70.0	16.00
- GRU	54:150:2	85.00	68.57	58.33	90.91	80.0	73.85	20.00
- ANN	54:450:2	77.50	68.57	58.33	90.91	72.72	70.59	27.27
- SEFRON [30]	324:1	95.14	91.43	87.50	100.0	69.23	72.97	30.77
- SNN with constant weights	324:2	93.12	88.57	83.33	100.0	64.58	74.69	35.42
- SNN with time-varying weights	324:2	93.12	97.14	95.83	100.0	66.67	79.07	33.33

selected. Although more layers can be used to increase the predictive performance of the models, GRU and ANN are designed with a single hidden layer to showcase a comparison with our SNN which contains no hidden layers. To evaluate the performance of the SNN with time-varying weights and no hidden layer, a comparison with the SEFRON [30] and a constant weights version of our method is also presented.

For the wavelet-based method presented in [2], the scale-averaged wavelet power (SAWP) was proposed to detect squats in which an empirical constant C related to the mother wavelet and a constant threshold are chosen to maximise the hit rate and reduce the number of false alarms. In this paper, the constant C is optimised and set to 35 and the same threshold of 0.5 m^2/s^4 is used to detect squats. For the other methods, the same implementation details described in Section 7.2 are followed and sensitivity analysis of hidden nodes used is performed.

Table 5 presents the comparative results of a squat detection obtained from the selected methods. It can be seen that all the methods achieve reasonable performance for non-defective rails (Class 1). For the detection performance of squats (Class 2), particularly light squats, SNN-based approaches, including SEFRON, SNN with constant weights, and SNN with time-varying weights, demonstrate higher accuracy. Among these, our SNN-based method achieves the highest detection performance for squats, particularly light squats. This benefit comes from its ability to produce time-encoded outputs that resemble the continuous scanning of ABA measurements.

The wavelet-based method can detect approximately 80.0% of light squats. This is consistent with the results reported in [2,9] in which light squats can be detected using ABA measurements with accuracy between 78% to 85%. The SVM and NN-based methods using a single hidden layer demonstrate similar detection performance for light squats. Their accuracy is less than 60.0%. Among all the methods, the SVM shows the lowest percentage of false alarms. Even though our SNN-based method produces more false alarms than the SVM, our SNN-based methodology has shown an ability to capture subtle changes in the responses of light squats in ABA signals using simple network architecture. Additionally, our method achieves a significantly higher F1-score, indicating a superior balance between precision and recall. Compared with the SEFRON, an SNN with the time-varying weights trained by our methodology achieves 8.3% higher accuracy for the detection of light squats but produces 2.6% more false alarms. Moreover, the comparison with the constant weights demonstrates that SNN with the time-varying weights trained by our methodology

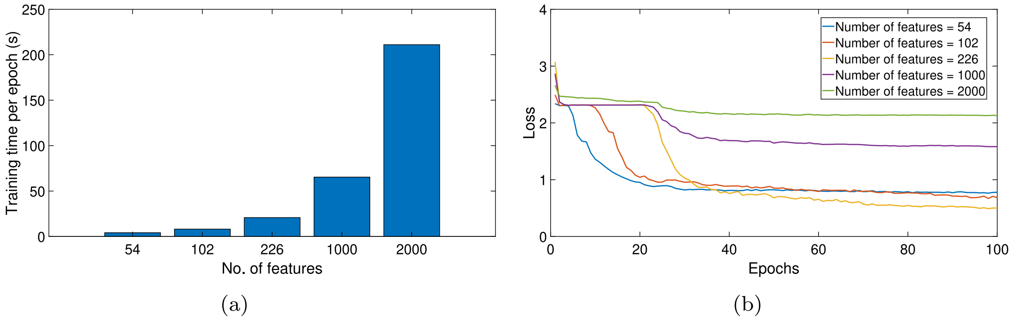


Fig. 12. Effect of data dimensions on the performance of our SNN-based method in terms of (a) convergence rate over 100 epochs and (b) computational time. This analysis identifies using 54, 102, and 226 features as Pareto-optimal solutions, demonstrating a trade-off between loss and computational time.

Table 6
Comparison of computational time obtained from our SNN-based methodology and other machine learning methods using the ABA measurements from the Netherlands.

Method	Architecture	Avg. time per epoch (s)
SVM with non-linear RBF kernel	–	25.2488
GRU	54:150:2	145.6138
ANN	54:450:2	34.0459
SEFRON [30]	324:1	4.2244
SNN with time-varying weights	324:2	1.9222

has improved accuracy in solving the complex spatiotemporal pattern presented in the light squat detection problem.

Table 6 presents a comparison of the computational time, measured as the average time taken to complete one epoch, for our SNN-based methodology and other methods using ABA measurements from the Netherlands. Our method outperforms the others in terms of computational efficiency. Despite having a single hidden layer, the NN-based methods, i.e., GRU and ANN, require the most time per epoch due to the complexity of their network structures. Among the simpler architectures, SVM consumes more computational time than both SEFRON and our method. The SNN-based methods require less computational time, highlighting their efficiency by utilising temporal information instead of numeric values. Notably, our method is faster than SEFRON, underscoring the effectiveness of using back-propagation over the STDP learning approach.

For the measurements from Sweden, the optimal set of hyper-parameters used in its SNN model is: $T = 3$, $T_{\text{sim}} = 4$, $\hat{t}^d = 0.4$, $\sigma = 0.1$, $\eta = 0.00085$, and $\tau = 3.76$. By setting the hyper-parameters as described, our SNN-based methodology can be used to detect squats in a railway line from Sweden with an accuracy of 95.65% with 8.33% false alarm rate. Fig. 11(b) illustrates the selected portion from the Swedish railway line with a length of 0.3 km, where 10 locations of squats are reported. The model has detected 12 locations of squats, of which 10 locations are correct, one location is missed, and 2 locations show as false alarms. Similar to the performance obtained for the Netherlands results, the Swedish railway line data demonstrate the applicability of our SNN-based methodology for rail squat detection. Similar to the measurements from the Netherlands, the locations of false alarms are at welds that are not in a good condition.

7.5. Scalability

This section evaluates the scalability of the proposed SNN-based methodology as data complexity increases. The railway data from the Netherlands is exploited and the data complexity is reflected through increased dimensionality and utilisation of training data from both the time and time–frequency domains. Detecting rail squats from high-dimensional time-domain ABA signals can be challenging due to significant variability in dynamic behaviours at different track locations, which are affected by local properties of rail infrastructures and changing operational conditions of trains. Incorporating a longer signal length includes more variability of dynamic behaviours, complicating the analysis. Consequently, time-domain analysis can be more sensitive to variations, which may obscure important patterns related to rail squat dynamics. Therefore, the effect of increased data complexity on performance is investigated through five cases with different feature dimensions. The first three cases address data complexity in the frequency domain by selecting three different frequency ranges to describe ABA signals: 200–2000 Hz resulting in 54 features, 50–4000 Hz resulting in 102 features, and the full frequency range resulting in 226 features. These frequency-based features are obtained as described in Section 4. The last two cases focus on the temporal aspect reflected by different rail segment lengths. In these cases, we simulate scalability by increasing the length of the time series corresponding to feature dimensions of 1000 and 2000. The simulations are repeated 10 times, and the performance of the method is evaluated in terms of convergence and computational efficiency across varying complexities, with cross-entropy loss employed as the evaluation metric. The reported values represent the average results from these ten repetitions.

Fig. 12 illustrates that both the computational time and the convergence rate of the proposed SNN-based methodology are affected as the dimensionality of the data increases. The training time per epoch for 54, 102, and 226 features remains relatively low, with a slight increase as the number of features rises from 54 to 226, as shown in Fig. 12(a). For 54 and 102 features, the convergence is relatively fast, with loss decreasing rapidly within the first 20 epochs and stabilising thereafter, as seen in Fig. 12(b). While the convergence rate slows slightly when the number of features increases to 226, the loss is significantly reduced compared to using 54 features. This demonstrates the method's ability to handle moderate increases in dimensionality without notable degradation in performance. Notably, using 226 features comes at a higher computational cost, as shown in the increased training time per epoch. As a result, the analysis identified using 54, 102, and 226 features as Pareto-optimal solutions, demonstrating a trade-off between loss and computational time. To achieve better performance (lower loss), an increase in computational time is required. Additionally, the analysis does not claim to identify the global optimal number of features. Instead, it highlights the practical challenges in balancing performance and computational efficiency. This trade-off underscores the importance of feature selection in real-world applications, where computational constraints must be considered alongside detection performance.

In contrast, when the feature dimension is significantly increased to 1000 and 2000, the convergence rate is notably slower, resulting in higher loss values. Moreover, the training time per epoch escalates dramatically, as depicted in Fig. 12(a). This indicates that while the method can still learn from high-dimensional data, the increased complexity from using time-domain features impairs its efficiency and effectiveness.

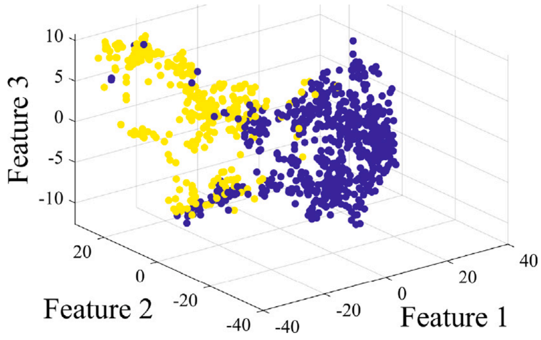


Fig. 13. The three-dimensional t-SNE embedding of the representative features for our SNN-based methodology. The class of non-defective rails is illustrated in blue and the class of rails with squats is illustrated in yellow.

7.6. *Explainability for representation learning*

Researchers have been working on developing models with a good balance between explainability and accuracy [52]. To achieve the explainability of squat detection, wavelet analysis is typically employed due to its direct physical interpretation. However, due to multiple sources of variability in the ABA responses, its accuracy is inferior to NN-based models (see Section 7.4). On the other hand, even though using NN-based models can achieve high accuracy, they are black-boxes and their prediction under new measurement conditions can be unreliable and not explainable.

To address the explainability of squat detection using SNN, we investigate (1) how well the SNN learns the data representations, (2) how the decision of the SNN is obtained, and (3) how the representative features learned by our methodology are favourable for early squat detection. Measurements from the Dutch railway are used as a showcase.

7.6.1. *Visualisation of our features using t-SNE*

For demonstration purposes, we perform a t-Distributed Stochastic Neighbour Embedding (t-SNE) of the representative features in three dimensions using the Euclidean distance metric. Fig. 13 shows the cluster assignments obtained with our SNN-based methodology. The visualisation demonstrates a minimal overlap between the class of non-defective rails (blue) and the class of rails with squats (yellow) in the three represented dimensions. As features that are close in the high-dimensional feature space are also close in the three t-SNE represented dimensions, it infers that a reasonably good separation of clusters is given by our method. This supports that the SNN learns well the represented data.

7.6.2. *Visualisation of representation learning*

Fig. 14 illustrates the time-varying synaptic weights from the input layer to the output layer after training for squat detection. Figs. 14(a) and 14(b) show the results obtained using our method for Class 1 and Class 2, respectively, while Figs. 14(c) and 14(d) present the corresponding results using SEFRON. As seen in Figs. 14(a) and 14(b), the synaptic weights in our method exhibit substantial variation over time. Notably, the values of the synaptic weights inside box A fluctuate as time varies. In box B, the synaptic weights for Class 1 switch in sign from negative to positive. The same behaviours are also observed for Class 2 but with small variation. In contrast, the synaptic weights produced by SEFRON show minimal variation across time as observed in Figs. 14(c) and 14(d).

The key characteristic that allows the variations in the synaptic weights of both our method and SEFRON is the use of an exponential decay function, as defined in (23). The exponential function plays an important role in controlling the weight variations, $\Delta w_{ij}^{k,e}\left(t_i^k\right)$, with the parameter σ determining the range of influence over time. This dynamic adjustment allows the method to effectively track changes in synaptic weights within a specific time window. Consequently, the variations in synaptic weights facilitate both excitatory and inhibitory postsynaptic potentials depending on the timing of presynaptic spikes t_i^k. However, the key distinction lies in how $\Delta w_{ij}^{k,e}(t_i^k)$ is obtained. Our method employs a gradient-based learning rule, as defined in (17), which allows for global error-driven updates across time. This leads to more adaptive temporal dynamics in the synaptic weights, as evidenced by the pronounced fluctuations. In contrast, SEFRON applies a local spike-timing-dependent plasticity (STDP) rule, where updates are based solely on the relative timing of individual pre- and postsynaptic spikes. This localised mechanism results in more static weight patterns, contributing to its lower performance relative to our method, as demonstrated in Table 5.

To provide information about which input features have contributed to the respective decision of the SNN-based methodology, the postsynaptic potentials for Class 1 and Class 2 are evaluated over the simulation time using the time-varying synaptic weights of the corresponding class and the spike response in order to determine the membrane potentials of the corresponding class. In the following, we illustrate how different representations of rail dynamics affected the performance of the detection of squats. The colour mapping of the postsynaptic potentials is symmetric around 0 (green), from dark-blue (strong inhibitory) to dark-red (strong excitatory). The change in sign from positive to negative reveals that our SNN can capture both excitatory and inhibitory states within a single synapse.

Figs. 15 and 16 demonstrate a correct detection of a weld and a light squat by our SNN-based methodology. It is observed in Figs. 15 (see D and F) and 16 (see H and J) that, for both Class 1 and Class 2, the excitatory postsynaptic potentials or positively activated parts of the SNN (red) increase towards the end of the simulation time. However, the excitatory postsynaptic potentials of Class 2 increase relatively slower than Class 1.

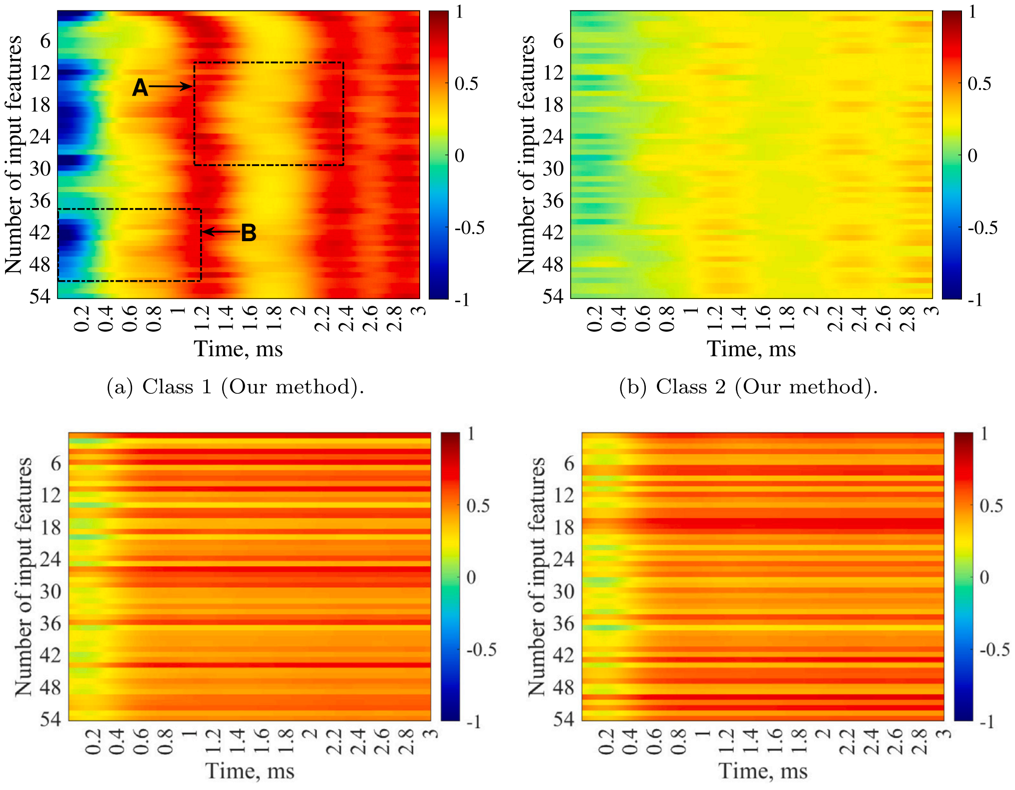
To determine the membrane potential of each class at a time instance, the excitatory postsynaptic potentials are compensated with the inhibitory postsynaptic potentials. If the measured ABA signals are at squats, their corresponding membrane potential of Class 2 has to cross the threshold (represented by the dotted red line) and emit the first spike. This infers that the membrane potential of Class 1 has to be weaker positively activated (lower amplitude) than that of Class 2 at the output spike time.

From our observations, the inhibitory postsynaptic potentials or deactivated parts of the SNN (blue) appear to influence the decision. It is shown that the inhibitory postsynaptic potentials in Fig. 16 (see I) are more pronounced and remain deactivated towards the end of the simulation time in Class 1, compared to those in Fig. 15 (see E). The final membrane potential of Class 1 in Fig. 16 is then weaker positively activated than that in Fig. 15. Therefore, the measured ABAs in Fig. 16 are classified as a rail at squats.

Fig. 17 provides an analysis of rail responses measured by ABAs at a weld that are misclassified by our SNN-based methodology. It is observed that the inhibitory postsynaptic potentials are more pronounced towards the end of simulation time in Class 1 as shown in Fig. 17 (see L). The wrong behaviour of inhibitory postsynaptic potentials in Class 1 is a possible reason that leads to misclassification.

7.6.3. *Explanation of the representative features*

Figs. 15 and 16 show that the spike response of the encoded input features from different classes are different. The response of all input spikes from the ABA signals of non-defective rails increases uniformly over the simulation time, as illustrated in Fig. 15 (see C). In contrast, Fig. 16 (see G) shows that there are input spikes from the measured ABA signals at squats whose response increases dominantly. These input spikes are representative features from both vertical and longitudinal ABA signals that are associated with the high-frequency



(a) Class 1 (Our method).

(b) Class 2 (Our method).

(c) Class 1 (SEFRON).

(d) Class 2 (SEFRON).

Fig. 14. Visualisation of the time-varying synaptic weights from the input layer to the output layer after training for squat detection (a)–(b) using our method and (c)–(d) using SEFRON.

band between 1000–2000 Hz. According to [9], this frequency band is a frequency characteristic of light and severe squats. When making a decision, the features from this band are informative for evaluating squats. Fig. 17 (see K) assures the finding as the wrong behaviour of the spike responses leads to misclassification. This infers that our representative features are favourable for early squat detection as the variation of energy of light squats in the high-frequency band is more pronounced using an SNN.

7.7. Discussion

The proposed SNN-based methodology achieves an improvement in the detection performance of light squats. However, it still suffers from misclassification from the measured ABA signals at non-defective rails into the class of rails with squats. At these non-defective rails with false detection, it is observed that our method provides consistent squat predictions in signals with different measurements. For instance, squats are predicted by our method in the signals of all wheels from the left and the right rails with the first measurement, whereas squats are predicted in the signals of all wheels from the left rail and in the signals of one out of two wheels from the right rail with the second measurement. This suggests that there might be some invisible rail defects located at these healthy rails. Further experiments can be conducted at those rails where a false detection is shown.

Further improvement on the false negative and positive rates is needed because they have an impact on maintenance costs. New methods that can support reducing the uncertainties coming from the labelling process are needed. As the presence of invisible defects is highly affected by the labelling process, having historical information allows investigating the complete time evolution of a defect, from the moment it was first detected by the system until the latest stage of its evolution. This can be obtained using the advancement of ABA measurements in which accelerometers can be mounted on operational trains and multiple measurements can be easily conducted over the same track and over different time periods.

Moreover, our SNN-based methodology has exhibited difficulty in differentiating between the ABA responses of welds and squats. An underlying reason for the misclassification is that a weld and a squat can be located in close proximity in certain rail sections. Consequently, the rail response of squats influences the measured ABAs at welds. Additionally, welds can be healthy (with ABA signals resembling healthy rail) or suffer from degradation (with ABA signals resembling local defects). This poses a challenge for our method to distinguish by using solely ABA measurements. Still, the location of welds is known, and they are visible, so they can be obtained and verified with measurement campaigns and technology such as video images. As monitoring data of rail health conditions are also available in other formats, e.g. images, our future research will use these monitoring data as well. Therefore, the fusion of information obtained from different data types will be studied to provide not only more accurate early detection of rail surface defects but also fewer false alarms.

Even though the frequency characteristics of squats in ABA signals are consistent and typically appear in a frequency band of 200–2000 Hz for the train speed of 100 km/h, making the method potentially applicable across different railway systems, variations in environmental conditions, infrastructure layouts, and operational characteristics can affect signal prominence. For example, lower loads and speeds produce less pronounced responses. To ensure generalisation, careful adaptation is required. This includes preprocessing and filtering to denoise and enhance signal prominence, and adjusting model parameters with new data to minimise false alarms and ensure accurate detection across different systems.

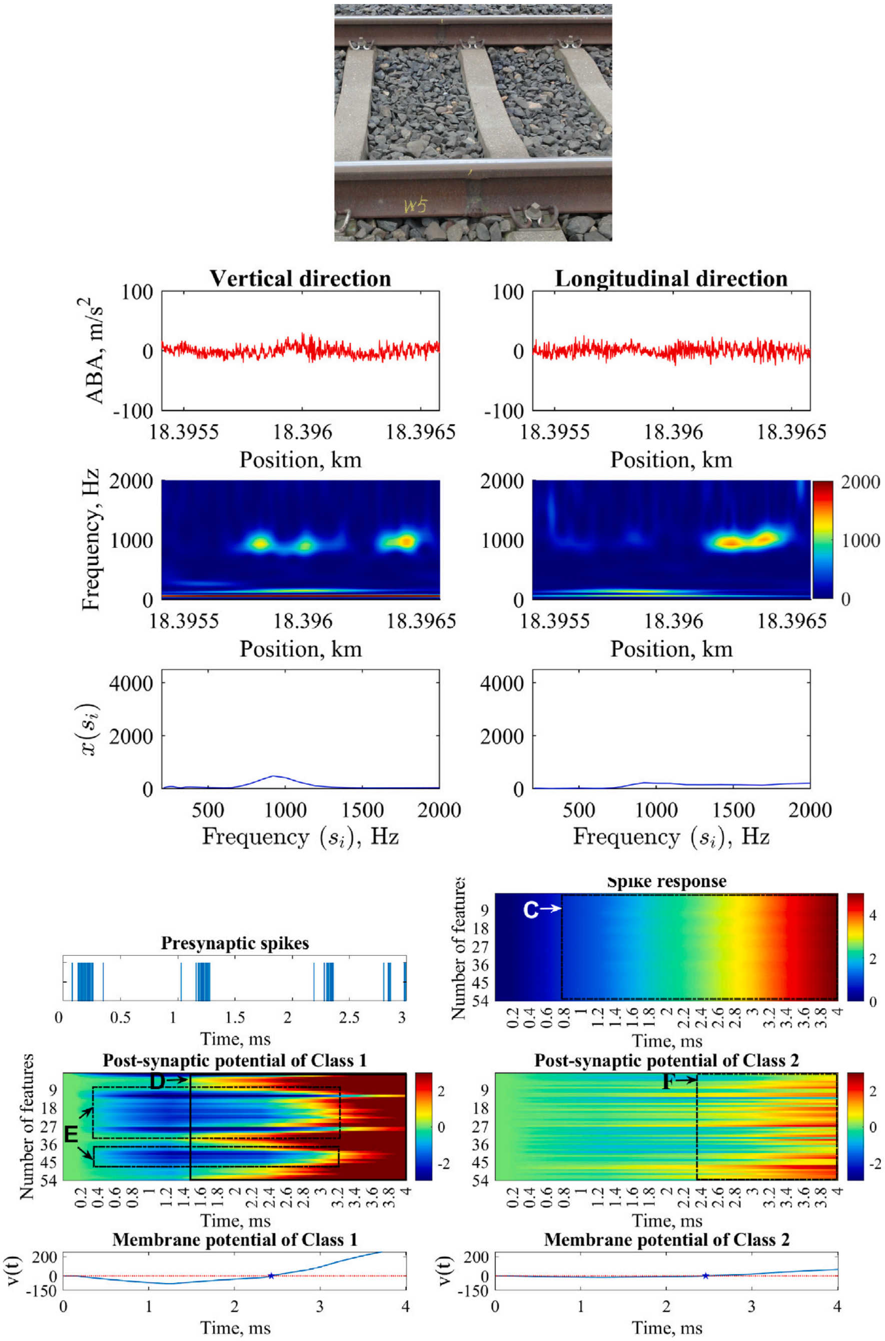


Fig. 15. Visualisation of correctly classified ABA responses at a weld by the SNN-based methodology. In the membrane potential graphs, the blue line represents the membrane potential $v(t)$. The red dotted line indicates the threshold for neuron firing. When the membrane potential crosses this threshold, the neuron fires a postsynaptic spike, which is marked by blue star symbols on the graph.

8. Conclusions

This paper proposes a spiking neural network with time-varying weights using no hidden layers and its training methodology based on a gradient-based approach. Testing on four UCI benchmarks, the accuracy obtained from our SNN-based methodology is competitive to other state-of-the-art SNNs when dealing with nonlinear classification problems for not only binary but also multiple classes. Using field measurements from the Netherlands and Sweden, the results from a squat detection have shown that our method is applicable of solving a complex spatio-temporal problem in railway. We have improved the detection performance of light squats from the traditional methods which is of 78%–85% to more than 93%. Furthermore, our method can be used to provide interpretability. The spike responses, postsynaptic potentials, and membrane potentials have offered an explainable way to analyse the ABA signals. These internal spike behaviours highlight a correspondence with high frequency band between 1000–2000 Hz of the detection problem of squats and offer an ability to capture subtle changes in the responses. Despite its success, the effectiveness of our SNN-based methodology can be affected by factors such as class imbalance, noise levels, and uncertainty in distinguishing subtle behaviours, such as rail dynamics. Future research can consider enhancing the robustness and effectiveness of the SNN-based methodology in handling

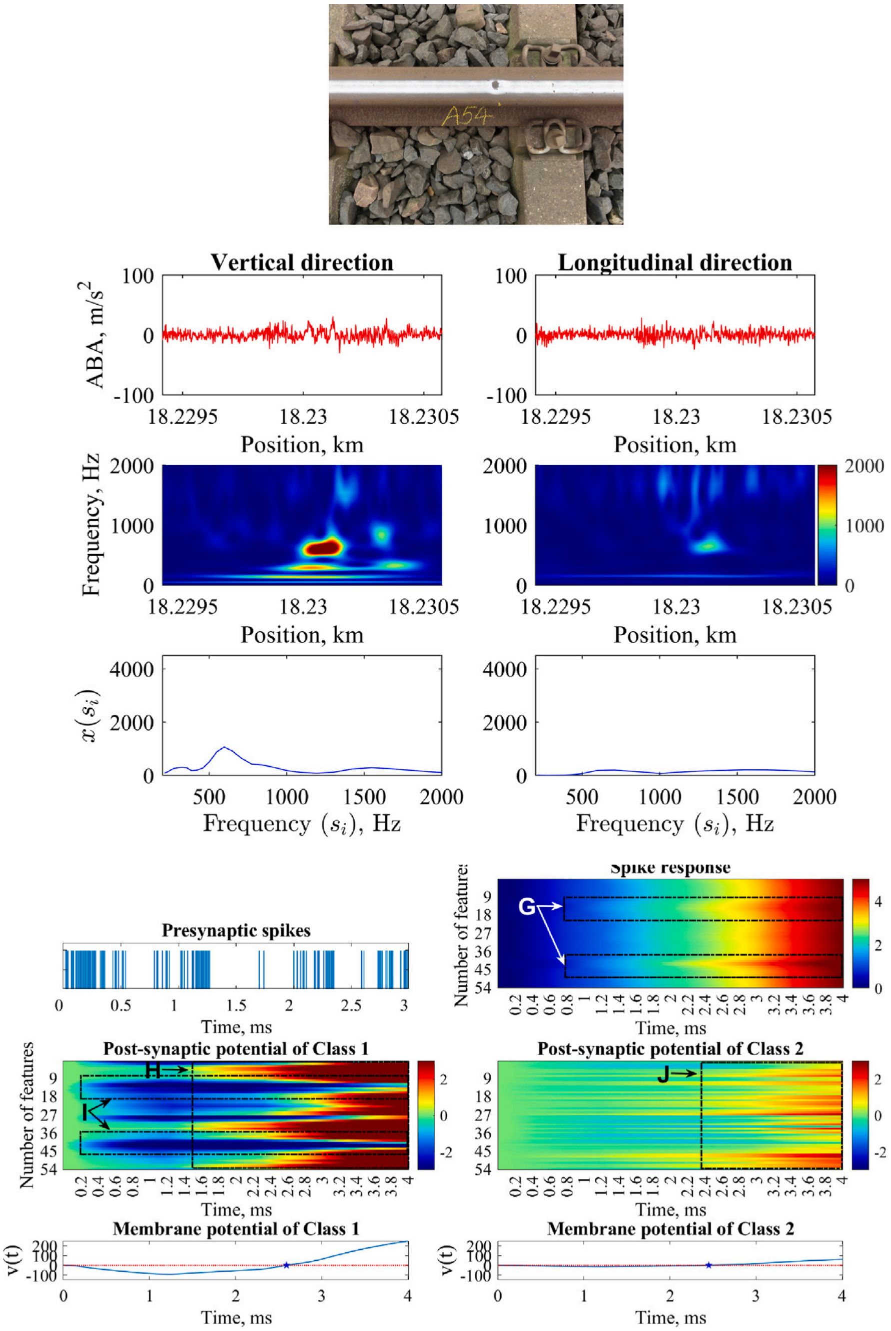


Fig. 16. Visualisation of correctly classified ABA responses at a light squat by the SNN-based methodology. In the membrane potential graphs, the blue line represents the membrane potential $v(t)$. The red dotted line indicates the threshold for neuron firing. When the membrane potential crosses this threshold, the neuron fires a postsynaptic spike, which is marked by blue star symbols on the graph.

these data characteristics by explicitly including possible stochasticity in the objective function and the robust parameters in the design.

Compared to current state-of-the-art methods, the SNN-based approach excels in situations with limited training data and complex spatiotemporal problems. These advantages arise because SNNs process data as discrete events (spikes) and capture information through spike timing patterns. Additionally, the proposed SNN incorporates time-varying weights to dynamically adjust synaptic connections based on encoded spatiotemporal patterns derived from ABA signals. While the input features are in the frequency domain, the temporal information is indirectly represented through variations in feature magnitudes and patterns. This allows the time-varying weights of SNN to capture and process these event-based data representations, which are essential for identifying subtle patterns indicative of rail squats. The use of a global optimisation approach for determining network parameters further enhances the performance of our method as the approach helps in mitigating overfitting and avoiding local minima. However, the proposed SNN-based methodology has limitations. The SNN is not explicitly designed to address class imbalance, noise, and uncertainties. However, it is interesting for further research to analyse how explicitly the SNN can tackle these issues and evaluate its performance compared to the existing methods, such as generative-based, probabilistic, and transfer learning methods. Understanding these limitations is essential for optimising the application of the proposed method and improving its performance and robustness across various datasets and conditions. Additionally, a hybrid approach that combines those state-of-the-art

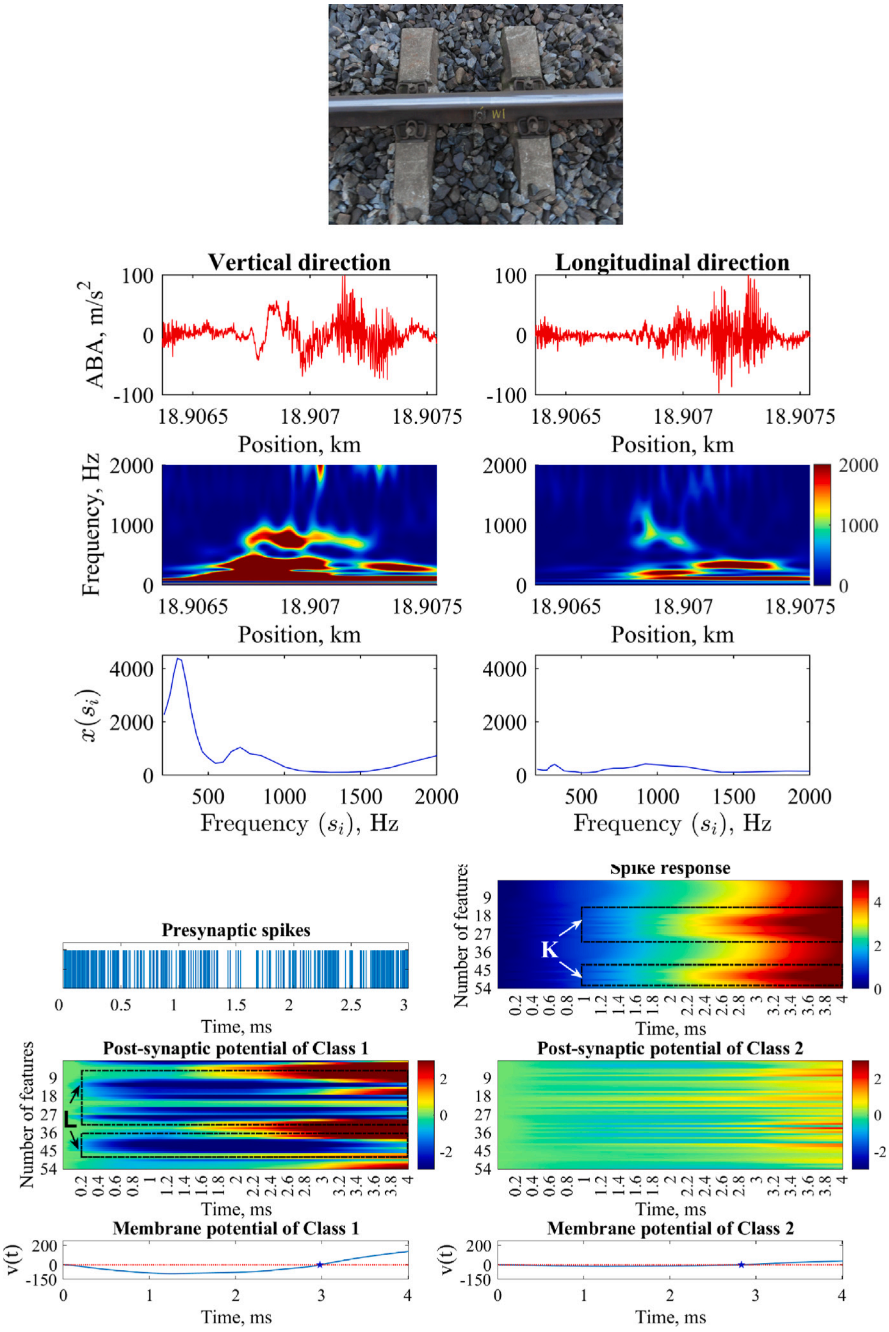


Fig. 17. Visualisation of misclassified ABA responses at a weld by the SNN-based methodology. In the membrane potential graphs, the blue line represents the membrane potential $v(t)$. The red dotted line indicates the threshold for neuron firing. When the membrane potential crosses this threshold, the neuron fires a postsynaptic spike, which is marked by blue star symbols on the graph.

methods with SNNs could further enhance performance in jointly tackling issues presented in the problems.

Future work will explore extending the method to multi-layer SNNs by developing new training mechanisms to handle spike discontinuities across layers and inter-layer error propagation. This will help assess the trade-off between model complexity and performance in real-time applications. Also, further research lines can include a technique to alleviate the issue of imbalanced dataset and the adaptations for using SNN considering multiple measurements from different sources and data types. An evaluation of the impact of different time-varying functions can be considered. The use of fuzzy interval methods is being explored to explicitly capture uncertainties from the training process, so to better understand the behaviour of an SNN-based classifier when dealing with rail data. In addition, advancements in machine learning, signal processing, and railway engineering can enhance the development of rail defect detection techniques [53]. For instance, deep learning models, such as convolutional neural networks (CNNs), can enable autonomous feature extraction, alleviating reliance on expert knowledge and improving detection performance. Applying transfer learning techniques can help models adapt to new conditions by using knowledge learned from different railway networks. This allows for defect detection in environments where models have not been previously trained or tested. Implementing advanced denoising algorithms can enhance the detection of subtle defects by improving the signal-to-noise ratio. This makes it easier to identify and address issues in their early stages. Deploying high-frequency sensors, such as ABAs, on operational trains can enable continuous monitoring of rail infrastructure, enhancing detection capabilities and predictive maintenance. Incorporating additional monitoring data sources and measurements can be considered with the development of data fusion techniques to enhance the accuracy and reliability of rail defect detection.

CRediT authorship contribution statement

Wassamon Phusakulkajorn: Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Conceptualization. **Jurjen Hendriks:** Writing – review & editing, Methodology, Experimental Work, Data Acquisition, Investigation, Data curation. **Zili Li:** Writing – review & editing, Methodology, Experimental Work, Investigation, Supervision, Resources. **Alfredo Núñez:** Writing – review & editing, Supervision, Resources, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

W. Phusakulkajorn work was supported by the Royal Thai Government. Measurement campaigns are from the project In2Track3 under grant agreement No. 101012456 by the Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme. This research was supported by ProRail and by the Europe's Rail Joint Undertaking Flagship Project IAM4RAIL - Holistic and Integrated Asset Management for Europe's RAIL System under Grant Agreement No 101101966. Disclaimer: Funded by the European Union. Views and opinion expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Europe's Rail Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them. The project FP3-IAM4Rail is supported by the Europe's Rail Joint Undertaking and its members.

Co-funded by
the European Union

Europe's Rail

Data availability

Data will be made available on request.

References

[1] Z. Li, Chapter 13 - squats on railway rails, in: R. Lewis, U. Olofsson (Eds.), Wheel–Rail Interface Handbook, Woodhead Publishing, 2009, pp. 409–436.

[2] M. Molodova, Z. Li, A. Núñez, R. Dollevoet, Automatic detection of squats in railway infrastructure, IEEE Trans. Intell. Transp. Syst. 15 (2014) 1980–1990.

[3] Y. Long, W. Guo, N. Yang, C. Dong, M. Liu, Y. Cai, Z. Zhang, Research progress of intelligent operation and maintenance of high-speed railway bridges, Intell. Transp. Infrastruct. 1 (2022) liac015.

[4] H. Zhu, H. Li, A. Al-Juboori, D. Wexler, C. Lu, A. McCusker, J. McLeod, S. Pannila, J. Barnes, Understanding and treatment of squat defects in a railway network, Wear 442–443 (2020) 203139.

[5] A. Jamshidi, S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, R. Dollevoet, Z. Li, B. De Schutter, A big data analysis approach for rail failure risk assessment, Risk Anal. 37 (2017) 1495–1507.

[6] Z. Song, T. Yamada, H. Shitara, Y. Takemura, Detection of damage and crack in railhead by using eddy current testing, Electromagn. Anal. Appl. 3 (2011) 546–550.

[7] P. Liu, S. Liu, C. Yu, L. Gu, Y. Zhou, C. Zhao, Z. Huang, Quantitative evaluation of the rotational stiffness of rail cracks based on the reflection of guided waves, Intell. Transp. Infrastruct. 1 (2022) liac008.

[8] S. Faghih-Roohi, S. Hajizadeh, A. Núñez, R. Babuska, B. De Schutter, Deep convolutional neural networks for detection of rail surface defects, in: Proceedings of the International Joint Conference on Neural Networks 2016, 2016, pp. 2584–2589.

[9] Z. Li, M. Molodova, A. Núñez, R. Dollevoet, Improvements in axle box acceleration measurements for the detection of light squats in railway infrastructure, IEEE Trans. Ind. Electron. 62 (2015) 4385–4397.

[10] A. Jamshidi, S. Hajizadeh, Z. Su, M. Naeimi, A. Núñez, R. Dollevoet, B. De Schutter, Z. Li, A decision support approach for condition-based maintenance of rails based on big data analysis, Transp. Res. Part C: Emerg. Technol. 95 (2018) 185–206.

[11] Z. Yuan, S. Zhu, C. Chang, X. Yuan, Q. Zhang, W. Zhai, An unsupervised method based on convolutional variational auto-encoder and anomaly detection algorithms for light rail squat localization, Constr. Build. Mater. 313 (2021) 125563.

[12] W. Maass, Networks of spiking neurons: The third generation of neural network models, Neural Netw. 10 (1997) 1659–1671.

[13] L. Guo, Q. Zhao, Y. Wu, G. Xu, Small-world spiking neural network with anti-interference ability based on speech recognition under interference, Appl. Soft Comput. 130 (2022) 109645.

[14] A. Solairaj, G. Sugitha, G. Kavitha, Enhanced elman spike neural network based sentiment analysis of online product recommendation, Appl. Soft Comput. 132 (2023) 109789.

[15] I. Vázquez, B. Ayasi, H. Seker, J. Luengo, J. Sedano, A. García-Vico, Combining traditional and spiking neural networks for energy-efficient detection of eimeria parasites, Appl. Soft Comput. 160 (2024) 111681.

[16] J. Zhang, T. Yang, C. Jiang, J. Liu, H. Zhang, Chaotic loss-based spiking neural network for privacy-preserving bullying detection in public places, Appl. Soft Comput. 169 (2025) 112643.

[17] P. Wu, E. Tian, H. Tao, Y. Chen, Data-driven spiking neural networks for intelligent fault detection in vehicle lithium-ion battery systems, Eng. Appl. Artif. Intell. 141 (2025) 109756.

[18] J. Qu, Z. Gao, T. Zhang, Y. Lu, H. Tang, H. Qiao, Spiking neural network for ultralow-latency and high-accurate object detection, IEEE Trans. Neural Netw. Learn. Syst. 36 (3) (2025) 4934–4946.

[19] S. Kim, S. Park, B. Na, S. Yoon, Spiking-YOLO: Spiking neural network for energy-efficient object detection, Proc. AAAI Conf. Artif. Intell. 34 (2020) 11270–11277.

[20] Q. Su, Y. Chou, Y. Hu, J. Li, S. Mei, Z. Zhang, G. Li, Deep directly-trained spiking neural networks for object detection, in: 2023 IEEE/CVF International Conference on Computer Vision, ICCV, 2023, pp. 6532–6542.

[21] H.K. Taluja, A. Taluja, I. Kala, B. Mallala, Smart office automation using multi-dimensional attention spiking neural network for face recognition in internet of things, Appl. Soft Comput. (2025) 112967.

[22] X. Bai, Y. Huang, H. Peng, Q. Yang, J. Wang, Z. Liu, Spiking neural self-attention network for sequence recommendation, Appl. Soft Comput. 169 (2025) 112623.

[23] L. Guo, C. Li, Y. Wu, M. Man, Evaluation of the anti-disturbance capability of fMRI-based spiking neural network based on speech recognition, Appl. Soft Comput. 175 (2025) 113069.

[24] S.M. Bohte, J.N. Kok, H.L. Poutré, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (2002) 17–37.

[25] O. Booij, H.T. Nguyen, A gradient descent rule for multiple spiking neurons emitting multiple spikes, Inform. Process. Lett. 95 (2005) 552–558.

[26] R. Gütig, H. Sompolinsky, The tempotron: a neuron that learns spike timing-based decisions, Nature Neurosci. 9 (2006) 420–428.

[27] S. Ghosh-Dastidar, H. Adeli, A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection, Neural Netw. 22 (2009) 1419–1431.

[28] J.J. Wade, L.J. McDaid, J.A. Santos, H.M. Sayers, SWAT: A spiking neural network training algorithm for classification problems, IEEE Trans. Neural Netw. 21 (2010) 1817–1830.

[29] F. Ponulak, A. Kasiński, Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting, Neural Comput. 22 (2010) 467–510.

[30] A. Jeyasothy, S. Sundaram, N. Sundararajan, SEFRON: A new spiking neuron model with time-varying synaptic efficacy function for pattern classification, IEEE Trans. Neural Netw. Learn. Syst. 30 (2019) 1231–1240.

[31] I. Hussain, D.M. Thounaojam, WOLIF: An efficiently tuned classifier that learns to classify non-linear temporal patterns without hidden layers, Appl. Intell. 51 (2021) 2173–2187.

[32] A. Jeyasothy, S. Suresh, S. Ramasamy, N. Sundararajan, Development of a novel transformation of spiking neural classifier to an interpretable classifier, IEEE Trans. Cybern. 54 (2024) 3–12.

[33] Z. Li, X. Zhaoa, C. Esvelda, R. Dollevoet, M. Molodova, An investigation into the causes of squats—Correlation analysis and numerical modeling, Wear 265 (2008) 1349–1355.

[34] A. Jamshidi, A. Núñez, R. Dollevoet, Z. Li, Robust and predictive fuzzy key performance indicators for condition-based treatment of squats in railway infrastructures, Infrastruct. Syst. 23 (2017) 04017006.

[35] F. Ponulak, A. Kasinski, Introduction to spiking neural networks: Information processing, learning and applications, Acta Neurobiol. Exp. 71 (2011) 409–433.

[36] E.M. Izhikevich, Which model to use for cortical spiking neurons? IEEE Trans. Neural Netw. 15 (2004) 1–8.

[37] G. Indiveri, S.-C. Liu, Memory and information processing in neuromorphic systems, Proc. IEEE 103 (2015) 1379–1397.

[38] P.A. Merolla, J.V. Arthur, R. Alvarez-Icaza, A.S. Cassidy, J. Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S.K. Esser, R. Appuswamy, B. Taba, A. Amir, M.D. Flickner, W.P. Risk, R. Manohar, D.S. Modha, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science 345 (2014) 668–673.

[39] S. Dora, S. Sundaram, N. Sundararajan, An interclass margin maximization learning algorithm for evolving spiking neural network, IEEE Trans. Cybern. 49 (2019) 989–999.

[40] Q. Zhou, C. Ren, S. Qi, An imbalanced R-STDP learning rule in spiking neural networks for medical image classification, IEEE Access 8 (2020) 224162–224177.

[41] P. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (1990) 1550–1560.

[42] H.A. Mallot, Coding and Representation, vol. 2, Springer, Berlin, Germany, 2013, pp. 743–765.

[43] W. Gerstner, W.M. Kistler, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge Univ. Press, Cambridge, U.K, 2002.

[44] C. Koch, I. Segev, Methods in Neuronal Modeling: from Ions to Networks, MIT Press, 1998.

[45] J. Ceccarini, H. Liu, K. Van Laere, E.D. Morris, C.Y. Sander, Methods for quantifying neurotransmitter dynamics in the living brain with PET imaging, Front. Physiol. 11 (2020) 1–8.

[46] T. Wang, L. Yin, X. Zou, Y. Shu, M.J. Rasch, S. Wu, A phenomenological synapse model for asynchronous neurotransmitter release, Front. Comput. Neurosci. 9 (2016) 1–15.

[47] D. Dua, C. Graff, UCI machine learning repository, 2017, URL http://archive.ics.uci.edu/ml.

[48] A. Jeyasothy, S. Ramasamy, S. Sundaram, Meta-neuron learning based spiking neural classifier with time-varying weight model for credit scoring problem, Expert Syst. Appl. 178 (2021) 1–12, 114985.

[49] W. Jianan, H. Haisong, Y. Liguo, H. Yao, F. Qingsong, H. Dong, New imbalanced fault diagnosis framework based on cluster-MWMOTE and MFO-optimized LS-SVM using limited and complex bearing data, Eng. Appl. Artif. Intell. 96 (2021) 103966.

[50] A. Islam, S.B. Belhaouari, A.U. Rehman, H. Bensmail, KNNOR: An oversampling technique for imbalanced datasets, Appl. Soft Comput. 115 (2022) 108288.

[51] S.X. Chen, L. Zhou, Y.Q. Ni, X.Z. Liu, An acoustic-homologous transfer learning approach for acoustic emission–based rail condition evaluation, Struct. Heal. Monit. 20 (2021) 2161–2181.

[52] Y. Cao, Z. Zhou, C. Hu, W. He, S. Tang, On the interpretability of belief rule-based expert systems, IEEE Trans. Fuzzy Syst. 29 (2021) 3489–3503.

[53] W. Phusakulkajorn, A. Núñez, H. Wang, A. Jamshidi, A. Zoeteman, B. Ripke, R. Dollevoet, B.D. Schutter, Z. Li, Artificial intelligence in railway infrastructure: Current research, challenges, and future opportunities, Intell. Transp. Infrastruct. liad016 (2023) 1–24.