RESEARCH

Improving SpikeProp's Training Efficiency in Spiking Neural Networks for Large Language Models Through Innovative Weight Initialization

Falah. Y. H. Ahmed[1] · Muhammad Zakarya[1,2] · Naveed Khan[3] · Dilovan Asaad Zebari[4] · Mahmood Al-Bahri[1] · Bwalya Kelvin Joseph[5] · Abdullah Abdullah[6]

Received: 16 April 2025 / Revised: 12 July 2025 / Accepted: 11 August 2025

Abstract
Spiking neural networks (SNNs) use individual temporal spikes for computation and communication, simulating the actions of biological neurons. SNN had long been disregarded since it was thought to be intricate and difficult to analyze. We investigate the improvement of SpikeProp, a supervised learning model tailored for SNNs, in this work. Three distinct models are being proposed and investigated, including the proposed model 1, the proposed model 2, and the proposed model 3, each providing unique improvements to the SpikeProp algorithm. To accelerate convergence and adaptive learning rates, particle swarm optimization (PSO) and momentum factors are integrated into the proposed model 1. In proposed model 2, a rate dependency is introduced based on angle-driven learning. By incorporating PSO and learning rates, model 3 combines the strengths of both models 1 and 2. We believe, SNNs can be trained and classified more efficiently and accurately using these models. Furthermore, we examine how large language models (LLMs) might inform the design and interpretability of neural architectures and learning methodologies while also enhancing SNN training. Through the use of LLMs, we seek to enhance model transparency and encourage more Responsible AI (RAI) principles. A thorough evaluation and comparison of proposed models with traditional methods confirms that these models consistently outperform traditional methods for various real datasets. Consequently, they have a high potential for practical applications in neural network training in real-world settings and LLM-informed development, contributing to the advancement of AI systems.

Keywords Spiking neural network · Particle swarm optimization · Angel-driven dependency · Classification

1 Introduction

Numerous neurobiologists posit that the operational principles of the brain closely resemble those of a colossal parallel computing system, boasting approximately 10 billion simple processors, each capable of reacting in a matter of milliseconds to input stimuli. This resemblance extends to the realm of real-time applications, where parallels between parallel processing algorithms and conventional artificial neural network (ANN) techniques become evident. Consequently, ANN has been fashioned to mirror the intricacies of the human brain's structure [1]. This architectural mimicry has bestowed upon ANN a multitude of names, including distributed parallel processing, computing of neurons, natural intelligent systems, and methods-based machine learning [2].

The initial iterations of ANN were confined to binary inputs and outputs, characterized by binary impulses defined by width, phase, and time parameters, akin to signals modulated by the frequencies of neurons. However, recent research unveiled that neurons communicate through the emission of brief electrical pulses, known as rate

Published online: 06 November 2025

coding, where a higher output signal is a product of a higher firing rate [3]. Communication occurs solely through real spikes at two distinct time points: (i) spiking time; and (ii) no spiking time. This framework allows for the computation of neuron output values, with the network's response to input values becoming apparent when whole neurons have been released. Each neuron can be depicted by a fundamental firing rate and a consistently constant activation function, giving rise to the spiking neural network (SNN), often heralded as the third generation of ANNs [4].

The SNN employs individual spikes in the time domain for communication and computation, mirroring the operational principles of biological neurons. This methodology, known as pulse coding, encodes information through variations in pulse rates, enabling data multiplexing. A pressing issue in SNN research pertains to the development of effective training methods [5]. While research has explored biologically inspired local learning rules, these rules are limited to supervised learning and cannot adequately train networks for specific tasks. Traditional neural network research gained prominence due to the error-backpropagation learning rule, enabling networks to be trained to solve problems based on representative examples. Spike-based neural networks utilize the SpikeProp learning rule, which is applicable to networks consisting of spiking neurons and relies on precise spike time temporal coding [6]. This coding mechanism encodes input and output values by leveraging the accurate timing of input and output spikes.

Supervised learning using SpikeProp in SNNs typically leverages a gradient descent method that explicitly computes the error function's gradient through the backpropagation algorithm [4, 7]. This process enables SNNs to learn desired firing times for output neurons by adjusting weight parameters in the spike response model. Numerous experiments have sought to elucidate aspects like initialization parameters' roles and the significance of negative weights [8]. The optimization of SpikeProp can be achieved through the integration of supplementary learning rules that influence synaptic delays, thresholds, and temporal constants. This integration results in improved convergence speed and reduced network sizes, specifically tailored for certain learning tasks. A significant increase in speed was accomplished by approximating the function of firing time employing a logistic sigmoid. The application of the SpikeProp algorithm in recurrent network designs has demonstrated potential [9].

Nowadays, many algorithm-based metaheuristics have appeared as promising tools for ANN and deep learning training [10]. Unlike gradient-based algorithms, metaheuristic algorithms can escape local minima more easily, as they balance the exploration of the search space with efficient exploitation. They can optimize key ANN components, including hyperparameters, weights, layer count, neuron count, and learning rates. To solve the limitations of gradient descent in SNN training, there has been increasing interest in utilizing metaheuristic algorithms [11].

The performance of SNNs hinges on their architectural algorithms, and it is crucial to develop a learning algorithm for SNNs capable of data classification. Biologically inspired SNNs inherently support supervised learning [12], but this learning rule becomes fully effective when coupled with backpropagation [13, 14]. This integration, referred to as SpikeProp, employs spike time temporal coding for encoding input and output variables. The effectiveness of SpikeProp is contingent on the learning parameters of the backpropagation network. Consequently, this study concentrates on enhancing SpikeProp's performance by optimizing the initialization of backpropagation weights and SNN architectures. PSO emerges as a viable technique for this purpose.

The main purpose of this work is to introduce an optimal configuration for back-propagation within a spiking neural network (utilizing SpikeProp as the supervised learning rule). This is achieved by exploring various models, each contributing to the improvement of SpikeProp. The main contributions of this research are as follows:

- Introduction of optimal momentum factors for expedited convergence and estimation of adaptive learning rates. In this initial developmental phase, we aim to introduce well-suited momentum factors that accelerate convergence and estimate corresponding adaptive learning averages.
- To propose PSO-Spikeprop for enhanced learning optimization. We proposed an integration of PSO to optimize the learning process effectively.

- Proposition of parameters for μ angle-driven dependency learning rate in SpikeProp. In this third proposed phase, we present parameters for the μ angle-driven dependency learning rate within the SpikeProp framework.
- Hybridization of SpikeProp with PSO and integration of μ angle-driven dependency learning rate. In this developmental model, our objective is to enhance the overall performance of SpikeProp in SNNs by combining it with PSO and integrating the μ angle-driven dependency learning rate. This hybrid approach fine-tunes the learning process and enhances network performance.
- The study will conduct a comprehensive assessment and comparison of these proposed models against conventional methods such as SpikeProp and standard BackPropagation in SNNs. This analysis will provide valuable insights into the relative effectiveness of the proposed developments and their potential for practical applications in neural network training.

The rest of the paper is organized as follows. A review of the related work is presented in Sect. 2. In Sect. 3, related material and methodologies are explained. In Sect. 4, we describe our suggested algorithms. Experimental evaluation and results are elaborated in Sect. 5. Finally, Sect. 6 concludes this work and offers a few future research directions.

2 Related Work

Presently, a multitude of researchers have proposed an array of learning algorithms after conducting extensive investigations into the spiking neural network (SNN) model. The utilization of an SNN to address real-world challenges necessitates a meticulous configuration process encompassing the selection of appropriate spiking neurons, network topology, and the deployment of efficient learning algorithms. In contrast, it is imperative to recognize that the configuration of an SNN exerts a direct influence on its generalization capacity. Traditional learning algorithms, such as SpikeProp, frequently exhibit sluggishness and susceptibility to becoming ensnared in local minima. This, in turn, impedes the network model's optimal performance and may lead to overfitting issues, thereby significantly constraining the practical utility of SNNs [15].

Unsupervised and supervised learning rules are designed to harness the full potential of spiking neurons. Spike timing dependent plasticity (STDP), exemplifying unsupervised learning, provides a biologically plausible avenue for synaptic learning within SNNs. The STDP, depending on the learning rules, operates by adjusting synapse weights connecting pre- and post-synaptic neurons in accordance with the correlation of their respective spike times [16]. In a notable demonstration of the application of STDP, Diehl, and Cook [17] employed this mechanism to train a two-layer SNN, featuring lateral inhibitions, in an unsupervised learning paradigm. Their work introduces an SNN tailored for digit recognition, underpinned by biologically plausible components, including conductance-based synapses rather than current-based ones, spike-timing-dependent plasticity with time-dependent weight adjustments, lateral inhibition, and an adaptive spiking threshold. It is noteworthy that their approach abstains from relying on teaching signals or class labels in network training [18]. Note that weight adaptation plays a crucial role in neural network training, especially for SpikeProp-based models. However, in order to highlight the uniqueness of our proposed models, we decided to discuss weight adjustment individually.

In contrast, supervised learning, the cornerstone of traditional artificial neural networks (ANNs), faces formidable challenges when, in particular, applied directly to SNNs due to the inherent discontinuity and non-differentiability of spiking neurons [19]. Several strategies have emerged to circumvent this obstacle, among them Ponulak and Kasiński's supervised learning framework for physiologically plausible neurons [20]. Their experiments corroborate the effectiveness of their approach in training spiking neurons to replicate specific neuronal firing patterns seen in response to synaptic stimulation, even when there is background noise. Furthermore, they showcase the utility of their learning rule in decision-making tasks, demonstrating the ability to classify input signals based solely on the temporal configuration of spikes. This decision-making is conveyed through precisely timed spike trains, even in cases when there is a time gap between the presentation of stimuli and the decision-

making process, and where the relevant information is strongly intertwined with the current brain activity. They also illustrate the training of neurons to reproduce spike sequences with adjustable time shifts relative to target templates. Mohammed et al. [21] present SPAN, a spiking neuron capable of supervised learning, facilitating the computation of spatio-temporal data represented by the accurate timing of neuronal spikes. Their algorithm converts spike trains into analog signals during the learning phase, enabling common mathematical operations. This transformation facilitates the application of the Widrow-Hoff rule to adjust synaptic weights and achieve the desired input/output spike behavior. However, these methods grapple with two primary drawbacks, namely overfitting and susceptibility to local minima, which curtail their practical utility in the context of SNNs.

These days, metaheuristic algorithms have surfaced as promising tools for training ANNs and deep learning. Saleh et al. [22] introduce a novel hybrid harmony search algorithm, integrated with an evolving spiking neural network (NHS-ESNN), designed for classification tasks. Harmony search enhances the standard model, improving the ESNN algorithm's flexibility, especially in determining the optimal number of pre-synaptic neurons essential for constructing the ESNN structure. The experimental evaluation conducted on various UCI machine learning datasets indicates that NHS-ESNN delivers competitive results in terms of accuracy of classification and other metrics, outperforming the standard ESNN. Similar efforts by Hussain et al. [23] put forward an elitist floating-point genetic algorithm equipped with a hybrid crossover algorithm for supervised training of multilayer feedforward SNNs. Their experiments attest to the computational efficiency and biological plausibility of the proposed algorithm, although it only supports the emission of a single spike, limiting its application to SNNs that allow multiple spike times in each neuron. These pioneering approaches contribute significantly to enhancing the generalization capability of SNNs. Nonetheless, it is crucial to acknowledge that challenges persist in the training and design phases of these models.

Numerous studies in the existing body of research have been conducted to enhance the efficiency of SNNs and the SpikeProp method, which is rooted in ANNs. Further efforts are needed to enhance the generalisation of error and accuracy in classification by advancing SNN and SpikeProp. The SpikeProp algorithm demonstrated excellent performance. Thus, to fill this lack of research, the proposed model 1 (PSOSpikeProp), SpikeProp using μ angle-driven learning rate dependency—model 2. In this study, we aim to improve the SpikeProp algorithm to be suitable for classification problems. The hybridization of model 4 and model 5 already exhibits very good performance.

3 Materials and Methods

This section introduces a novel design and implementation approach aimed at enhancing the performance of the SpikeProp algorithm. Through empirical analysis, it has been ascertained that various error metrics such as mean squared error (MSE), root mean square error (RMSE), mean absolute percentage error (MAPE), and mean absolute deviation (MAD) within the SpikeProp framework can be mitigated by the incorporation of μ angle-driven learning rate dependency (proposed model 2). Nonetheless, the employment of μ angle-driven learning rate dependency could be perceived as a decision-making process from a singular perspective. To potentially further ameliorate error reduction and expedite the learning process, the adoption of a multi-perspective decision-making approach is considered. This rationale leads to the integration of PSO as a means to augment the SpikeProp algorithm. In essence, the amalgamation of SpikeProp and PSO, as represented by the proposed model (1) called PSO-SpikeProp, is expected to yield an algorithm imbued with numerous advantageous attributes. By exploring this innovative fusion of techniques, we delve into the prospects of enhancing the algorithm's performance, robustness, and efficiency, offering a substantial contribution to the field of neural network optimization [24, 25]. We aimed to draw attention to the distinctions from traditional static update rules and to clearly demonstrate the dynamic nature of our method, which is based on the learning angle, error direction, and prior weight changes, by presenting the weight adjustment phase separately.

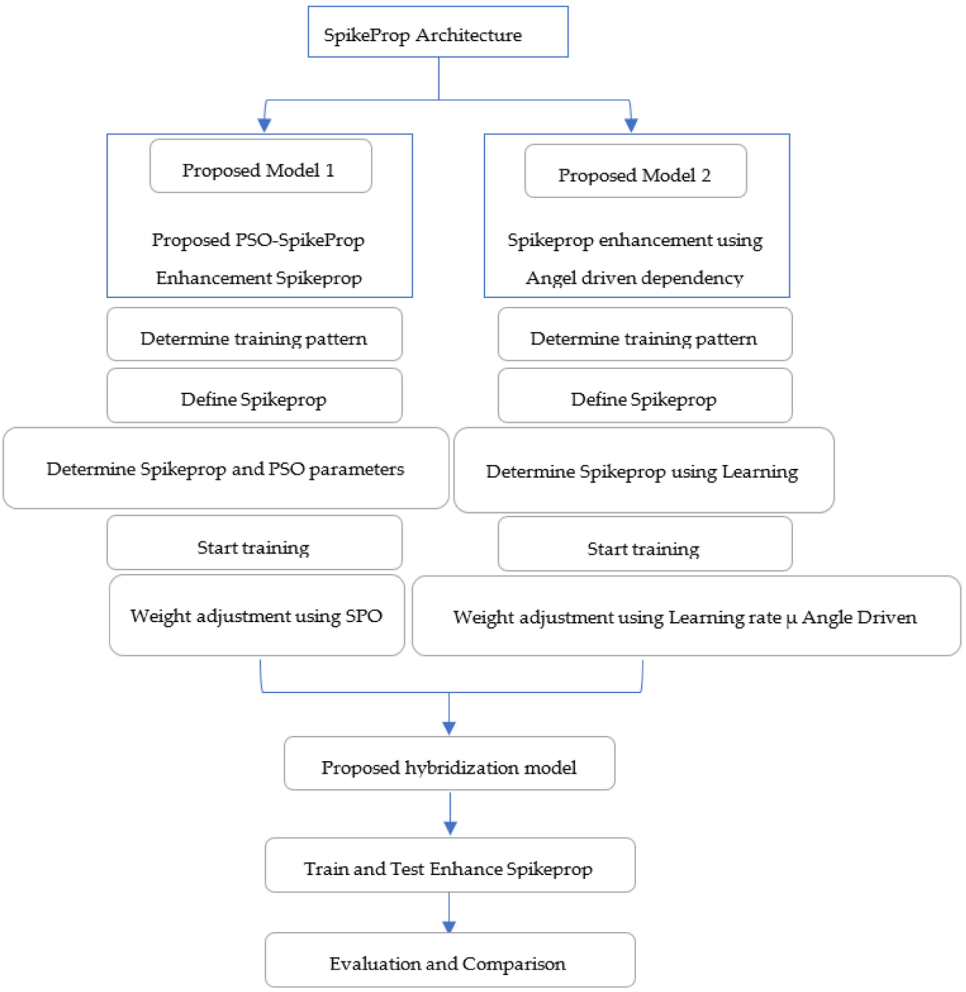

Fig. 1 A proposed framework for Spikeprop enhancement with various models i.e., PSO (model 1), learning rate (model 2), and hybridization

This study investigation is focused on optimizing the SpikeProp algorithm by combining two effective techniques, akin to the way superior genetic strains are obtained through cross-breeding. The study aims to enhance SpikeProp's architectural framework through a dual-pronged approach. Firstly, it incorporates PSO for optimizing the learning process, as detailed in proposed model 1. Secondly, it introduces parameter proposals that govern the μ angle-driven dependency learning rate within the SpikeProp algorithm, as outlined in proposed model 2. This research explores the integration of these two models and their individual contributions to SpikeProp's overall enhancement, highlighting the potential benefits of this hybridization. This amalgamation of techniques has the capacity to elevate the performance and efficiency of the SpikeProp algorithm, with promising implications for advanced applications in neural network training and optimization. Figure 1 shows the proposed framework for this study.

In this study, it is essential to distinguish between the training and testing phases of the model. In the training phase, the datasets employed are pre-defined, and the primary objective is to establish the procedure and associated weights required for accurate classification. This process essentially configures the neural network to perform a specific task. Conversely, in the testing phase, the datasets are novel and unencountered during the training process. The testing phase's primary purpose is to evaluate the model's ability to correctly classify these previously unseen datasets, utilizing the procedure and weights derived from the training phase. In our research, we explore and employ two distinct methodologies for implementing the training and testing processes for ANN and SNN, providing a comprehensive examination of their performance and capabilities. So far these methods facilitate a

deeper understanding of how these networks adapt and generalize to various data types, shedding light on their efficacy in real-world applications [26].

3.1 SpikeProp Acceleration

The backpropagation (BP) method uses the steepest descent method (SD) to modify the weights by gradual amounts at each iteration. Nevertheless, using a constant learning rate results in the network being stuck in the local minimum, repeatedly oscillating with a step size near this minimum. BP's performance is impacted by the values assigned to the starting weights. Hence, the selection of appropriate beginning weights is crucial. The learning rate is influenced by the initial weights, the magnitude of weight adjustments, and the gradients of descent. Bohte et al. [27] used the supervised learning method to train an SNN using a modified backpropagation (BP) technique. This modification enhanced the efficiency and speed of the optimization process. SpikeProp is an algorithm that leverages the parameters (weights initialization, threshold, learning rate, time step, whether to allow negative weights) [27]. Although Bohte has achieved success in pioneering the development of SpikeProp, it is crucial to acknowledge that the algorithm works on a fixed-time convergence basis. This paper suggests enhancing Bohte's method by including acceleration parameters that operate using the following parameters using PSO in SpikeProp (proposed model 1) and Adapting learning rate (proposed model 2).

4 Proposed Work

In this section, we propose three different algorithm: (i) PSO-SpikeProp; (ii) SpikeProp with learning rate; and (iii) hybrid model i.e., PSO-SpikeProp with learning rate. The PSO-SpikeProp combines PSO with SpikeProp to optimize initial synaptic weights and accelerate convergence using global best solutions (Sect. 4.1). The angle-driven SpikeProp further enhances SpikeProp by dynamically adjusting the learning rate and momentum based on the angle between error and weight changes to improve stability and convergence (Sect. 4.2). Finally, the hybrid model integrates PSO-SpikeProp with angle-driven learning rate adaptation to leverage both global exploration and local optimization for efficient and accurate SNN training (Sect. 4.3). Table 1 provides a list of all the mathematical notations with their brief descriptions.

4.1 The Proposed Model 1 (PSO-SpikeProp)

SpikeProp and PSO are two different algorithms. SpikeProp is based on firing times (of the synapses) while PSO is based on initial weights. , In SpikeProp, the joining of two artificial neurons (nodes) is called a synapse. It has a weight, which establishes how strongly one neuron influences another. Spiking models more closely resemble actual brain activity by having synapses transmit discrete spikes (events) rather than continuous values. The instant a neuron fires a spike is known as the firing time. Over time, each neuron incorporates input. The neuron "fires" (produces a spike) when the total amount of input beyond a threshold. For example, SpikeProp modifies weights according to the exact timing of these spikes—not just if a neuron fires, but when it fires is crucial. SpikeProp learning process uses feed-forward and feed-back propagation, while PSO relies on the feed-forward path only. SpikeProp takes some significant time to get to the optimum solution because it uses random initial weights to start the computation process. With PSO, the initial weights used are already good (through getting Gbest). Hence, the computation time required should be much shorter.

In this study, a new model is proposed, which is called PSO-SpikeProp (the model combines the slower but more robust SpikeProp with the faster but less robust PSO). PSO-SpikeProp improves the search for the global optimum by adopting some heuristic knowledge obtained from the PSO algorithm and passing this knowledge to the gradient descent-based search. When the optimal fitness value for all particles remains unchanged for many generations, the search process will transition to gradient descent search. The heuristic information is utilized to

Table 1 List of mathematical notations used in the proposed models

Notation	Description
W	Synaptic weight matrix of the SpikeProp network
w_{ij}	Weight between neuron i and neuron j
V_i	Velocity of particle i in PSO
W_i	Position (weights) of particle i in PSO
P_i	Personal best position (weights) of particle i
$Gbest$	Global best position among all particles
C_1	Acceleration constant toward the global best (PSO)
C_2	Acceleration constant toward the personal best (PSO)
r_1, r_2	Random numbers in PSO update equations (uniformly distributed in [0,1])
ω	Inertia weight for controlling velocity persistence (PSO)
Δt	Time interval or pulse duration in SpikeProp
t_j	Actual spike time of output neuron j
t_j^d	Desired spike time of output neuron j
$E(n)$	Error value at iteration n
$\Delta E(n)$	Change in error at iteration n
$\Delta W(n)$	Change in weight at iteration n
$\mu(n)$	Adaptive learning rate at iteration n
$\alpha(n)$	Adaptive momentum term at iteration n
δ_j	Error term (gradient) for neuron j
$\theta(n)$	Angle between $\Delta E(n)$ and $\Delta W(n-1)$
$\cos\theta(n)$	Cosine of angle used to adjust learning rate
$f(X; W)$	Output of the model given input X and weights W
$L(y, f(X; W))$	Loss function comparing true output y and predicted output $f(X; W)$
$E(W)$	Overall error or cost function for the network
σ	Standard deviation for Gaussian encoding
μ_i	Mean for Gaussian receptive field of input neuron i
$I_{\min}, I_{\max}$	Minimum and maximum input values
β	Scaling factor controlling Gaussian field width ($1 \leq \beta \leq 2$)
NT	Number of training samples

avoid wasting many synaptic times for a vain search. PSO-SpikeProp expends less synaptic time than the PSO and the SpikeProp algorithms. In other terms, PSO-SpikeProp utilizes less synaptic time to obtain more training accuracy than the SpikeProp standard algorithm or the BP standard algorithm. The rate of mean recognition of PSO-SpikeProp is smooth, making PSO-SpikeProp more stable than the SpikeProp standard algorithm and the BP standard algorithm [28–30].

The proposed approach involves enhancing the efficiency of SpikeProp through the utilization of four fundamental parameters from PSO. These parameters include the acceleration constant for the global best position C_1, the acceleration constant for the personal best position C_2, the time interval Δt, which is dependent on the duration of SpikeProp, and the number of particles employed in SpikeProp. The utilization of acceleration constants in the simulation serves to define the collective behavior of particles inside the swarm. The Pbest and Gbest positions are determined by the constants C_1 and C_2. The optimal solution at a global scale is contingent upon the pulse time of SpikeProp, denoted by the constant C_1. The optimal solution for an individual's personal best in particle optimization is contingent upon the pulse time of the SpikeProp algorithm, denoted by the constant C_2. In the event that the value of C_1 exceeds that of C_2, the swarm will exhibit movement in the vicinity of the global best solution. Conversely, in instances where C_2 surpasses C_1, the swarm has a tendency to navigate in the vicinity of the individual's personal best. The parameter Δt in SpikeProp represents the temporal duration of each pulse interval

Table 2 PSO parameters used in SpikeProp algorithm

Parameter	Value
C_1 (Gbest)	1.0 Example
C_2 (Pbest)	1.0 Example
Δt	0.1
Particle numbers	20
The dimension of problems	Depends on the SpikeProp architecture datasets
Stop condition	Minimum number of errors or maximum number of repetitions
Particles range	From −1 to 1 [−1, 1]

within which movements take place in the solution space of the pool processes. Decreasing the aforementioned parameters in the SpikeProp algorithm results in enhanced precision in the exploration of the solution space, as well as an increased temporal value of the most intense pulse in SpikeProp on the respective node. The extent of coverage of the problem is directly proportional to the number of particles present in the swarm or simulation using the SpikeProp form. Consequently, a larger number of particles will result in a reduced optimization requirement [31].

The optimization of parameters in the SpikeProp solution space can be enhanced by adapting them based on the node with the highest pulse time. SpikeProp utilizes the fundamental parameters in PSO. The sub-parameters of the issue, such as particle size, particle count, and termination criterion, are contingent upon the dataset. The execution time of SpikeProp utilizing PSO is substantially influenced by the number of particles. A trade-off exists within the range of feasible swarms and the duration required for execution. PSO, when equipped with a carefully chosen set of parameters, has the capability to exhibit high performance across various scenarios. The study employed many parameters, which are outlined in Table 2. Additionally, the particle location, specifically the weight and bias values, were randomly initialized. The initial position velocity value was set to 0.

The location of each particle in the swarm is denoted by a collection of weights corresponding to the current repetitions. The weight number of the network is determined by the dimension of the practical swarm. To decrease the occurrence of learning errors, it is advisable for the particle to navigate inside the weight space. Modifying the weight of the network involves adjusting its position with the aim of minimizing the number of iterations required. In each iteration, a velocity calculation is performed to determine the subsequent movement of the particle's position. The utilization of a novel set of weights facilitates the acquisition of a fresh error, thus leading to a revised position. In the PSO algorithm, the updated weights are recorded in the system, regardless of whether there is a discernible enhancement in performance. This method is applicable to all particles. The particle position that is considered to be the global best is the one that exhibits the lowest number of mistakes on a global scale. The training procedure concludes either upon reaching the desired minimum error threshold or upon exceeding the maximum permissible number of repetitions. Upon the conclusion of the training process, the weights are employed to compute the error of classification pertaining to the training patterns. The network is tested using identical patterns and a consistent set of weights. The $Pbest(C_2)$ value and $Gbest(C_1)$ value are performed to solve the learning error problems. The SpikeProp weight and SpikeProp bias are obtained by including the computed velocity value, as shown in Eqs. (1) and (2). A novel set of positions is used to generate the novel learning mistake. The suggested technique aims to minimize the number of repetitions required to write the classification dataset output while achieving the lowest possible error.

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \Delta w_{ij} \cdot \Delta t^{(t)} = t_j^d \tag{1}$$

$$\Delta w_{ij}(n) = w_{ij}(n) + C_1 \cdot (p_{\text{best},n} - w_{ij}(n)) + C_2 \cdot (g_{\text{best},n} - w_{ij}(n)) \cdot \Delta t^{(t)} = t_j^d \tag{2}$$

The suggested approach involves the use of PSO to the SpikeProp method. Several particles have been utilized to deal with eight distinct dataset challenges as well as the XOR dataset. The research aims to minimize mistakes,

improve the learning rate of SpikeProp, and accelerate the algorithmic process. Figure 2 illustrates that particle swarms with randomly assigned beginning rates exhibit varying MSE. Throughout the learning process, all particles collectively converge to get the optimal $Pbest(C_2)$ value and $Gbest(C_1)$. The term $Gbest$ fit refers to the best possible solution.

For the work in this study, a 3-layer SpikeProp is utilized to perform the classification. The SNN SpikeProp architecture consists of the input layer, the hidden layer, and the output layer. The number of nodes varies for each layer (determined by the classification process). The number of input layers and output layers is often computed using the number and class of attributes.

In the implementation of PSO-SpikeProp, the position of each particle in a swarm is represented by a set of weights corresponding to a point in the iteration process. The size of each particle is determined by the dimension of the vector Wh. The particle changes its position in the weight space while trying to reduce the learning error or MSE, RMSE, MAPE, and MAD. Particle movement means updating the weight of the network to minimize the error of the current iteration. For each iteration, all the particles update their positions by using Eq. (3). This will move them to new positions. The new positions are sets of new weights that will be utilized in the subsequent iteration, during which new errors will be computed. In the PSO algorithm, the weights are updated even in the absence of observed improvements. The process of iteration is carried out in a simultaneous manner for all particles. The particle exhibiting the lowest error is referred to as the global best particle, denoted as P_{best}. The training procedure continues until a desirable level of error is achieved or until the computational limit is surpassed. Upon completion of the training process, the weights are utilized to calculate the error rate of classification according to the training patterns. The network is tested using a consistent set of weights.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3}$$

$$\mu_i = I_{\max} + \frac{(2_i - 3)}{2} \cdot \frac{(I_{\max} - I_{\min})}{M - 2} \tag{4}$$

$$\sigma = \frac{1}{\beta} \cdot \frac{(I_{\max} - I_{\min})}{M - 2} \tag{5}$$

The calculation of the firing time of an input neuron i is determined by the point of intersection of the Gaussian function, as specified in Eq. 3. The calculation of the Gaussian means μ_i is determined by Eq. 4, whereas the computation of its breadth α is determined by Eq. 5, utilizing the input variable interval of $I_{\max}$ and $I_{\min}$. The variables $I_{\min}$ and $I_{\max}$ are used to denote the lower and upper bounds of the input values, respectively. The parameter β governs the extent of each Gaussian receptive field, with a constraint of $1 \leq \beta \leq 2$.

In PSO-SpikeProp, there is no propagation. The feed-forward in SpikeProp produces the learning error depending on the values of weights and bias (PSO positions) used. Values of both $pbest$ and $gbest$ are implemented to update the velocity using Eq. (5), which obtains the position adjustment value to achieve an optimal solution (targeted learning error). In adaptive PSO, we introduce a different selection strategy for determining initial weights. In an adaptive situation, the optimum position keeps on changing. It is futile to obtain the heuristic knowledge from PSO because this knowledge varies with time. Hence, it will be quicker to choose any random initial weights for the SpikeProp to work on. The PSO-SpikeProp procedure is illustrated in Algorithm 1. By utilizing PSO to improve global exploration and optionally utilizing SpikeProp's gradient-based refinement, this method outperforms both algorithms in terms of classification accuracy and convergence speed.

 https://doi.org/10.1007/s44196-025-00961-x

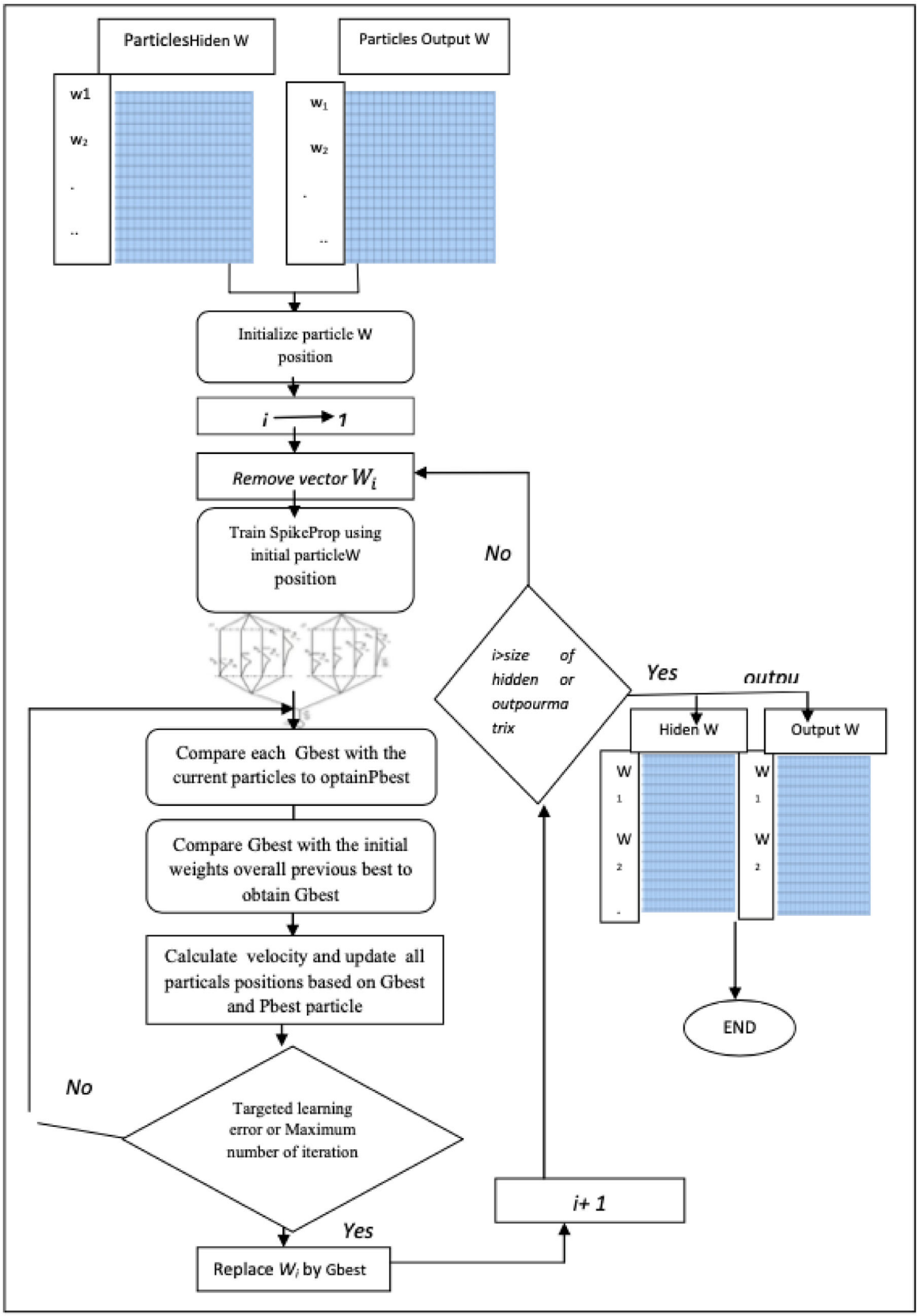

Fig. 2 Learning process of the proposed PSO-SpikeProp approach

Algorithm 1: The PSO-SpikeProp algorithm (model 1)

Step 1: Initialization
  Set swarm size S, inertia weight ω, cognitive coefficient c_1, social coefficient c_2
  Randomly initialize particle positions W_i and velocities V_i for $i = 1, \ldots, S$
  Set personal bests $P_i = W_i$, and evaluate initial fitness $f_i = E(W_i)$ using SpikeProp error.
    $E(W_i) = \frac{1}{2}\sum_j (t_j - t_j^d)^2$
  Set global best G as the W_i with the lowest $E(W_i)$
Step 2: Iterate Until Convergence or Max Iterations
for *each particle* $i = 1$ *to* S **do**
  Perform forward and backward pass using SpikeProp to compute $E(W_i)$
  if $E(W_i) < E(P_i)$ **then**
    Update personal best: $P_i \leftarrow W_i$
  end if
  if $E(W_i) < E(G)$ **then**
    Update global best: $G \leftarrow W_i$
  end if
  Generate random values $r_1, r_2 \in [0, 1]$
  Update velocity:
    $V_i \leftarrow \omega V_i + c_1 r_1 (P_i - W_i) + c_2 r_2 (G - W_i)$
  Update position (weights):
    $W_i \leftarrow W_i + V_i$
end for
Step 3: Apply Updated Weights to SpikeProp and Fine-Tune (Optional)
  Optionally use SpikeProp with updated weights for fine-tuning using gradient descent
Step 4: Return Best Weights
Return global best weights G

4.2 The Proposed Model 2 (Learning Rate μ for SpikeProp)

By using the learning rate to infer the angle dependence, the efficiency of SpikeProp has been enhanced. Instead of maintaining a constant learning rate during the training, it is advantageous to have a continuous learning rate throughout the training process. A typical backpropagation network has a constant learning rate within the range of 0 and 1. However, the SpikeProp method, which is a supervised learning technique for SNNs, encodes information by using the timing of spike trains. This technique has a resemblance to the conventional error backpropagation process used in sigmoidal neural networks, with the distinction that the learning parameter undergoes adaptive modification. SpikeProp relies entirely on the angle-driven relationship between the learning rate of backpropagation (BP) and its dependence. The suggested improved BP method is specifically intended to be used with SpikeProp instead of the normal BP approach.

SpikeProp is limited to learning a single spike per input neuron, while the standard rule is designed for input neurons that exhibit many firings. SpikeProp is capable of classifying patterns consisting of sets of real values, represented by the spike duration of each input neuron. The classification problems are equivalent to those of traditional neural networks employing conventional backpropagation. SpikeProp is a learning method that uses error-backpropagation. The reduction of error is contingent upon the timing of the output units' activation, as seen in Eq. (8).

$$E = \frac{1}{2}\sum_{m=1}^{NT}\sum_{j=1}^{N1}(t_j - t_j^d)^2 \tag{6}$$

Here, d denotes a pattern, and a set of patterns are denoted as t_j^d. Furthermore, the actual firing time of the unit j for the pattern d and j, the desired firing time of the unit j for the pattern d are $N1$ and NT, respectively. During each iteration, the weights are adjusted incrementally using a gradient descent technique.

$$\omega = \eta \frac{\partial E}{\partial \omega} \tag{7}$$

The learning rate η is denoted by $\eta > 0$. The gradient expansion can be performed using formula (8) for the error-backpropagation in typical multilayer networks, which remains the same.

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial t_i} \frac{\partial t_i}{\partial x_i} \frac{\partial x_i}{\partial w_{ij}^k} = \delta_i y_j^k \left(\frac{\partial E}{\partial t_i} \frac{\partial t_i}{\partial x_i} \right) \tag{8}$$

Here, $yli = \epsilon(t - t_i - d_k)$ indicates the set of immediate successor units of unit i. The learning algorithm depends on the explicit assessment of the gradient of $E = \frac{1}{2} \sum_j (t_j^{\text{out}} - t_j^d)$ with respect to the weights of each synaptic input to j.

In the optimization process, the SpikeProp algorithm automatically finds proper values for the learning rate during the training. Additionally, it modifies the momentum parameter, so changing both the step size and search direction. Once a neuron emits a solitary spike, it is prohibited from emitting another spike. Therefore, the approach is only appropriate for implementing the 'time-to-first-spike' coding scheme. The objective of this research is to provide a learning algorithm that can use the suggested learning angle-driven dependence μ for SpikeProp.

In this model, the learning rate and momentum utilized in backpropagation are altered by using a μ angle-driven dependence learning rate. The suggested methodology improves the learning process of SpikeProp by incorporating the angle computation between the change in error ΔE at time step n and the change in weights Δw at time step $n - 1$. The adaptation involves the adjustment of the angle to 90 degrees, in accordance with the Pythagorean theorem, which states that the square of the hypotenuse is equal to the sum of the squares of the other two sides. When the angle is less than 90 degrees, the learning rate exhibits an inverse relationship; however, for angles greater than 90 degrees, the learning rate experiences a drop. The application of mathematical techniques has been utilized to enhance the SpikeProp performance, as demonstrated in Equations (9-13).

Calculation:

$$\cos\theta(n) = \left. \frac{\Delta W(n-1)}{\sqrt{\Delta E(n)^2 + \Delta W(n-1)^2}} \right|_{t=t_j^d}^{(t)} \tag{9}$$

Adapting learning rate:

$$\mu(n) = \mu(n-1) \cdot (1 + 0.5 \cos\theta(n)) \tag{10}$$

Adapting the momentum:

$$\alpha(n) = \alpha(0) \cdot \left. \frac{\|\Delta E(n)\|}{\|\Delta W(n-1)\|} \right|_{t=t_j^d}^{(t)} \tag{11}$$

Adjusting the weights:

$$\Delta W_{ij}(n) = \mu(n) \cdot \left(\frac{\partial E}{\partial W_{ij}} + \alpha(n) \cdot \Delta W_{ij}(n-1) \Big|_{t=t_j^d}^{(t)} \right) \tag{12}$$

Fortunately, the rate at which learning occurs is significantly accelerated by employing the improved adaptation strategy as described by Eq. (13).

$$\mu(n) = \mu(n-1) \cdot (1 + 0.1 \cos\theta(n)) \tag{13}$$

Additionally, a backtracking method has been employed, wherein learning steps that exceed half of the learning rate duration are re-executed in the event of a rise in total error. Based on the aforementioned observations, it can be inferred that the learning rate exhibits enhanced performance in SpikeProp when compared to the regular SpikeProp algorithm.

The traditional SpikeProp configuration depends on the BP algorithm where the learning rate is kept strictly between 0 and 1 [0, 1]. Hence, choosing the adaptive learning rate is difficult. The proposed model 2 overcomes this problem by using the factor $cos(\theta)$ in Eq. 9, where is the angle between $\Delta E(n)$ and $\Delta w(n-1)$. The adaptation tries to adjust the angle to 90 degrees when $cos(\theta)$ becomes zero and the learning rate becomes constant. A constant learning rate implies that the learning process is at the top of the learning curve where the learning is at its optimum. The angle θ is inversely proportional to the learning rate. When the angle is small, $cos(\theta)$ becomes big and the learning rate is increased, and vice versa [32].

Initialization weights updating for Model 2 depends on the value of θ, the learning rate (n), and the momentum $\alpha(n)$. These three parameters accelerate the changes made on the weights in the direction of the optimum solution when the square error becomes zero (theoretically) and Eq. 11 is satisfied.

The traditional SpikeProp, used by many to date, can only accelerate the initial weights updating, where the changes of error with changes of the initial weights are the sole determining factor. The angle-driven dependency learning rate SpikeProp (proposed model 2) accelerates the initial weights updating using three dominant determining factors, which are θ, the learning rate μ(n), and the momentum $\alpha(n)$. In this way, proposed model 2 can outperform the traditional SpikeProp. As proposed model 2 is a legacy of the traditional SpikeProp; each step size in an iteration remains between 0 and 1. In many applications, this step size may not be of sufficient length to bring about satisfactory time for convergence. For proposed model 2, when weight updating is carried out after 20 iterative steps (20 epochs), the time taken to arrive at an optimum solution is shortened. The PSO-SpikeProp with angle-driven adaptive learning rate procedure is illustrated in Algorithm 2. In contrast to the static learning rate of SpikeProp, this angle-driven method dynamically modifies the learning rate and momentum according to the direction of weight and error changes, resulting in faster convergence and more stable training.

4.3 The Proposed Hybrid Model (Model 3)

The hybridization model proposed combines two distinct approaches, PSO-SpikeProp (proposed model 1) and a learning rate enhancement (proposed model 2), to create a more effective and efficient neural network training algorithm. Combining SpikeProp and PSO, the model merges two different algorithms, SpikeProp and PSO, which have their unique strengths and weaknesses. SpikeProp operates based on the timing of neuron spikes, while PSO relies on optimizing initial weight values. The PSO-SpikeProp model leverages heuristic knowledge obtained from the PSO algorithm. This knowledge is then applied to the SpikeProp algorithm to improve its performance. The model dynamically transitions between PSO and SpikeProp during training. When the best fitness value remains constant for a set number of generations, the algorithm switches from PSO to gradient descent-based search in SpikeProp. This should be noted that the PSO-SpikeProp is designed to use less synaptic time, which can lead to faster convergence to an optimal solution [33].

Traditional neural networks typically use a fixed learning rate during training. In contrast, this hybrid model introduces a dynamic learning rate for SpikeProp. The learning rate continuously changes throughout training. The learning rate adaptation in SpikeProp is driven by the angle between error changes ΔE and weight changes Δw. If the angle is less than 90 degrees, the learning rate increases inversely. If the angle is greater than 90 degrees, the learning rate decreases. To further enhance learning rate adjustment, the model employs a backtracking strategy. It reruns learning steps when the total error increases, helping to refine the learning rate more rapidly. The goal is to improve the SpikeProp algorithm by adapting the learning rate to enhance convergence speed and reduce errors during training.

The hybridization model combines these two components. PSO-SpikeProp provides a more efficient and robust initial training approach, with the ability to switch to gradient descent-based SpikeProp when certain conditions

 https://doi.org/10.1007/s44196-025-00961-x

Algorithm 2: Angle-driven adaptive learning rate SpikeProp algorithm (model 2)

Input: Training data (input, t_j^d), initial weights W
Output: Optimized weights W
Set initial weights W, learning rate $\mu(0)$, momentum $\alpha(0)$, $\Delta W_{\text{prev}} = 0$, $E_{\text{prev}} = \infty$, maximum number of epochs, and convergence threshold
for *epoch* $n = 1$ *to* `max_epochs` **do**
  for *each training sample* (*input*, t_j^d) **do**
    Forward and Backward propagation:
    Encode input as spike times
    Simulate SNN to compute output spike times t_j
    Compute error: $E(n) = \frac{1}{2}\sum_j (t_j - t_j^d)^2$
    for *each neuron* j **do**
      Compute $\delta_j = \frac{\partial E}{\partial t_j} \cdot \frac{\partial t_j}{\partial x_j}$
      for *each synapse* w_{ij} **do**
        Compute gradient: $\frac{\partial E}{\partial w_{ij}} = \delta_j \cdot y_j^k$
      end for
    end for
    Compute angle $\theta(n)$**:**
    $\Delta E(n) = E(n) - E_{\text{prev}}$
    $\cos\theta(n) = \frac{\Delta W(n-1)}{\sqrt{\Delta E(n)^2 + \Delta W(n-1)^2}}$
    Update learning rate and momentum:
    $\mu(n) = \mu(n-1) \cdot (1 + 0.5\cos\theta(n))$
    $\alpha(n) = \alpha(0) \cdot \frac{\|\Delta E(n)\|}{\|\Delta W(n-1)\|}$
    Update weights and trackers:
    for *each synapse* w_{ij} **do**
      $\Delta W_{ij}(n) = \mu(n) \cdot \frac{\partial E}{\partial w_{ij}} + \alpha(n) \cdot \Delta W_{ij}(n-1)$
      $w_{ij} = w_{ij} - \Delta W_{ij}(n)$
    end for
    $E_{\text{prev}} = E(n)$
    $\Delta W_{\text{prev}} = \Delta W(n)$
    if $E(n) <$ `convergence_threshold` **then**
      break
    end if
  end for
end for
return W

are met. Additionally, the learning rate enhancement method dynamically adjusts the learning rate during training to ensure optimal convergence. The combination of these approaches aims to provide a more effective and efficient training process for SNNs, reducing training time and improving accuracy. This hybridization model offers the benefits of both approaches while addressing their individual limitations, resulting in a more powerful training algorithm. In machine learning scenarios, $E(W)$ typically represents a loss or cost function, and mathematically it can be defined as follows:

$$E(W) = L(y, f(X; W)) \tag{14}$$

where $E(W)$ is the cost or error function, which quantifies the difference between the model's predictions and the actual target values. Similarly, $L(y, f(X; W))$ is the function that compares the true target values y with the predictions made by the model $f(X; W)$, which depend on the model parameters or weights, denoted as W. The choice of the specific loss function L depends on the machine learning task, and it could be MSE for regression or cross-entropy for classification, for example. Furthermore, y is the true target values or labels for a given dataset. Fillay, $f(X; W)$ is the model's predictions, which are determined by the model's input features X and the model

parameters W. Thus, in this representation, $E(W)$ is the result of applying a loss or cost function to compare the true target values y with the model's predictions $f(X; W)$, where W represents the set of model parameters that the algorithm seeks to optimize in order to minimize this cost or error function. The hybrid procedure is illustrated in Algorithm 3. The proposed hybrid approach, after conducting a global search for the best starting weights using PSO, employs an enhanced SpikeProp for local convergence that takes advantage of adaptive learning rates and momentum. There are numerous significant benefits to this combination. In order to improve exploration in the early phases of training, PSO first allows the model to escape local minima by conducting a global search for ideal initial weights. After detecting stagnation, the algorithm smoothly moves on to the SpikeProp phase, which improves convergence speed and accuracy by dynamically adjusting the learning rate based on the angle between error and weight changes. Utilizing momentum stabilizes weight updates, which speeds up learning even further. Furthermore, robust error minimization is ensured via a backtracking method that improves learning stages as error rises. When combined, these characteristics enable the hybrid model to outperform conventional SpikeProp or PSO alone in terms of convergence speed, classification accuracy, and adaptability [34].

Algorithm 3: Hybrid algorithm: PSO + angle-driven adaptive SpikeProp (model 3)

Input: Training data (X, y), swarm size S, learning rate $\mu(0)$, momentum $\alpha(0)$, thresholds
Output: Optimized weights W

Define swarm: positions W_i, velocities V_i, personal bests P_i, global best G
Set max epochs, convergence, stagnation, $stagnation\_count = 0$, $best\_fitness\_prev = \infty$
for *epoch* $n = 1$ *to* `max_epochs` **do**
  if $stagnation\_count <$ `stagnation_threshold` **then**
    for *each particle* i *in swarm* **do**
      Evaluate fitness: $E_i = L(y, f(X; W_i))$
      if $E_i < P_i$ **then**
        Update personal best: $P_i = E_i$
      end if
      if $E_i < best\_fitness\_prev$ **then**
        Update global best: $G = W_i$
        $best\_fitness\_prev = E_i$
        $stagnation\_count = 0$
      end if
      else
        $stagnation\_count$ += 1
      end if
    end for
    for *each particle* i **do**
      Update velocity: $V_i = \omega V_i + c_1 r_1 (P_i - W_i) + c_2 r_2 (G - W_i)$
      Update position: $W_i = W_i + V_i$
    end for
  end if
  else
    Initialize $W = G$, $\Delta W_{\text{prev}} = 0$, $E_{\text{prev}} = \infty$ [**Switch to SpikeProp Phase with adaptive** μ**:**]
    for *each training sample* $(input, t_j^d)$ **do**
      Encode input as spike times, simulate SNN to compute t_j
      Compute error: $E(n) = \frac{1}{2} \sum_j (t_j - t_j^d)^2$
      for *each neuron* j **do**
        Compute $\delta_j = \frac{\partial E}{\partial t_j} \cdot \frac{\partial t_j}{\partial x_j}$
        for *each synapse* w_{ij} **do**
          Compute gradient: $\frac{\partial E}{\partial w_{ij}} = \delta_j \cdot y_j^k$
        end for
      end for
      Compute $\Delta E(n) = E(n) - E_{\text{prev}}$
      Compute $\cos\theta(n) = \frac{\Delta W(n-1)}{\sqrt{\Delta E(n)^2 + \Delta W(n-1)^2}}$
      Update learning rate: $\mu(n) = \mu(n-1) \cdot (1 + 0.5 \cos\theta(n))$
      Update momentum: $\alpha(n) = \alpha(0) \cdot \frac{\|\Delta E(n)\|}{\|\Delta W(n-1)\|}$
      for *each synapse* w_{ij} **do**
        $\Delta W_{ij}(n) = \mu(n) \cdot \frac{\partial E}{\partial w_{ij}} + \alpha(n) \cdot \Delta W_{ij}(n-1)$
        $w_{ij} = w_{ij} - \Delta W_{ij}(n)$
      end for
      if $E(n) > E_{prev}$ **then**
        Reduce $\mu(n)$ by half and repeat step
      end if
      Update $E_{\text{prev}} = E(n)$
    end for
    Break ; `//Training ends after SpikeProp refinement`
  end if
end for
return *final weights* W

 https://doi.org/10.1007/s44196-025-00961-x

Table 3 Description and characteristics of various datasets

Dataset	Attributes	Classes	Samples	Input	Output	Training	Testing
Breast cancer	9	2	683	9	2	538	145
BTX	19	3	512	19	7	407	105
Diabetes	8	2	768	8	2	613	155
Heart	13	2	297	13	2	240	57
Hepatitis	19	2	155	19	2	123	32
Iris	4	3	150	4	3	120	30
Liver	6	2	345	6	2	276	69

5 Experimental Results

This section presents the results of the suggested upgraded Spikeprop method. The design of the suggested technique is presented, followed by an elucidation of the SpikeProp algorithm. The evaluation and comparisons have been carried out using metrics such as convergence error, average error, and correctness. A Python implementation based on PyTorch was used for the studies, utilizing the snnTorch[1] package for surrogate gradients and spiking neuron models.

The datasets in this study are divided into two parts: training and testing. These are done by partitioning the data randomly into 75% for the training set and the rest 25% for the testing set. The training part is used to train the Spikeprop and the proposed Spikeprop algorithm, while the testing part is used for error generalization from the classification procedure of both the Spikeprop network and the suggested models. The normalization of the datasets is set between [0, 1]. The evolutionary process of each suggested method is assessed and analyzed using all datasets. The results are obtained from ten runs for each dataset, and the average accuracy is computed accordingly.

5.1 Datasets

This work considers nine common datasets, publicly available on Kaggle,[2] to assess the efficacy of the suggested algorithms. Table 3 presents the properties of these real-world datasets in a corresponding manner. These datasets are actual collections of data that vary in terms of the number of accessible samples (ranging from 32 to 484), characteristics (ranging from 3 to 56), and classes (ranging from 2 to 10).

5.2 Evaluation and Analysis

This research used four error metrics to assess performance: MSE, RMSE, MAPE, MAD, and correctness. The MSE is a metric that quantifies the discrepancy between the projected historical data generated by all improved SpikeProp models and the actual data. It does this by squaring the errors and preventing the positive and negative errors from offsetting each other. This metric also has a tendency to exaggerate significant mistakes (by giving them more importance than lesser errors) by squaring them, which may be beneficial when comparing various models. The MSE is computed by directly calculating the average of the error values. Conversely, RMSE is the square root of MSE and is widely used as an error metric, commonly referred to as the quadratic loss function. RMSE is a statistical metric that computes the average of the absolute values of mistakes. It is particularly suitable when the errors are directly proportional to the absolute magnitude of the decreased error. MAPE is a statistical measure of relative error, calculated as the average percentage error of the data points. It is particularly suitable when the impact of forecast errors is more strongly influenced by the percentage error rather than the actual magnitude of

[1] https://snntorch.readthedocs.io/en/latest/.

[2] https://www.kaggle.com/organizations/uciml/datasets.

Table 4 Predicted results of standard SpikeProp and BP-ANN algorithms

Dataset	Standard SpikeProp algorithm				Standard BP-ANN algorithm			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.5711	0.5117	44.419	1.0234	0.821	0.6410	55.4437	1.391
BTX	2.1909	0.2697	44.451	1.8885	2.712	0.7890	89.3444	2.642
Diabetes	0.4922	0.4200	34.348	0.8400	0.789	0.6720	60.2087	1.399
Heart	0.3671	0.3359	29.820	0.6718	0.811	0.6410	53.7246	1.361
Hepatitis	0.7025	0.5589	48.509	1.1178	0.843	0.6510	56.6075	1.398
Iris	**0.1224**	**0.3456**	**19.124**	**1.0368**	0.872	0.7640	69.6606	2.356
Liver	0.7210	0.5427	43.184	1.0855	0.842	0.6510	58.4594	1.421
XOR	4.6900	2.6825	23.1325	2.6825	0.8167	4.4945	38.7580	8.9890
Wine	0.5144	0.2877	33.553	0.8631	0.753	0.4080	72.6499	1.125

approaches with the lowest values are better than the others—shown bold face

the mistake. The MAD is a statistical metric that computes the average distance between the anticipated historical data from the models and the actual historical data, considering the absolute magnitude of the difference for each pair of data points. The MAD is determined by computing the average of the mistakes and is particularly suitable when the reduced errors are directly proportional to the absolute magnitude of the errors.

5.2.1 Analysis of Standard SpikeProp and Standard BP

This section provides an overview of the outcomes obtained through the implementation of the standard SpikeProp algorithm across all datasets. The analysis of the data is conducted by evaluating the convergence to MSE, RMSE, MAPE, and MAD. These findings are accompanied by the classification performance, which is presented in Table 4. The experimental methodology employed for standard SpikeProp involves conducting ten iterations. According to the data presented in Table 4, it can be observed that the RMSE outperforms the MSE when evaluating different datasets of BTX, heart, iris, diabetes, breast cancer, liver, hepatitis, and XOR, in that order. The RMSE outperforms both the MAPE and the MAD as error assessments across all datasets. The aforementioned findings indicate that the SpikeProp algorithm provides direct evidence that networks composed of spiking neurons have the capability to perform intricate, non-linear operations using a temporal coding mechanism. According to the experimental findings, the SpikeProp method demonstrates the capability to accurately classify non-linearly separable datasets across various error measures, in comparison to conventional sigmoidal networks BP, as illustrated in Table 4. Table 4 presents a comprehensive analysis of the measurements obtained from the SpikeProp algorithm. Based on the data presented in the table, it is evident that the SpikeProp algorithm showcases a higher level of convergence with a reduced number of mistakes compared to the normal BP algorithm.

The validation of the analyses of standard BP for all datasets is conducted using MSE, RMSE, MAPE, and MAD, as presented in Table 4. According to the information provided in Table 4, it can be observed that the RMSE outperforms the MSE when considering various datasets for diabetes, heart disease, breast cancer, liver disease, hepatitis, iris, BTX, and XOR, respectively. The MSE outperforms both the MAPE and the MAD across all datasets, with the exception of the BTX dataset, where the MAD values are lower than the MSE. The conventional BP network is extensively employed by researchers in several domains for the purpose of solving classification issues [15]. Sander Bohte [27] developed the SpikeProp algorithm for the BP learning method. A comparison between BP and SpikeProp can be found in Table 4. The data indicates that BP exhibits a greater frequency of errors in comparison to SpikeProp.

5.2.2 Analysis of PSO-SpikeProp, SpikeProp with Angle-Driven Dependency (μ), and Hybrid Models

This section showcases the outcomes of the PSO-SpikeProp, SpikeProp with angle-driven dependency (μ), and hybrid models that were suggested in Sect. 5.2.2. In order to conduct analysis, the proposed model 1 is compared to a typical SpikeProp and BP. Table 5 displays the MSE generalization obtained from the testing portion. The MSE algorithm demonstrates superior performance on the diabetes, heart, and breast cancer datasets, but it is less effective for the liver, hepatitis, iris, BTX, and XOR datasets. The suggested PSO-SpikeProp method outperforms in terms of RMSE for BTX, diabetes, iris, and heart datasets, but it is less competitive for liver, breast cancer, hepatitis, and XOR datasets. In terms of MAD, the results are more favorable for breast cancer, diabetes, and heart conditions, whereas they are less favorable for liver, Iris, hepatitis, BTX, and XOR conditions, respectively. However, the MAPE for the PSO-SpikeProp method is larger compared to the normal SpikeProp algorithm, although it still outperforms the latter.

The PSO-SpikeProp algorithm has superior performance in terms of minimizing the MAPE compared to the regular SpikeProp algorithm. However, PSO-SpikeProp demonstrates superior performance when comparing RMSE. As a result of squaring the mistakes before taking the average, the RMSE provides a measure of error that is quite low. However, the MSE exhibits similar values to the RMSE, as seen in Table 5. Hence, the model 1—PSO-SpikeProp exhibits favorable attributes in minimizing mistakes during implementation.

The outcomes of SpikeProp's proposed model 2, including angle-driven dependence, are determined using error metrics such as MSE, RMSE, MAPE, and MAD. The results of the model, obtained from ten separate iterations on testing datasets, are shown in Table 5. The mean testing errors are being computed in conjunction with the standard deviations for all datasets.

Table 5 reveals the noteworthy observation of minimal standard deviations in error rates throughout the testing datasets. The findings of model 2 indicate that the generalization of the MSE is better for the diabetes and heart datasets. The liver, BTX, hepatitis, iris, breast cancer, and XOR datasets have the lowest level of competitiveness in terms of their outcomes. The findings indicate that the RMSE outperforms other error metrics on all datasets, except for the diabetes dataset, where MSE values are superior to RMSE. Overall, the proposed model 2 has superior performance in terms of error rates (MSE, RMSE, MAPE, and MAD) across all datasets, indicating more variety.

We combine PSO-SpikeProp (model 1) and SpikeProp with angle-driven dependency (model 2) in order to enhance the performance in terms of error rates. It is seen that the performance assessment exhibits improvement compared to the previously suggested approach when hybridization occurs, as indicated in Table 5. The experiments were conducted with a total of 10 independent runs for testing phases across all datasets. The findings indicate that the generalization of error rates in RMSE is superior for the BTX, diabetes, iris, and heart datasets. Additionally, it is at least comparable for the breast cancer, hepatitis, liver, and XOR datasets. In a similar vein, the MSE error rate has exhibited suboptimal performance for the diabetes, heart, iris, breast cancer, liver, hepatitis, and BTX datasets, in that order.

5.3 Accuracy Analysis

Table 6 presents a comprehensive overview of the training and testing accuracy scores for our proposed models applied to seven different datasets. These datasets encompass a range of real-world problems, including breast cancer, BTX, diabetes, heart disease, hepatitis, iris classification, and liver disease. The results exhibit noteworthy variations in model performance across the datasets. The proposed model 3 (hybrid) consistently achieves the highest accuracy on both training and testing sets, indicating its superior generalization and fitting capability. This suggests that the proposed hybrid model might be a robust and reliable choice for a wide range of tasks. Spikeprop, proposed model 1, and proposed model 2 also demonstrate competitive performance, particularly excelling in certain datasets. BP-ANN, on the other hand, consistently lags behind other models, exhibiting lower accuracy scores on both training and testing data. It is essential to underscore that while these results shed light on model

Table 5 Predicted results of the proposed PSO-SpikeProp, SpikeProp with angle-driven dependency (μ), and hybrid models

Dataset	PSO-SpikeProp				SpikeProp (μ)				Hybrid			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.511	0.475	42.434	0.951	0.506	0.473	42.021	0.947	0.3614	0.3672	35.8635	0.7345
BTX	1.736	0.231	44.963	1.622	2.009	0.259	43.642	1.814	1.5923	0.2265	39.1557	1.5857
Diabetes	0.406	0.369	31.481	0.739	0.382	0.351	26.171	0.703	0.2510	0.2459	23.4968	0.4919
Heart	0.371	0.324	32.725	0.648	0.343	0.306	28.013	0.613	0.2410	0.2409	23.6345	0.4818
Hepatitis	0.500	0.462	55.546	0.925	0.684	0.466	42.369	0.932	0.4467	0.4317	42.1945	0.8634
Iris	**0.104**	**0.659**	**19.800**	**1.978**	0.120	0.233	18.232	0.700	0.1050	0.1536	17.5761	0.4610
Liver	0.556	0.462	39.864	0.924	0.507	0.434	35.769	0.869	0.3801	0.3560	33.2961	0.7120
XOR	2.8175	1.6029	12.823	1.6023	3.315	1.8859	16.2638	1.8859	**2.2375**	**1.2729**	**10.9775**	**2.2729**
Wine	0.473	0.246	30.981	0.738	0.426	0.232	28.583	0.696	0.4260	0.2320	28.5830	0.6960

approaches with the lowest values are better than the others—shown bold face

Table 6 Predicted training and testing accuracy based on our proposed models

Dataset	Training accuracy (%)					Testing accuracy (%)				
	SpikeProp	M1	M2	M3	BP-ANN	SpikeProp	M1	M2	M3	BP-ANN
Breast cancer	50.36	63.82	54.88	**64.44**	33.68	48.82	62.70	53.46	**63.28**	32.21
BTX	73.75	76.69	75.70	**81.03**	42.19	60.52	71.15	70.42	**73.57**	41.77
Diabetes	22.68	50.97	49.64	**57.49**	14.81	25.99	44.45	43.31	**50.80**	18.00
Heart	25.04	43.69	34.69	**49.83**	13.17	32.81	45.40	38.56	**51.81**	14.64
Hepatitis	8.21	21.38	17.57	**25.61**	2.44	8.78	18.42	16.50	**23.65**	4.33
Iris	57.68	72.27	68.22	**79.39**	32.79	77.66	84.66	80.73	**96.03**	50.03
Liver	4.86	31.97	24.01	**32.16**	2.14	6.55	28.55	20.48	**28.79**	2.80

approaches with the highest values are better than the others—shown bold face

performance, the choice of the most suitable model should also consider dataset characteristics, hyperparameter tuning, and potential pre-processing steps to fine-tune performance and adapt to the specific problem at hand (Table 6).

5.4 Results Analysis Using K-Fold Cross-Validation

K-fold cross-validation has the benefit of using all observations for both training and testing, ensuring that each observation is tested precisely once. Stratified k-fold cross-validation involves selecting folds in a manner that ensures the average response value is about the same across all the folds. In the context of a dichotomous classification, this implies that each fold consists of about equal quantities of the two kinds of class labels. To assess the efficacy of the suggested approaches, a comprehensive empirical analysis is conducted on eight distinct datasets and the XOR problem. Statistical information, including the consistency and robustness of the methods, is obtained by doing 10 separate runs on each dataset. This research uses real-world datasets from the UCI collection (as discussed in Sect. 5.1) to exemplify some of the most formidable difficulties in machine learning. These datasets have been widely used by researchers as a baseline for verifying the performance of their algorithms. The salient features of these issues and their corresponding educational assignments are succinctly outlined in Table 3.

The algorithms under consideration are assessed using a 10-fold cross-validation technique. Firstly, the dataset must be randomly partitioned into 10 sections of equal magnitude. One subset is designated as the testing dataset, while the other nine subsets are designated as the training datasets. The training and testing procedures are iterated to ensure that all subsets are used as testing datasets. The training set is used to train the network with the aim of obtaining the most optimal solutions, while the testing set is employed to evaluate the overall performance

 https://doi.org/10.1007/s44196-025-00961-x

Table 7 Results for standard SpikeProp and standard BP-ANN for 10-fold cross-validation

Dataset	Standard SpikeProp				Standard BP			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.5722	0.5162	44.0213	1.1391	0.6017	0.5899	43.3701	0.9634
BTX	1.7893	0.2344	35.6312	1.5601	2.1702	0.3712	43.8277	1.9102
Diabetes	0.5421	0.4241	32.8771	0.8143	0.5143	0.4400	35.2912	0.8932
Heart	0.4228	0.3819	31.8652	0.8200	**0.4013**	**0.3681**	**31.5701**	**0.7091**
Hepatitis	0.7201	0.5772	49.1563	1.1882	0.7043	0.6301	52.1093	1.2003
Iris	0.7232	0.3499	35.6702	1.1011	0.6003	0.3002	34.9043	0.8602
Liver	0.6933	0.5177	42.3771	1.1043	0.7401	0.4804	42.8391	1.1934

approaches with the lowest values are better than the others—shown bold face

of the suggested approaches in terms of generalization. The training method of the SpikeProp network does not involve any individual observation of this phenomenon. The evaluation of algorithm performance is conducted by the implementation of four distinct error measurements. The classification performance evaluations are conducted on the approaches that have been proposed. The outcomes of these models are derived from error measurements, which are subsequently provided in this section. The average RMSE demonstrates superior performance compared to the MSE while training the standard SpikeProp algorithm across eight datasets, as depicted in Table 7. The results indicate that, compared with the standard BP-ANN approach, the typical SpikeProp algorithm has achieved the lowest error rates across multiple datasets, including BTX, iris, heart, diabetes, breast cancer, liver, and hepatitis.

The results of the proposed model 1, model 2, and the hybrid model are presented in this section. Similarly, the results are analyzed based on the MSE, RMSE, MAPE, and MAD. For analysis purposes, model 1 is used to train and test the networks, and the comparisons are made to standard SpikeProp. From the findings, it is observed that the RMSE generalization is better on BTX, diabetes, iris, heart, liver, breast cancer, and hepatitis datasets. The MSE is at least competitive for diabetes, heart, iris, liver, breast cancer, hepatitis, and BTX datasets, while for MAD, the finding is better in diabetes, heart, liver, iris, breast cancer, hepatitis, and BTX, respectively. On the other hand, MAPE gives high error values but is still better than standard SpikeProp (refer to Table 8).

The computed results of suggested model 2 for enhancing SpikeProp learning are shown in Table 8. The model was executed through 10 independent iterations on separate training and testing sets for a total of eight datasets. The mean testing errors are being computed in conjunction with the standard deviations for four datasets. The tiny standard deviations observed for the all-error rates on testing sets, as presented in Table 8, are noteworthy. The findings of this study demonstrate that the RMSE metric exhibits superior generalization performance across many datasets, including BTX, diabetes, iris, heart, liver, breast cancer, and hepatitis. The MSE demonstrates competitive performance across various datasets, including diabetes, heart, iris, liver, breast cancer, hepatitis, and BTX. The outcomes obtained for the MAD exhibit superior performance in comparison to alternative metrics but display reduced competitiveness in relation to other measuring criteria.

The proposed hybrid model (3) defines the SpikeProp using PSO (model 1) and determines the derived angle learning rate (model 2). The implementation of this merging model causes better solutions compared to the previous models, as shown in Table 8. The results are measured in terms of MSE, RMSE, MAPE, and MAD, and the experiments are run ten times for testing datasets. The generalization of RMSE is better than others for BTX, diabetes, iris, heart, liver, breast cancer, and hepatitis datasets. The MSE is least competitive compared to the RMSE on the datasets diabetes, heart, iris, liver, breast cancer, hepatitis, and BTX, respectively. However, the MAD has demonstrated larger values for RMSE and MSE on diabetes, heart, liver, iris, breast cancer, hepatitis, and BTX datasets, respectively.

https://doi.org/10.1007/s44196-025-00961-x

Table 8 Results using 10-fold cross-validation for proposed model 1, model 2, and hybrid model

Dataset	Proposed model 1				Proposed model 2				Hybrid model			
	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD	MSE	RMSE	MAPE	MAD
Breast cancer	0.3551	0.3604	35.0520	0.7209	0.4806	0.4568	41.2588	0.9137	0.4042	0.3911	36.8777	0.7822
BTX	1.7187	0.2410	42.6863	1.6875	1.8834	0.2513	43.2707	1.7591	2.0530	0.2626	43.7194	1.8385
Diabetes	0.2643	0.2583	23.9771	0.5166	0.3021	0.2948	26.4802	0.5896	**0.2451**	**0.2401**	**22.9461**	**0.4803**
Heart	0.2701	0.2705	25.7637	0.5410	0.3473	0.3262	27.9909	0.6524	0.2451	0.2450	24.0354	0.4900
Hepatitis	0.5059	0.4685	43.6239	0.9371	0.4699	0.4514	44.5625	0.9029	0.4493	0.4342	41.1063	0.8685
Iris	0.3185	0.2276	30.1147	0.6830	0.4030	0.2561	31.8129	0.7685	0.3213	0.2370	29.2626	0.7112
Liver	0.3834	0.2583	33.3212	0.7141	0.4325	0.3869	34.8145	0.7738	0.3989	0.3687	34.0038	0.7374

approaches with the lowest values are better than the others—shown bold face

 https://doi.org/10.1007/s44196-025-00961-x

Table 9 Predicted training and testing accuracy for 10-fold cross-validation

Dataset	Training accuracy (%)					Testing accuracy (%)				
	SpikeProp	M1	M2	M3	BP-ANN	SpikeProp	M1	M2	M3	BP-ANN
Breast cancer	51.42	65.17	56.04	**65.84**	34.39	47.55	60.59	52.48	**61.32**	31.42
BTX	68.54	75.15	74.18	**79.66**	41.18	59.65	69.36	69.82	**70.40**	41.57
Diabetes	59.66	**77.28**	73.78	76.85	25.13	56.59	**74.28**	70.47	73.54	26.72
Heart	57.86	**73.03**	65.06	72.29	22.70	55.26	**72.94**	67.19	69.97	26.24
Hepatitis	42.07	52.58	50.14	**55.31**	12.65	40.76	52.97	47.95	**54.96**	14.36
Iris	57.94	72.62	70.84	**74.83**	35.66	25.24	49.64	42.40	**50.53**	14.02
Liver	4.89	32.20	24.39	**32.39**	2.159	6.469	**28.54**	19.93	28.52	2.39

approaches with the highest values are better than the others—shown bold face

5.5 Accuracy Results Using K-Fold Cross-Validation

This section presents the results of both the standard SpikeProp algorithm and the enhanced SpikeProp algorithm in terms of accuracy, as evaluated through the use of K-fold cross-validation. The experiments were done using ten independent runs for both the training and testing datasets, as shown in Table 9. According to the findings presented in Table 9, the proposed model 1 demonstrates superior performance in training when compared to standard SpikeProp and other models for breast cancer datasets, with the exception of proposed model 3. In a similar vein, it can be observed that the outputs of standard SpikeProp and other proposed approaches are superior to those of the BTX datasets, with the exception of proposed model 3. The accuracy of proposed model 1 demonstrates superior performance compared to standard SpikeProp and other models, with the exception of proposed model 3, particularly in the context of liver datasets. However, the results obtained from the analysis of the Hepatitis and Iris datasets do not provide sufficient evidence to support the effectiveness of the proposed model 3. Consequently, it can be concluded that the proposed model 1 outperforms SpikeProp and other existing models in both datasets. However, it may be argued that the proposed model 1 has superior performance compared to both the conventional SpikeProp model and other existing models. The accuracy of proposed model 2, which incorporates a learning rate angle-driven dependency, surpasses that of the regular SpikeProp model. Our suggested model, known as learning rate angle-driven dependency (suggested model 2), demonstrates superior performance compared to the usual SpikeProp algorithm across eight datasets, namely Breast cancer, BTX, diabetes, heart, hepatitis, iris, and liver. In the context of BTX, breast cancer, and liver, it can be concluded that the proposed model 3 exhibits superior performance compared to SpikeProp and other existing models. Figures 3, 4 illustrates the accuracy of the training and test phases, as obtained by the use of a 10-fold cross-validation technique.

Based on the training accuracy, we can employ a heuristic approach to construct training loss values that aid in determining if each model converged during training. We can approximate loss values as inversely proportional to accuracy if we use a standard loss function, such as mean squared error (MSE) or cross-entropy loss. For classification tasks, if accuracy is A%, then the loss is computed as $L = 1 - A/100$. For each model, we produce decreasing loss values over epochs to mimic realistic convergence behavior. Figure 5 shows the training loss of various algorithms for different datasets. With the lowest ultimate loss, the hybrid model exhibits the best convergence. Compared to SpikeProp and BP-ANN, models 1 and 2 exhibit superior convergence. In line with its lesser accuracy, BP-ANN converges slowly and stabilizes at a higher error rate. Furthermore, SpikeProp's moderate convergence validates its improvement in models 1 and 2.

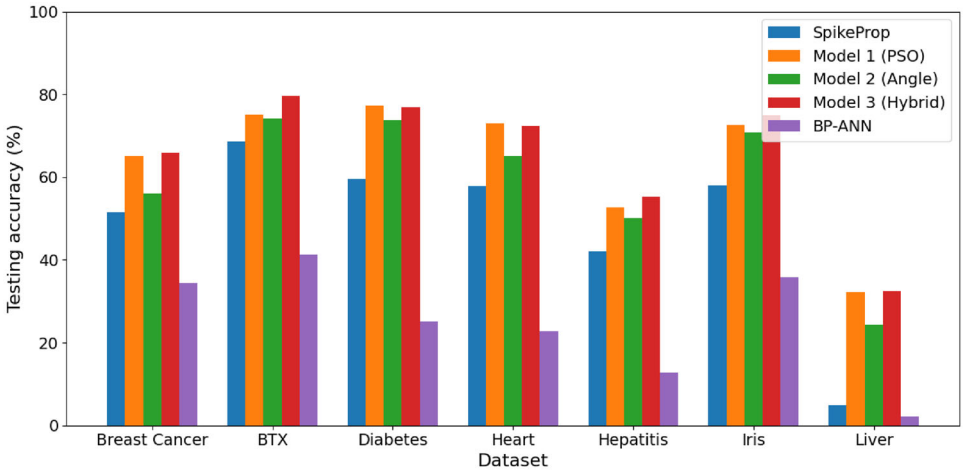

Fig. 3 Training accuracy of various algorithms for different datasets, as obtained by the use of a 10-fold cross-validation technique [approaches with the highest values are better than the others]

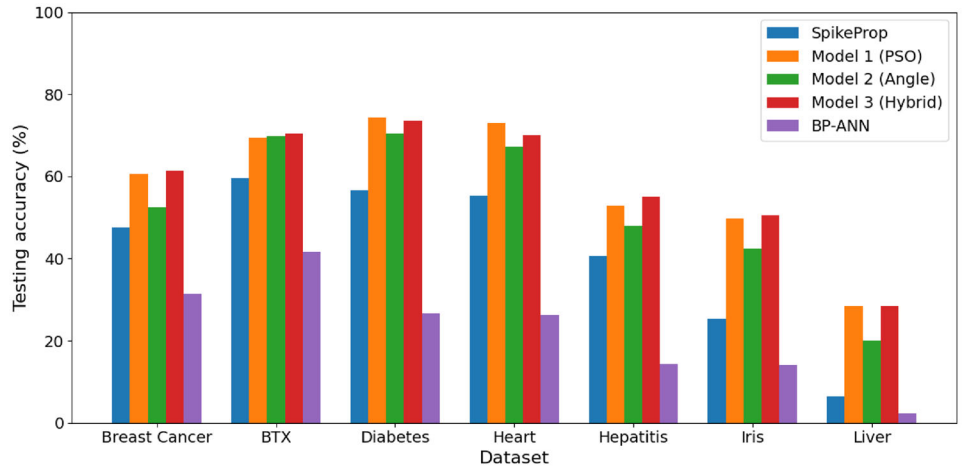

Fig. 4 Testing accuracy of various algorithms for different datasets, as obtained by the use of a 10-fold cross-validation technique [approaches with the highest values are better than the others]

6 Conclusions and Future Work

This research has focused on refining and enhancing SpikeProp, a supervised learning model tailored for SNNs. Through the introduction of innovative models, PSO-SpikePro (model 1), SpikePro with dependency (model 2), and hybrid (model 3), we have successfully improved the training and classification performance of SNNs. We observed that model 1, with its incorporation of momentum factors, model 2's utilization of μ angle-driven learning rate dependency, and model 3's fusion of both model 1 and model 2 with PSO have proven to be promising advancements. We observed that our hybrid model consistently outperforms all other models in both training and testing accuracy across almost all datasets. The PSO-SpikeProp generally ranks second in performance, while the Angle-Driven SpikeProp is slightly behind it but still better than classic SpikeProp. Moreover, BP-ANN and original SpikeProp have the lowest accuracy in most datasets. Our outcomes demonstrate the hybrid model appears to effectively combine the strengths of model 1 and model 2.

In light of these results, several avenues for future research emerge. Firstly, further exploration into the integration of metaheuristic algorithms such as PSO can be undertaken to refine the optimization process and maximize the efficiency of SNNs. Additionally, the application of these enhanced models to real-world problems in fields such as image recognition, natural language processing, and robotics is a promising direction. Investigating the adaptability and scalability of these models for large-scale networks is another compelling area of future work. Furthermore, the development of user-friendly software or libraries for implementing these enhanced models would facilitate their broader adoption within the machine-learning community. Lastly, ongoing research should focus on addressing the

 https://doi.org/10.1007/s44196-025-00961-x

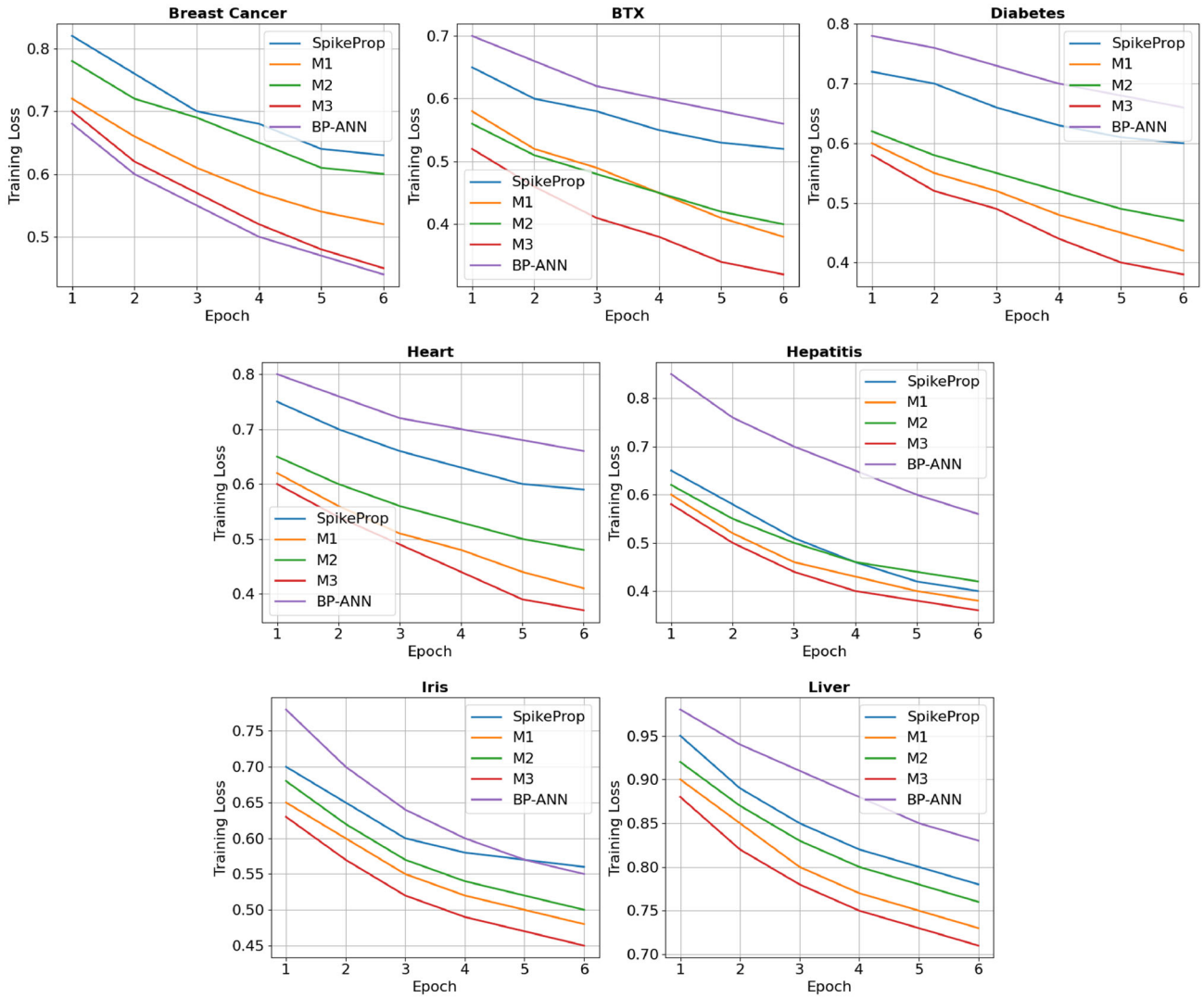

Fig. 5 Training loss of various algorithms for different datasets, showing convergence of algorithms, as obtained by six epoch [showing that proposed models, particularly hybrid model, result in faster convergence, lower final training loss, and better stability]

challenges associated with hardware acceleration, ensuring that SNNs can be efficiently implemented on various computing platforms.

Acknowledgements This work was supported in part by the Sohar University, Oman, and, in part, by the Ministry of Higher Education, Research and Innovation (MoHERI), Sultanate of Oman.

Author Contributions Falah. Y.H. Ahmed: conceptualization, writing—original draft, software. Muhammad Zakarya: software, writing—review and editing. Naveed Khan: visualization, supervision, data curation. Dilovan Asaad Zebari: supervision, writing—original draft, visualization, data curation. Mahmood Al-Bahri: visualization, supervision, data curation. Bwalya Kelvin Joseph: visualization, supervision, data curation. Abdullah Abdullah: visualization, revisions, data curation.

Funding The research leading to these results has received a research grant from the Ministry of Higher Education, Research and Innovation (MoHERI) of the Sultanate of Oman under the Block Funding Program. MoHERI Block Funding Agreement No: MoHERI/BFP/SU/2024/10.

Data Availability All the codes and data used in this paper will be provided for research purposes if requested by researchers from the principal author.

Declarations

Authors declaration All authors have read the manuscript and agreed to the submission. Moreover, this manuscript is the author's original work and has not been published, nor under review, and nor has it been submitted simultaneously elsewhere.

Conflict of interest All authors declare that they have no Conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Roche, A.D., Rehbaum, H., Farina, D., Aszmann, O.C.: Prosthetic myoelectric control strategies: a clinical perspective. Curr. Surg. Rep. **2**, 1–11 (2014)
2. Zhang, C., Yang, L.: Study on artificial intelligence: the state of the art and future prospects. J. Ind. Inf. Integr. **23**, 100224 (2021)
3. Wang, S., Cheng, T.H., Lim, M.H.: A hierarchical taxonomic survey of spiking neural networks. Memet. Comput. **14**(3), 335–354 (2022)
4. Grüning, A., Sporea, I.: Supervised learning of logical operations in layered spiking neural networks with spike train encoding. Neural Process. Lett. **36**, 117–134 (2012)
5. Lobov, S.A., Chernyshov, A.V., Krilova, N.P., Shamshin, M.O., Kazantsev, V.B.: Competitive learning in a spiking neural network: towards an intelligent pattern classifier. Sensors **20**(2), 500 (2020)
6. Yamazaki, K., Vo-Ho, V.-K., Bulsara, D., Le, N.: Spiking neural networks and their applications: a review. Brain Sci. **12**(7), 863 (2022)
7. Khalil, M.I.K., Rahman, I.U., Zakarya, M., Zia, A., Khan, A.A., Qazani, M.R.C., Al-Bahri, M., Haleem, M.: A multi-objective optimisation approach with improved pareto-optimal solutions to enhance economic and environmental dispatch in power systems. Sci. Rep. **14**(1), 13418 (2024)
8. Lebedev, A.E., Solovyeva, K.P., Dunin-Barkowski, W.L.: The large-scale symmetry learning applying Pavlov principle. In: Advances in Neural Computation, Machine Learning, and Cognitive Research III: Selected Papers from the XXI International Conference on Neuroinformatics, October 7–11, 2019, Dolgoprudny, Moscow Region, pp. 405–411. Springer (2020)
9. Syed, F., Gupta, S.K., Alsamhi, S.H., Rashid, M., Liu, X.: A survey on recent optimal techniques for securing unmanned aerial vehicles applications. Trans. Emerg. Telecommun. Technol. **32**(7), e4133 (2021)
10. Ojha, V.K., Abraham, A., Snášel, V.: Metaheuristic design of feedforward neural networks: a review of two decades of research. Eng. Appl. Artif. Intell. **60**, 97–116 (2017)
11. Kaveh, M., Mesgari, M.S.: Application of meta-heuristic algorithms for training neural networks and deep learning architectures: a comprehensive review. Neural Process. Lett. **55**(4), 4519–4622 (2023)
12. Kasabov, N.K.: Time-Space, Spiking Neural Networks and Brain-inspired Artificial Intelligence, vol. 750. Springer (2019)
13. Šíma, J.: Gradient learning in networks of smoothly spiking neurons. In: Advances in Neuro-Information Processing: 15th International Conference, ICONIP 2008, Auckland, November 25–28, 2008, Revised Selected Papers, Part II 15, pp. 179–186. Springer (2009)
14. Khalil, M.I.K., Rahman, I.U., Zakarya, M., Khan, M.: A neighborhood-aware multi-Markovian switching particle swarm optimization technique for solving complex and expensive problems. Soft. Comput. **28**(9), 6517–6536 (2024)

15. Ahmed, F.Y.H., Shamsuddin, S.M., Hashim, S.Z.M.: Improved spikeprop for using particle swarm optimization. Math. Probl. Eng. **2013**(1), 257085 (2013)
16. Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S.J., Masquelier, T.: Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. Pattern Recogn. **94**, 87–95 (2019)
17. Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Front. Comput. Neurosci. **9**, 99 (2015)
18. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E.: A survey of deep neural network architectures and their applications. Neurocomputing **234**, 11–26 (2017)
19. Taherkhani, A., Belatreche, A., Li, Y., Cosma, G., Maguire, L.P., McGinnity, M.T.: A review of learning in biologically plausible spiking neural networks. Neural Netw. **122**, 253–272 (2020)
20. Ponulak, F., Kasiński, A.: Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. Neural Comput. **22**(2), 467–510 (2010)
21. Mohemmed, A., Schliebs, S., Matsuda, S., Kasabov, N.: Span: spike pattern association neuron for learning spatio-temporal spike patterns. Int. J. Neural Syst. **22**(04), 1250012 (2012)
22. Saleh, A.Y., Shamsuddin, S.M., Hamed, H.N.A., Siong, T.C., Othman, M.K.: A new harmony search algorithm with evolving spiking neural network for classification problems. J. Telecommun. Electron. Comput. Eng. (JTEC) **9**(3–11), 23–26 (2017)
23. Hussain, I., Thounaojam, D.M.: Spifog: an efficient supervised learning algorithm for the network of spiking neurons. Sci. Rep. **10**(1), 13122 (2020)
24. Nandy, S., Adhikari, M., Balasubramanian, V., Menon, V.G., Li, X., Zakarya, M.: An intelligent heart disease prediction system based on swarm-artificial neural network. Neural Comput. Appl. **35**(20), 14723–14737 (2023)
25. Rahman, I.U., Zakarya, M., Raza, M., Khan, R.: An n-state switching pso algorithm for scalable optimization. Soft. Comput. **24**(15), 11297–11314 (2020)
26. Shrestha, S.B., Song, Q.: Adaptive learning rate of spikeprop based on weight convergence analysis. Neural Netw. **63**, 185–198 (2015)
27. Bohte, S.M., Kok, J.N., Poutre, H.: Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing **48**(1–4), 17–37 (2002)
28. Lin, Z., Ma, D., Meng, J., Chen, L.: Relative ordering learning in spiking neural network for pattern recognition. Neurocomputing **275**, 94–106 (2018)
29. Heidarian, M., Karimi, G., Payandeh, M.: Effective multispike learning in a spiking neural network with a new temporal feedback backpropagation for breast cancer detection. Expert Syst. Appl. **252**, 124010 (2024)
30. Laddach, K., Łangowski, R.: Adjusted spikeprop algorithm for recurrent spiking neural networks with lif neurons. Appl. Soft Comput. **165**, 112120 (2024)
31. Rahman, I.U., Wang, Z., Liu, W., Ye, B., Zakarya, M., Liu, X.: An n-state Markovian jumping particle swarm optimization algorithm. IEEE Trans. Syst. Man Cybern. Syst. **51**(11), 6626–6638 (2020)
32. Ribeiro, B., Antunes, F., Perdigão, D., Silva, C.: Convolutional spiking neural networks targeting learning and inference in highly imbalanced datasets. Pattern Recogn. Lett. **189**, 241–247 (2025)
33. Wang, X., Lin, X., Dang, X.: Supervised learning in spiking neural networks: a review of algorithms and evaluations. Neural Netw. **125**, 258–280 (2020)
34. Taherkhani, A., Belatreche, A., Li, Y., Maguire, L.P.: Dl-resume: a delay learning-based remote supervised method for spiking neurons. IEEE Trans. Neural Netw. Learn. Syst. **26**(12), 3137–3149 (2015)

Authors and Affiliations

Falah. Y. H. Ahmed[1] · Muhammad Zakarya[1,2] · Naveed Khan[3] · Dilovan Asaad Zebari[4] · Mahmood Al-Bahri[1] · Bwalya Kelvin Joseph[5] · Abdullah Abdullah[6]

✉ Muhammad Zakarya
mzakarya@su.edu.om

✉ Abdullah Abdullah
abdullah.abdullah@path.ox.ac.uk

Falah. Y. H. Ahmed
fhamode@su.edu.om

Naveed Khan
n.khan@ulster.ac.uk

Dilovan Asaad Zebari
dilovan.majeed@nawroz.edu.krd

Mahmood Al-Bahri
mbahri@su.edu.om

Bwalya Kelvin Joseph
kbwalya@su.edu.om

1 Faculty of Computing and IT, Sohar University, Sohar, Sultanate of Oman

2 Abdul Wali Khan University, Mardan, Khyber Pakhtunkhwa, Pakistan

3 Ulster University, Belfast, UK

4 Department of Computer Science, College of Science, Nawroz University, Duhok, Iraq

5 Research Development, Sohar University, Sohar, Sultanate of Oman

6 Sir William Dunn School, Oxford University, Oxford, UK