Evolutionary Intelligence
https://doi.org/10.1007/s12065-024-00983-y

RESEARCH PAPER

A novel classification technique using a biologically plausible spiking neuron and noisy synapses

Irshed Hussain[1] · Dalton Meitei Thounaojam[2]

Received: 19 April 2024 / Revised: 29 August 2024 / Accepted: 3 September 2024

Abstract
The organic evidence from neuroscience proves that precise spike times are used for information exchange between two biological neurons rather than the firing rates. One of the prominent reasons, along with energy and computational efficiency, is that spiking neural networks (SNNs) are getting more attention nowadays. The spiking neurons in SNN mimic the biological neuron more than its predecessors. Despite the few efficient supervised learning algorithms for SNN, only some investigated the biological properties such as axonal noise, random synaptic delays, spontaneous spike-firing, and random switching of the gamma-aminobutyric acid (GABA)-switch. The aforementioned properties are essential for making spiking neurons more biologically realistic, which is one of the major strengths of SNN. The GABA switch decides the most crucial activity, whether a neuron will be excitatory or inhibitory. This paper proposes a novel and efficient approach to handle non-linear patterns using a single leaky-integrate-and-fire (LIF) spiking neuron connected with many noisy synapses with random synaptic delays. In addition, the spontaneous firing of a neuron and random switching of signs in synaptic weights having equal probability akin to GABA-switch are efficiently implemented. Moreover, a hybrid kernel is proposed as the synapse model to cope with the noise properly, which makes the synapse model more efficient. The error-tuning is carried out using the elitist floating-point genetic algorithm. Four datasets were used for benchmarking, and experimentally, better results were obtained than state-of-the-art methods.

Keywords Axonal noise · Random synaptic delay · Spontaneous spikes · SNN

1 Introduction

There has been a curiosity among artificial intelligence (AI) community researchers to mimic the most complex human brain artificially for decades. Although it could not reach a satisfactory level, the AI community is gradually moving towards the goal by developing bio-inspired computational systems such as the spiking neural network (SNN). Due to the energy-efficiency [1–3], dynamic capability [3], computational-efficiency [4], and biological plausibility [1, 2], SNN is grabbing the attention of researchers nowadays. Note that the organic evidence from neuroscience proved the mechanism of exchanging information between two biological neurons regarding precise spike-timings [5, 6]. The spiking neurons in SNN also use precise spike-timings for communication, which are discrete events rather than the traditional continuous firing rates (which is still in use with the artificial neural network (ANN) [7] which is considered as the second generation of ANN). On the other hand, the third-generation network SNN [7] is emerging in the neuroscience community. It is efficiently implemented in the neuromorphic hardware, which consumes less energy to carry out complex specific tasks. The neuromorphic chips such as Intel Loihi [8], IBM TrueNorth [9] uses SNN as the intelligent model.

Mathematically, SNN can be defined as a network of a finite set consisting of spiking neurons S_{neu}, a set

Irshed Hussain and Dalton Meitei Thounaojam have equally contributed to this research work.

✉ Irshed Hussain
irshedhussain@soa.ac.in

Dalton Meitei Thounaojam
dalton@cse.nits.ac.in

[1] Computer Science and Information Technology, Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar, Odisha 751030, India

[2] Computer Science and Engineering, National Institute of Technology Silchar, Silchar, Assam 788010, India

$Q \subseteq S_{neu} \times S_{neu}$ of synapses having connection strength $W_{i,j} \in \mathbb{R}$, each synapse $< i, j > \in Q$ has a response function $\xi_{i,j} : \mathbb{R}^+ \rightarrow \mathbb{R}$ (where $\mathbb{R}^+ := q \in \mathbb{R}^+ : q \geq 0$), and for each $S_{neu}, z \in Q$ there is a threshold function $V_{th} : \mathbb{R}^+ \rightarrow \mathbb{R}$. In SNN, the spiking neurons do not fire at every cycle of propagation; instead, they fire only when its cell membrane potential reaches a finite threshold value [1]. There are two types of neurons in SNN, pre-synaptic neurons and post-synaptic neurons, when the categorisation is done based on information propagation. The information sender neuron is called the pre-synaptic neuron, and the information receiver neuron is called the post-synaptic neuron. Each pre-synaptic neuron is connected with its corresponding post-synaptic neuron through various synapses. When the input stimuli received from pre-synaptic neurons by a post-synaptic neuron changes its cell membrane potential, called the post-synaptic potential (PSP), up to the defined threshold, then the neuron fires a spike. Note that after firing a spike, a neuron is bound to follow the absolute refractory time to fire another spike. During the absolute refractory period, no neuron can issue spikes [1]. Initially, the cell membrane remains at resting potential when there are no input stimuli to receive.

In order to work with the network of spiking neurons, the three most crucial parts to be followed are mapping the real-valued continuous features into the precise spike-timings (i.e., information encoding), selecting a neuron model and a synapse model, and the error-tuning mechanism to get optimal results. Although many information encoding schemes exist, the most popularly used one is the population encoding [10, 11]. Every spiking neuron model such as leaky-integrate-and-fire (LIF) [12, 13] (an improvisation in the dynamics of the integrate-and-fire [14]), Hodgkin-Huxley [15], spike response model (SRM) [1, 16], Izhikevich model [17], etc., has its own merits and demerits. The neuroscience community would like to use a neuron model, which is the most biologically plausible, although it is complex or computationally costly. On the other hand, computer engineers would like to use a neuron model, which is computationally inexpensive, although it is biologically less plausible. However, there is a need to maintain the proper balance between biological plausibility and computational cost for a pattern classification problem, which is very challenging.

SNN, being less explored than its ancestor ANN, requires a generalised framework, proper implementation of biological properties, and a supervised learning algorithm. In addition, it is also less explored than deep neural networks due to the need for efficient learning algorithms. Deep neural networks are widely used in the image field, such as image forgery detection as discussed in [18]. SNN can also be applied for image recognition if an efficient learning mechanism exists. Although the few supervised learning algorithms for SNN are both computationally efficient and biologically plausible, none of the algorithms investigates and implement all the possible properties of biological neurons, such as the presence of axonal noise [19], random synaptic delays [20], the spontaneous firing of spikes [21, 22], and rightly balanced gamma-aminobutyric acid (GABA)-switch [23, 24] together, while tuning the overall network error. The axonal noises are present for various reasons, such as other neighbouring biological activities. The delay in information processing is mainly due to the length of the axon cable connected between the neurons. The spontaneous firing and its effect on the spiking activity is discussed in [21] and [22].

Various supervised learning algorithms to train SNN have been developed with the flow of time, although only some of the algorithms are satisfactory. The first gradient-based popular supervised learning algorithm to train SNN is developed by Bohte et al. called SpikeProp [10]. It uses the population encoding scheme combined with the concept of time-to-first-spike firing, i.e., in every neuron, the first firing time is more important than the latter. The error direction was investigated in SpikeProp by finding the slope. Although SpikeProp succeeded to some extent, it fails to prop up synaptic weights if a post-synaptic neuron no longer fires a spike after receiving the input stimuli. To improve the SpikeProp algorithm, it was investigated further in [25–30]. However, the main problem of SpikeProp being gradient-based learning persists that is the stagnation at the local minimum.

Therefore, developing a supervised learning algorithm for SNN shifted in a different direction based on the plasticity concept of learning as the biological neuron learns. In [31] and [32], the spike time-dependent plasticity (STDP) based supervised learning algorithm is proposed. Various ways exist where STDP is used as the learning algorithm. The STDP approach is examined in [33] considering the hardware-friendly approach. In [34], SNN is combined with deep learning methods to detect the images of the weather, where STDP is used as the learning algorithm. Nevertheless, STDP works better in unsupervised learning and is not generally considered a fully functional learning algorithm. It just changes the sign of synaptic weights based on the pre-synaptic spike timings. Other supervised algorithms such as [11, 35] also use STDP but in a different manner.

Wade et al. [36] proposed the algorithm SWAT that also uses STDP for training. However, this algorithm is computationally costly due to many hidden neurons, making the synaptic load very high. On the other hand, some supervised learning algorithms such as SEFRON [37] reduce the computational cost and explore the power of spiking neurons by using a single spiking neuron to classify non-linear patterns. However, from the literature, it is found that the right balance between biological plausibility and computational cost poses a challenging task and is still an unsolved problem. In [38–41], the detailed review of different supervised learning algorithms developed to train SNN using various approaches is discussed lucidly.

The ability of metaheuristics to work without too much mathematical complexity is an advantage of using it to optimise synaptic elements of an SNN. Although many metaheuristic approaches are proposed for SNN based on particle swarm optimisation (PSO), [42], and differential evolution (DE) [43], the most efficient and rightly balanced metaheuristic-based supervised learning algorithm is proposed in [44] called SpiFoG. The SpiFoG uses the elitist floating-point genetic algorithm to optimise error, and the synapse model is a combination of excitatory and inhibitory neurons having random synaptic delays, which are also fine-tuned along with the synaptic weights efficiently. However, synaptic load and initialisation of synaptic weights and delays of SpiFoG can be further improved to enhance the performance. Another metaheuristic-based learning method using a single LIF neuron is proposed in [45]. The mentionable merit of WOLIF is its ability to learn using only readout neurons and its few network parameters in terms of synaptic load and input neurons. Note that WOLIF also works without hidden layer(s). Although WOLIF has a low computational cost, biological plausibility regarding synaptic elements is compromised.

This research focuses mainly on biological plausibility and makes the synapse model more efficient for classification tasks. We propose an efficient synapse model capable of dealing with the non-linear pattern efficiently and coping with the noise. The population encoding [10] transforms the real-valued features into temporal spikes. The weighted presynaptic spikes and the spike time of the only bias neuron are passed through the noisy synapse to the single readout neuron, the LIF neuron. Also, we introduced a hybrid kernel, which is used to cope with the axonal noise. Note that spontaneous spike firing activity is also implemented efficiently since its implementation does not hamper the overall complexity of the model, as it is also an essential component in the biological neuron. Moreover, we implemented a GABA switch efficiently, making it more random and having an equal probability of sign change. Finally, the error produced by the mean squared error (MSE) loss function is fine-tuned using a floating-point or real-coded genetic algorithm, which uses elitism and hybrid crossover. The single point crossover and the crossover mechanism discussed in [46] are used together to generate newly optimised set-off solutions. The uniform mutation technique adds some diversity to the search space.

The major contributions include:

1. We propose a more efficient and robust synapse model capable of coping with the axonal noise and spontaneous spikes. It improves the biological plausibility of a spiking neuron.
2. We introduced a hybrid kernel to handle noisy inputs efficiently without hampering the overall complexity of the model since, after using the hybrid kernel, the total simulation time did not increase, which is crucial for complexity.
3. We have efficiently implemented inhibitory and excitatory neurons to have the same probability (50%–50%) of sign-change using the GABA switch.

2 Development of pre-learning phase

The elements of the pre-learning phase include information encoding, proper implementation of the GABA switch and noisy synapse, and the one-to-one connection of the noisy synapses with the LIF readout neuron and its dynamics. Here, in this phase, the readout neuron outputs the predicted class to be optimized in the learning phase to match the desired class.

2.1 Encoding of information

The transformation of real-valued continuous features into temporal spikes is carried out using the population encoding [10] method. According to this method, real-valued continuous features $x_M^{(P)} \in \mathbb{R}$ (where M represents the number of real-valued continuous features for P number of patterns) can be segregated into several discrete temporal values $f_F^{(P)}$ (where $F = 1, 2, 3, ..., M \times M_{en}$) with the help of M_{en} overlapping Gaussian curves. The response values where a real-valued continuous feature intersects the Gaussian curves are computed from (1).

$$\mathcal{G}_F^P = exp\left(- \frac{(x_M^l - \mu_k)^2}{2\sigma^2} \right) \tag{1}$$

where $l = 1, 2, 3, ..., P$, $\mathcal{G}_F^P$ is the response given by the Gaussian curves for all P number of patterns, μ_k is the mean value of the $k = 1$ to M_{en} overlapping Gaussian curves and it is computed from (2). The standard deviation of the overlapping Gaussian curves is denoted by σ, calculated from (3).

$$\mu_k = \left(\frac{2k - 3}{2} \right) \times \left(\frac{1}{M_{en} - 2} \right) \tag{2}$$

$$\sigma = \frac{1}{\beta} \times \left(\frac{1}{M_{en} - 2} \right) \tag{3}$$

where β amount of overlapping is within the Gaussian curves, the value is set to 1.5 for better overlap between two curves.

Finally, after getting all response values, $x_M^{(P)}$ is converted into discrete temporal spikes $f_F^{(P)}$ using (4).

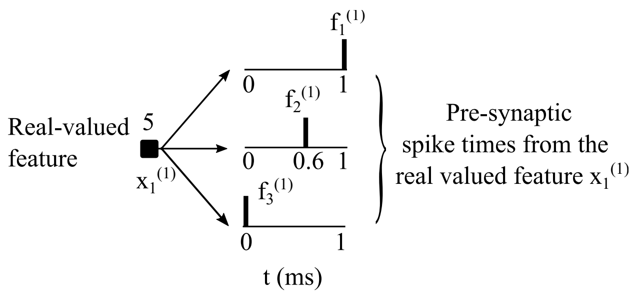

Fig. 1 Mapping of a continuous real-valued feature $x_1^{(1)}$=5 (taken from the WBC data set) into three discrete temporal spikes such as $f_1^{(1)}$=1 ms, $f_2^{(1)}$=0.6 ms, $f_3^{(1)}$=0 ms. These are part of pre-synaptic spikes

$$f_F^{(P)} = T_{ref} \times [1 - \mathcal{G}_F^P] \tag{4}$$

where encoding interval $\Delta T = [0, T_{ref}]$ ms and T_{ref} is the upper bound of spike times, i.e., 1 ms.

Figure 1 shows the encoding procedure in case of a real-valued feature 5 taken from the benchmarked data set WBC. It is observed that for the first pattern and first feature of the WBC dataset, $x_1^{(1)}$=5 is converted into three discrete temporal spikes $f_1^{(1)}$=1 ms, $f_2^{(1)}$=0.6 ms, $f_3^{(1)}$=0 ms and these are the value of pre-synaptic spike time for the first pattern and first feature of the WBC data set.

In the case of all binary classification problems used in this research, such as WBC, ION, LIV, and PID (discussed in Sect. 4), the desired output temporal spikes are coded as $S_{des}^{(C_1)} = 1$ ms for class 1, and $S_{des}^{(C_2)} = 2$ ms for class 2. The time step δt is set to 0.1 ms to keep the classes well separated from each other.

2.2 Proposed synapse model

This research emphasises the synapse model because mostly unexplored properties such as axonal noise, random synaptic delay, spontaneous firing, and GABA-switch behaviour are explored using the static synapse model. This section discusses the impact of the aforementioned biological properties and the proposed synapse model in detail. The primary impact of properties such as axonal noise, random synaptic delay, and spontaneous firing upon improving the proposed model's performance is that of mimicking the massive parallel human brain to some extent. Properly using these parameters can make a model more robust and inculcate the ability to handle highly non-linear data. Figure 2a and b shows the behaviour of two kernels $K_1(t)$ and $K_2(t)$ concerning the total simulation time T=2.0 ms at each time step δt having value 0.1 ms when the kernels mentioned above are multiplied with a positive random synaptic weight. The pre-synaptic spike times add a constant spontaneous spike time of 0.5 ms. The spontaneous spike time is added only when a random number $r \in [0, 1] > 0.5$. The definition of kernel $K_1(t)$ and $K_2(t)$ is given in (5) and (6) respectively. The proposed kernel function $K_2(t)$ is derived from the radial basis kernel function (RBF) [47], where the mean term is neglected. The standard deviation value is replaced with the cell membrane time constant

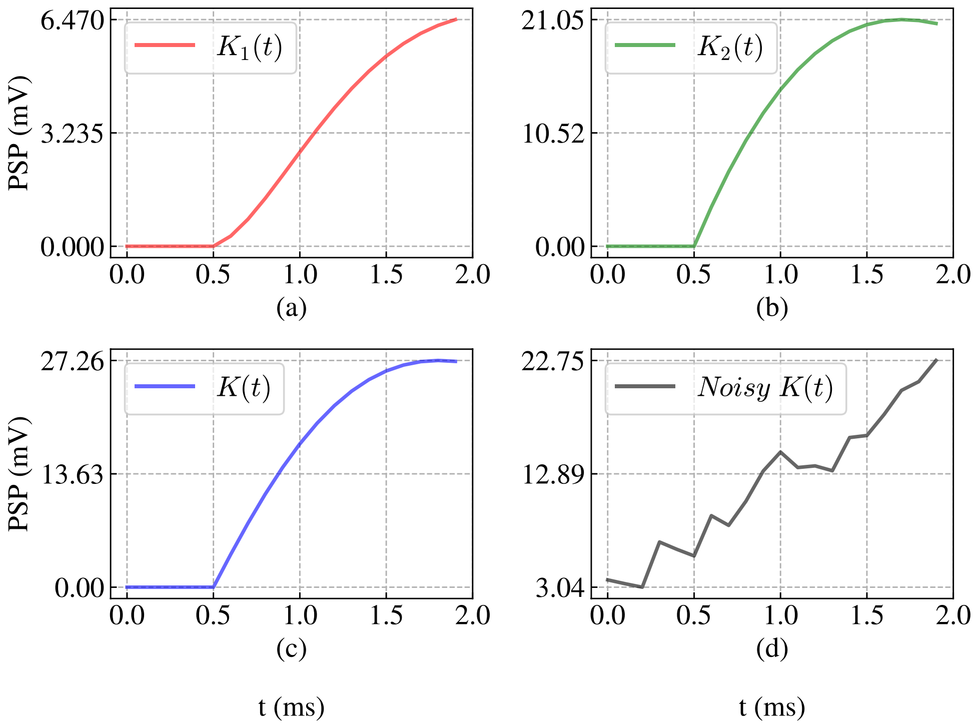

Fig. 2 Behavior of excitatory PSPs (positive synaptic weights are multiplied with PSPs) with **a** double decaying kernel function $K_1(t)$ within T=2 ms **b** RBF like kernel function $K_2(t)$ within T=2 ms **c** hybrid kernel function $K(t) = K_1(t) + K_2(t)$ within the total simulation time T=2 ms (d) noisy $K(t)$ (added Gaussian noise)

τ_{mem}, termed RBF-like kernel function. The definition of RBF kernel is given in (7).

$$K_1(t) = \left[exp\left(-\frac{t}{\tau_{mem}}\right) - exp\left(-\frac{t}{\tau_{syn}}\right)\right]\mathcal{H}(t) \tag{5}$$

where τ_{mem} and τ_{syn} are two-time constants called membrane time constant and synaptic time constant, respectively. The value of τ_{mem} is calculated from (8). The function $\mathcal{H}(t)$ is the Heaviside function given in (9).

$$K_2(t) = \left[exp\left(-\frac{t^2}{2\tau^2_{mem}}\right)\right]\mathcal{H}(t) \tag{6}$$

$$K_{rbf}(t) = \left[exp\left(-\frac{(t-\mu_{rbf})^2}{2\sigma^2_{rbf}}\right)\right] \tag{7}$$

where μ_{rbf} is the mean of the distribution, and σ_{rbf} is the standard deviation of the distribution.

The kernel $K(t)$ is the addition of kernel $K_1(t)$ and kernel $K_2(t)$ which is shown in the Fig. 2c. It can be observed from Fig. 2a and b that the kernel $K_1(t)$ is slowly increasing its PSP than that of kernel $K_2(t)$. Therefore, $K_1(t)$ will give late temporal spikes as compared to $K_2(t)$, which increases the computational cost of the LIF neuron since there is a possibility of not firing within the value of T. The parameter T is taken as 2 ms for all datasets. Note that inputs are not supplied to the kernels as they are a demonstration of T. Since the positive synaptic weights are multiplied here, the PSPs are rising curve. Now, if we observe the kernel $K(t)$, we find that its PSP is increasing faster than that of both the kernels $K_1(t)$ and $K_2(t)$ which will help towards the computational cost. In this case, non-firing activity within the value of T is much less, and learning can also be efficient with early temporal spikes. To make the synapse robust against noise, the noisy $K(t)$ kernel shown in Fig. 2d is used in this research.

The noise added with the kernel $K(t)$ is the Gaussian noise having mean 0 and standard deviation 1. The PSP values increasing capability of the noisy $K(t)$ is slightly lower than that of $K(t)$.

Note that because of the positive synaptic weight, the pre-synaptic neuron, in this case, is excitatory. The value of τ_{mem} used in (6) is calculated as given in the (8).

$$\tau_{mem} = R_{mem} \times C_{mem} \tag{8}$$

where R_{mem} is the cell membrane resistance, and C_{mem} is the capacitance of the cell membrane. The values of R_{mem}, and C_{mem} are set to 100 MΩ, and 0.11 pF respectively. Hence, the value of τ_{mem} becomes 1.1 ms (from (8)). The value of τ_{syn} is kept at half of the τ_{mem}, i.e., 0.55 ms.

$$\mathcal{H}(t) = \begin{cases} 1, \text{ if } t > 0 \\ 0, \text{ otherwise} \end{cases} \tag{9}$$

Fig. 3a, b, c, and d shows the behaviour of the same kernel functions as discussed earlier in Fig. 2 but multiplied with a negative random synaptic weight. It can be observed here that the PSPs are decaying since negative synaptic weight is multiplied by the PSPs. However, kernel $K(t)$ attains lower negative values compared to $K_1(t)$ and $K_2(t)$ which makes $K(t)$ more computationally efficient. Note that because of the negative synaptic weight, the pre-synaptic neuron, in this case, is inhibitory. We allowed 50% inhibitory and 50% excitatory pre-synaptic neurons.

The presence of synaptic delays and the axonal noise can affect the actual predicted spike times poorly collected from the output of the readout LIF neuron, illustrated in Fig. 4. It can be observed that the spike S_{pre} having spike time 0.1 ms is passed through the weighted noisy synapse before feeding as the input to the LIF neuron, which is supposed to fire at 0.7 ms that, is the predicted spike time S_{out}. However, due to the synaptic delay and noise, there is a possibility of S_{out} to fire too early ($S^{(a)}_{out}$=0.4 ms), early ($S^{(b)}_{out}$ = 0.6 ms), late ($S^{(c)}_{out}$ = 0.8 ms), or too late ($S^{(d)}_{out}$ = 1 ms). The scenario where hybrid kernel $K(t)$ plays a crucial role in maintaining a trade-off between axonal noise and synaptic delay. Finally, synaptic current $I_{syn}(t)$ at time t is calculated from (10) using the noisy hybrid kernel $K(t)$.

$$I_{syn}(t) = \sum_{i=1}^{F} \mathbf{W}_i \times K(t - S^{(i)}_{pre}) \tag{10}$$

where W_i is the synaptic weight of the i^{th} synapse (the values of W are within the interval [-1, 1] for uniform distribution **U**), and $S^{(i)}_{pre}$ is the combined input which is defined in (11) for a single pattern. The definition of the kernel $K(t - S^{(i)}_{pre})$ is given in (12).

$$S_{pre} = (f_F + t_{del} + a_{noise} + I_{spon}) \tag{11}$$

where t_{del}, a_{noise}, and I_{spon} are the synaptic delay time, axonal noise, and spontaneous spike firing time, respectively. The I_{spon} value is 0.5 ms. The values of t_{del}, and a_{noise} are within the interval [0, 1] for both uniform distribution **U**, and normal distribution **N** respectively, including the values 0 and 1.

$$K(t) = K_1(t) + K_2(t) \tag{12}$$

where $K_1(t)$ and $K_2(t)$ are the double decaying kernel function and RBF-like kernel, respectively.

$$S_{out} = \left\{t | V_{mem}(t) \geq V_{th}\right\} \tag{13}$$

The predicted spikes from the LIF neuron can be characterized using (13). The whole implementation procedure is summarized with the help of a block diagram given in Fig. 5.

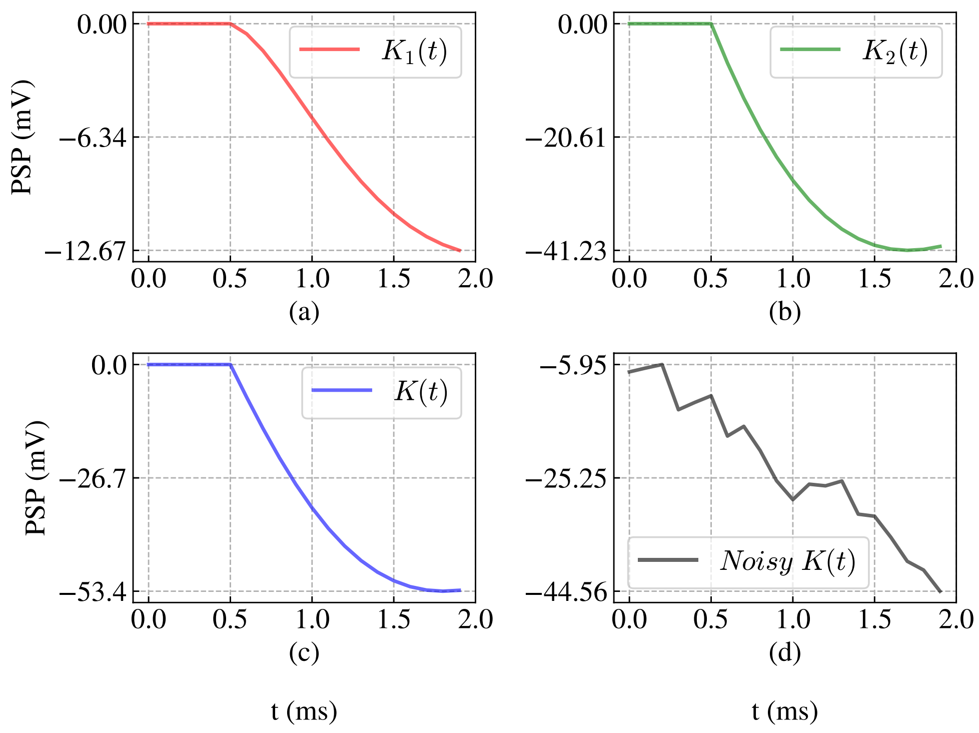

Fig. 3 Behavior of inhibitory PSPs (negative synaptic weights are multiplied with PSPs) with the same kernel functions as in Fig. 2a, b, c, and d

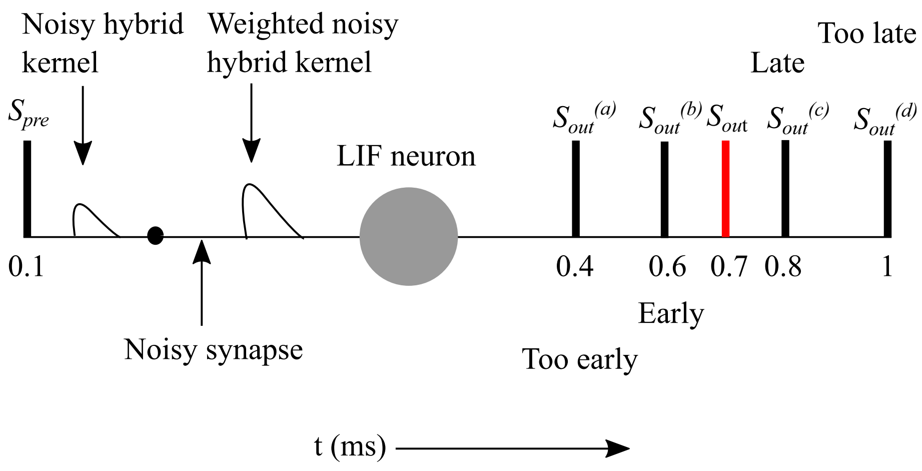

Fig. 4 Effect on actual spike times (late firing and early firing) due to synaptic delays and axonal noise

2.3 Dynamics of the readout neuron

The responsibility of this step is to produce the predicted spikes after getting the input current $I_{syn}(t)$ at time t from the synapse model. In this research, the computationally most straightforward LIF neuron is used. An electrical RC circuit generally describes the properties of the cell membrane of a LIF neuron. For the only post-synaptic neuron j, the electrical activity that changes the PSP of the neuron upon getting the input stimuli from a pre-synaptic neuron i is given in (14).

$$\tau_{mem} \frac{\delta V^{(j)}_{mem}(t)}{\delta t} = -V^{(j)}_{mem}(t-1) + I_{syn}(t) \times R_{mem} \tag{14}$$

where $j = 1$ (since single LIF neuron is used), $V^{(j)}_{mem}(t-1)$ is the PSP of the cell membrane of neuron j at time $(t-1)$. From (14), the change in PSP at time t is calculated, and the final value of the PSP for neuron j is given by (15).

$$V^{(j)}_{mem}(t) = V^{(j)}_{mem}(t-1) + \delta V^{(j)}_{mem}(t) \tag{15}$$

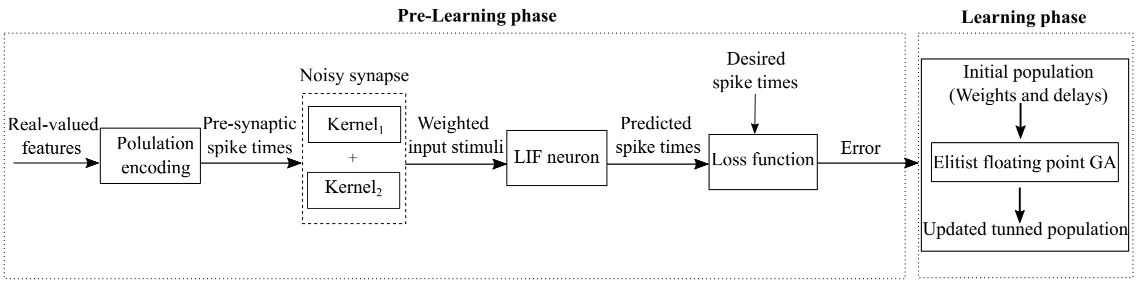

Fig. 5 Block diagram illustrating the phases involved in the pre-learning and the learning phase. The pre-learning phase is responsible for yielding the untrained predicted temporal spikes for the classification of non-linear patterns, and the learning phase tunes the erroneous predicted temporal spikes to improve the performance of the model

When $V_{mem}^{(j)}(t)$ reaches a threshold value V_{th}, neuron j fires a spike at time t and the recorded firing time is the spike time generated by neuron j. The V_{th} value is 1 mV. Since it is found from the literature that the first temporal spike always carries the most relevant information than the lateral spikes in biological neurons [48], we neglected the lateral spike times. We did not use the absolute refractory period.

Figure 6a and b shows the tracing of two different PSPs. In Fig. 6a, a spike is fired at 2 ms, which is the predicted spike time (it used the trained weights and delays) in the case of the WBC data set for a single pattern. It is found that this 2 ms is precisely the desired spike time for that pattern in the WBC data set. On the other hand, in Fig. 6b, the tracing of PSP is shown using untrained weights and delays here spiking happens at 0.1 ms, which is to be trained efficiently to reach close to either 1 ms or 2 ms (since these two are the desired spike times for the WBC data set).

2.4 Neuronal connections

The design and organization of the synapses, along with the readout neuron, are described in this section. Figure 7 (a) shows the feed-forward (direction of the input stimuli towards the readout neuron is only in the forward fashion) single layer SNN (the only layer for the feeding of the input stimuli to the readout neuron). The pre-synaptic spike times of the first pattern of a data set $f_1^{(1)}, ..., f_{F-2}^{(1)}, f_{F-1}^{(1)}, f_F^{(1)}$, are calculated from the population encoding in the information encoding phase. Note that $f_0^{(1)}$ in the architecture represents the bias neurons spike, which is assigned to 0 ms. The bias spike helps the initial starting when most spikes are lateral.

Although this architecture is proposed for classifying the non-linear patterns, no hidden layer(s) and no hidden neuron(s) exist. The exciting part is that it is possible only with the SNN.

When an excitatory neuron excites the PSP, it is called the excitatory post-synaptic potential (EPSP). When an inhibitory neuron inhibits the PSP, it is called inhibitory post-synaptic potential (IPSP). In Fig. 7b, the EPSP and IPSP are shown along with the positive and negative synaptic weights W_1 and W_2 respectively.

Also, it is observed that there is no actual processing at the pre-synaptic neurons since the task of these neurons is to pass the inputs to the readout neuron. The actual information processing is only in the readout neuron. The synaptic delays can affect the pre-synaptic spikes and sometimes the spontaneous firing of spikes, as shown in Fig. 7b. In Fig. 7c, the activity inside the sub-threshold regime is shown where

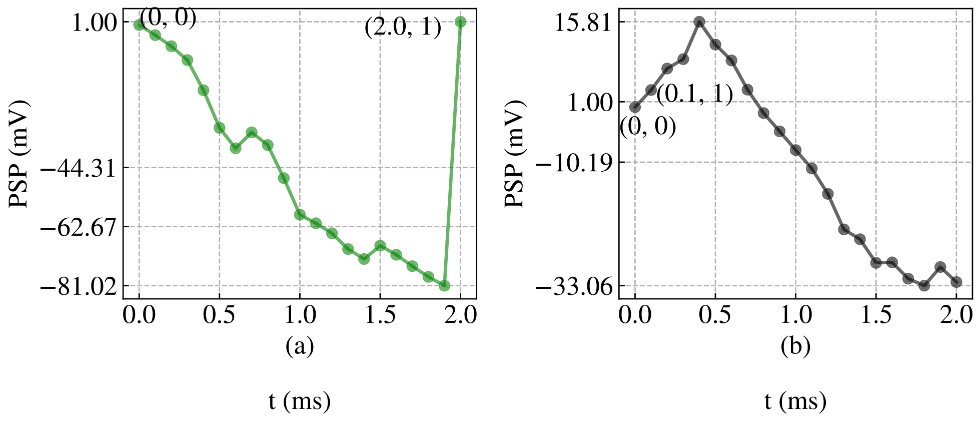

Fig. 6 Tracing of two different PSP curves for a single pattern of the WBC data set using **a** the trained synaptic weights and delays, **b** untrained synaptic weights and delays. The spiking happens in **a** at 2 ms, i.e., considered as the fine-tuned predicted spike time and in **b** at 0.1 ms, i.e., to be trained to reach close to 1 ms or 2 ms

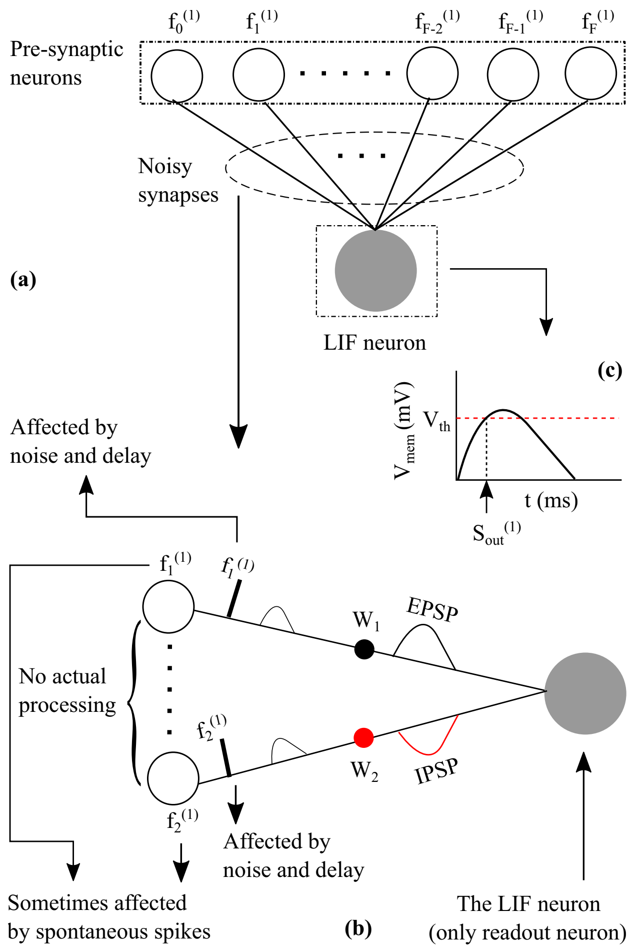

Fig. 7 **a** Architecture of the proposed model where inputs to the LIF neuron is coming from the first pattern of F pre-synaptic neurons such as $f_0^{(1)}, f_1^{(1)}, ..., f_F^{(1)}$ through the connected (one-to-one) noisy synapse **b** The synaptic connections of two different types of pre-synaptic neurons, one is excitatory (produces EPSP) and the other is inhibitory (produces IPSP) **c** The activity in the sub-threshold regime where $S_{out}^{(1)}$ is the first predicted output spike time for the first input pattern

$S_{out}^{(1)}$ is found to be the first predicted spike time produced from the LIF neuron when the V_{mem} reaches the V_{th}.

3 Learning of synaptic elements

In this phase of the classification task, the primary goal is to minimise the error produced while comparing the predicted and desired outputs. We have selected the metaheuristic approach for the optimisation task as it is among the most efficient, effective, and robust approaches. The metaheuristic used in this research is based on the concept of the evolutionary method, i.e., the genetic algorithm (GA) [49]. The reason for using GA over other optimization techniques is that it is simple to implement, and there is no need for any complex derivative information. Moreover, it has excellent parallel capabilities. The version of GA used is the floating-point GA or the real-coded GA, which can be directly utilised on the real floating-point numbers without mapping to binary numbers as in binary GA [46]. We have used the elitist selection method, the hybrid crossover method in [44], and the uniform mutation method [50, 51]. The elitism [52] process is used to retain and pass the best chromosomes to the next generation for better production of chromosomes in the next generation. The selection method, which does not use elitism, uses a crossover rate compared with a randomly generated number. The crossover operation is performed if the generated random number exceeds the crossover rate. However, here we have used elitist selection to retain a set of best chromosomes $Ch_{best}^{(i)}$ or solutions of the current generation i to be used in the next generation $(i + 1)$. Thus, the set of best chromosomes is always preserved if the chromosomes generated in generation $(i + 1)$ have lower fitness than that of its ancestors. We have used 20% elitism on the total number of population N. In this case, the objective function depends on synaptic weights W and delays t_{del} indirectly, which is the fitness value of the chromosomes defined in (16).

$$C_{fit}\left(W, t_{del}\right) = \frac{1}{1 + MSE} \tag{16}$$

where $C_{fit}(W, t_{del})$ is the function to be maximised. From (16), it is observed that to maximise $C_{fit}(W, t_{del})$, we have to minimise the MSE. GA does the maximisation task by default, although we can change the objective function to perform the minimisation task. The definition of MSE is given in the (17).

$$MSE = \frac{1}{P}\sum_{i=1}^{P}\left(S_{out}^{(i)} - S_{des}^{(i)}\right)^2 \tag{17}$$

where $S_{des}^{(i)}$ is the desired output spike, which is then labelled outputs in the case of a supervised learning paradigm.

We have used 20% elitism on the total number of population N to retain a set of best chromosomes $Ch_{best}^{(i)}$ or solutions of the current generation i to be used in the next generation $(i + 1)$. Thus, the set of best chromosomes is always preserved if the chromosomes generated in generation $(i + 1)$ have lower fitness than that of its ancestors.

In the hybrid crossover as used in [44], the first crossover method is given in (18) and (19) [46].

$$Ch_1^{(g+1)} = Ch_1^{(g)} - r_1 \times \left(Ch_1^{(g)} - Ch_2^{(g)}\right) \tag{18}$$

$$Ch_2^{(g+1)} = Ch_2^{(g)} + r_2 \times \left(Ch_1^{(g)} - Ch_2^{(g)}\right) \tag{19}$$

where $Ch_1^{(g+1)}$ and $Ch_2^{(g+1)}$ are the generated first and second chromosomes of $(g+1)^{th}$ generation from its ancestors $Ch_1^{(g)}$ and $Ch_2^{(g)}$ respectively. The value of r_1 and r_2 are two random numbers within the open interval (0, 1). The second crossover method is the single-point crossover method. We have selected the median of genes position as the crossover point and interchanged the genes of all chromosomes between the first and second half of the crossover point. The hybrid crossover method improves the convergence rate while searching for the optimal solutions in the search space.

For adding diversity to the search space, we have used uniform mutation [50, 51]. The rate of mutation is set to 0.1. The definition of uniform mutation is given in (20).

$$Ch_{mut} = lb + r_3 \times (ub - lb) \tag{20}$$

where Ch_{mut} is the mutated chromosomes, lb is the lower bound, and ub is the upper bound of the genes value, and r_3 is a random number within the open interval (0, 1).

3.1 Algorithms for learning

The Algorithm 1 shows the steps followed while training the proposed model. The mechanism of PSP updating and receiving input stimuli from the noisy synapses by the receptor or the readout neuron is given in the Algorithm 2. Algorithm 3 is the synapse model implemented using the Heaviside function.

Algorithm 1 Learning-Parameters $\left(f_F^{(P)}, S_{des}^{(P)}\right)$

```
   Input: $f_F^{(P)}$ and $S_{des}^{(P)}$ where $P$ is the number of patterns.
   Output: $N_{\alpha\times\beta}^{g+1}$
 1 $a_{noise} = \mathbf{N}[0,1]$
 2 $I_{spon} = 0.5$
 3 $t_{del} = \mathbf{U}[0,1]$
 4 $W = \mathbf{U}[-1,1]$
 5 $g = 0$
 6 $N_{\alpha\times\beta}^{g} = [W \; t_{del}]$
 7 $\gamma = 0.8 \times \alpha$
 8 $\delta = \mu_R \times \alpha$
 9 While $g < G$
10     $\mathcal{F}_{\alpha}^{g}$ = Evaluate-Fitness$\left(t_{del}, f_F^{(P)}, S_{des}^{(P)}, W, a_{noise}, I_{spon}\right)$
11     $N_{\alpha\times\beta}^{g}$ = Elitist-Selection$\left(N_{\alpha\times\beta}^{g}, \mathcal{F}_{\alpha}^{g}\right)$
12     $N_{(\alpha-\gamma)\times\beta}^{g+1} = N_{(\alpha-\gamma)\times\beta}^{g}$
13     $N_{\gamma\times\beta}^{g}$ = Hybrid-Crossover$\left(N_{\gamma\times\beta}^{g}\right)$
14     $N_{\gamma\times\beta}^{g}$ = Uniform-Mutation$\left(N_{\gamma\times\beta}^{g}, \delta\right)$
15     $N_{\alpha\times\beta}^{g+1} = \left[N_{(\alpha-\gamma)\times\beta}^{g} \; P_{\gamma\times\beta}^{g}\right]$
16     $g = g + 1$
17 Return $N_{\alpha\times\beta}^{g+1}$
```

Algorithm 2 Evaluate-Fitness $\left(t_{del}, f_F^{(P)}, S_{des}^{(P)}, W, a_{noise}, I_{spon}\right)$

```
1  $S'^{(P)}_{pre} = \left(f_F^{(P)} + t_{del} + a_{noise}\right)$
2  $S^{(P)}_{pre} = \left(f_F^{(P)} + t_{del} + a_{noise} + I_{spon}\right)$
3  For $p = 1$ to $P$
4      For $t = \delta t$ to $T$
5          $r = rand()$
6          If $r > 0.5$
7              $\delta V_{mem}(t) = \frac{\delta t\left(-V_{mem}(t-1) + W \times \text{Synapse-Model}\left(t - S_{pre}^{(p)}\right)\right)}{\tau_{mem}}$
8          Else
9              $\delta V_{mem}(t) = \frac{\delta t\left(-V_{mem}(t-1) + W \times \text{Synapse-Model}\left(t - S'^{(P)}_{pre}\right)\right)}{\tau_{mem}}$
10         $V_{mem}(t) = V_{mem}(t-1) + \delta V_{mem}(t)$
11         If $V_{mem}(t) \geq V_{th}$
12             $S_{out}^{(p)} = t$
13             Break
14 $MSE = \frac{1}{P}\sum_{i=1}^{P}\left(S_{out}^{(i)} - S_{des}^{(i)}\right)^2$
15 $fitness = \left(\frac{1}{1+MSE}\right)$
16 Return $fitness$
```

Algorithm 3 Synapse-Model (t)

```
1  If $t > 0$
2      Return $exp\left(-\frac{t}{\tau_{mem}}\right) - exp\left(-\frac{t}{\tau_{syn}}\right) + exp\left(-\frac{t^2}{2\tau_{mem}^2}\right)$
3  Else
4      Return 0
```

4 Results and discussion

We have used four binary datasets for benchmarking. The description of these datasets, along with the other details, are discussed in the following subsections.

4.1 Wisconsins breast cancer (WBC) dataset

The objective of the WBC [53] data set is to classify whether a person has breast cancer (*Malignant*) or not (*Benign*). The data set consists of 699 patterns, with 16 missing features for some patterns. We have removed the missing values to get a total number of 683 patterns. The *Benign* class has 444 patterns out of 683, and the *Malignant* class has 239 patterns. The 9 real-valued continuous features are transformed into 28 presynaptic input temporal spikes ($9 \times 3 + 1 = 28$ where 3 is the number of encoding neurons and 1 is the number of bias neurons). The desired output spikes are encoded as 1 ms represents the *Benign* class, and 2 ms represents the *Malignant* class.

4.1.1 Ionosphere (ION) dataset

The radar data was acquired for the ION dataset [54]. The dataset is used to identify whether the condition of the ionosphere is Good or Bad. A collection of radar data is collected through an antenna to classify the condition of the ionosphere, whether it is in *Good* (Class 1) or *Bad* (Class 2) condition. There are 351 samples, each with 33 attributes representing features. Of 351 samples, 225 samples belong to the *Good* class, and 126 belong to the *Bad* class. The 33 features are converted into 33×3+1=100 presynaptic input spikes (3 encoding neurons and 1 bias neuron).

4.1.2 Liver disorder (LIV) dataset

There are a total of 345 samples, each sample having 6 attributes that describe the features of the samples [55]. The first 5 variables in the LIV dataset are all blood tests known to be sensitive to liver diseases caused by excessive alcohol intake. The dataset is for the classification of the condition of a liver into two classes, namely *Healthy* (Class 1) and *Unhealthy* (Class 2). The *Healthy* class has 145 samples, and the *Unhealthy* class has 200 samples. There are 6×3+1=19 (3 encoding neurons and 1 bias neuron) presynaptic input spikes.

4.1.3 Pima Indian diabetes (PID) dataset

Based on specific diagnostic parameters included in the PID dataset [56], it is used to diagnose whether a patient is *Diabetic* (Class 1) or *Non-diabetic* (Class 1). There are 768 samples, of which 500 are for *Diabetic* class, 268 for*Non-diabetic* class. The 8 real-valued features represent the information about the disease. A total of 8×3+1=25 presynaptic spikes are there in the network topology.

4.2 Performance metrics

The Precision, Recall, F1 Score, and AUC experimental results are presented in Table 1 for WBC, ION, LIV, and PID datasets. The whole experiment is tested for 10 trials for each size of N. The value of $N = 60$ shows the best results compared to the others.

The training phase of any classification model is the critical and crucial phase where a model learns from the given example patterns. The more a model learns, the more precisely the model can perform in testing. Since we found the value of $N = 60$ to be the best, tracing the convergence while training the patterns for the data sets WBC and ION is also presented for $N = 60$ in Fig. 8a and b.

In Fig. 8a, which is for the WBC data set, it is observed that the convergence curve does not vary so much for approximately 20 generations. The primary exploration of the search space happens with those 20 generations searching for the best results.

On the other hand, in Fig. 8b, which is for the ION data set, it is found that, in this case, the primary exploration of the search space is within the first 100 generations. The training for all the data sets was carried out for 1000 generations, but only 100 generations are shown in the training curves for better clarity.

The training curve, along with the exploration for the dataset LIV and PID, is shown in Fig. 9a and b, respectively.

Table 1 Experimental results for the datasets WBC, ION, LIV, and PID

Dataset	Class	Precision	Recall	F1 Score	AUC
WBC	Benign	0.96	0.98	0.97	0.95
	Malignant	0.97	0.93	0.95	
ION	Good	0.80	0.91	0.85	0.73
	Bad	0.74	0.55	0.63	
LIV	Healthy	0.48	0.57	0.52	0.54
	Unhealthy	0.60	0.52	0.56	
PID	Diabetic	0.50	0.72	0.59	0.66
	Non-Diabetic	0.79	0.60	0.68	

4.3 Sensitivity analysis of parameters

In this section, a sensitivity analysis of the critical parameters such as synaptic delay (t_{del}) distribution, axonal noise level (a_{noise}), GABA switch probability (P_r(GABA switch)) upon the overall performance of the proposed model is conducted. All the datasets used for bench-marking are analysed in terms of accuracy to observe the impact of the aforementioned parameters. It is observed from Table 2, that the performance of the model improved when the values of the synaptic delay are drawn from the uniform distribution within the range [0, 1], axonal noise is from normal distribution within the range [0, 1], and GABA-switch probability is 50% as compared to the other values.

4.4 Performance comparison

Table 3 shows the performance comparison of our proposed model for the WBC, ION, LIV, and PID datasets, respectively, with the state-of-the-art methods. The value of synaptic load is represented by L_{syn}, and the value of computational cost is represented by C_{cost}. The definition of L_{syn}, and C_{cost} is given in (21) and (23) respectively.

$$L_{syn} = \mathcal{N}_{in} \times \mathcal{N}_{hid} + \mathcal{N}_{hid} \times \mathcal{N}_{out} \tag{21}$$

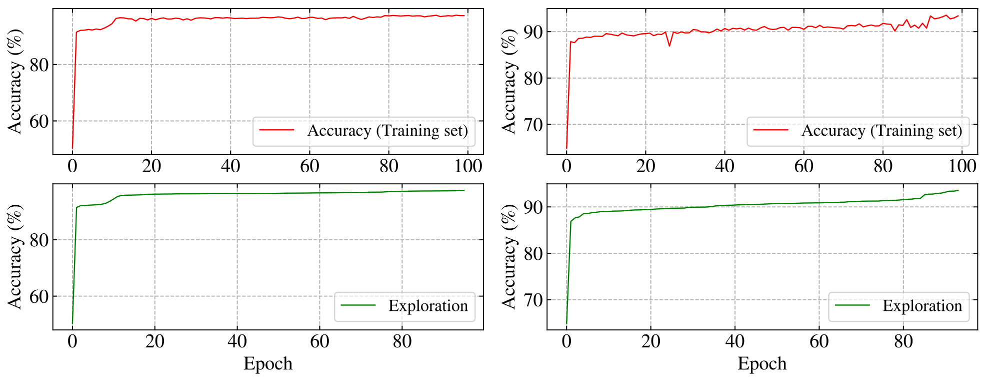

Fig. 8 **a** Accuracy curve (Training) showing the convergence to the optimum value of the Training Accuracy with the generation for the WBC data set **b** Accuracy curve (Training) showing the convergence to the optimum value of the Training Accuracy with the generation for the ION data set

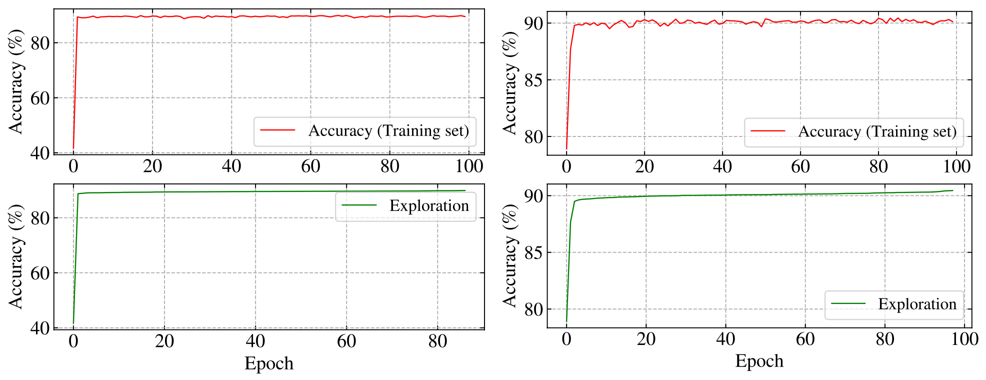

Fig. 9 **a** Accuracy curve (Training) showing the convergence to the optimum value of the Training Accuracy with the generation for the LIV data set **b** Accuracy curve (Training) showing the convergence to the optimum value of the Training Accuracy with the generation for the PID data set

Table 2 Sensitivity analysis to evaluate the impact of various parameters such as synaptic delay (t_{del}) distribution, axonal noise level (a_{noise}), GABA switch probability upon accuracy

Synaptic delay (t_{del})	Axonal noise (a_{noise})	P_r(GABA switch)	Dataset	Accuracy
U[0, 1]	N[0, 1]	0.5	WBC	**98.4**
			ION	**93.0**
			LIV	**88.9**
			PID	**90.4**
U[1, 2]	N[1, 2]	0.2	WBC	95.8
			ION	91.2
			LIV	88.7
			PID	89.8
U[-1, 1]	N[-1, 1]	0.8	WBC	95.9
			ION	89.1
			LIV	88.4
			PID	89.7

Bold values indicate the best results

where $\mathcal{N}_{in}$ is the number of input neurons, $\mathcal{N}_{hid}$ is the number of hidden neurons, and $\mathcal{N}_{out}$ is the number of output neuron(s).

Thus $\mathcal{N}_{in} \times \mathcal{N}_{hid}$ represents the total number of synaptic connections between $\mathcal{N}_{in}$ and $\mathcal{N}_{hid}$. Similarly $\mathcal{N}_{hid} \times \mathcal{N}_{out}$ represents the total number of synaptic connections between $\mathcal{N}_{hid}$ and $\mathcal{N}_{out}$. Since we did not use any hidden layer(s) as well as hidden neuron(s) (21) reduces to (22) in our case.

$$L_{syn} = \mathcal{N}_{in} \times \mathcal{N}_{out} \tag{22}$$

The drastic reduction in the value of L_{syn} is visible from (22). Moreover, we have used a ratio given in (23) to analyse and compare the computational cost.

$$C_{cost} = \frac{M_{en} \times G \times T}{\delta t} \tag{23}$$

where a total number of generations is denoted by G, to which the training is performed. From the (23), it is found that when the values of M_{en}, G, and T increase, the value of C_{cost} also increases.

However, our objective is to attain the lower value of C_{cost} since the lower value of C_{cost} indicates a model computationally more efficient. On the other hand, increasing δt decreases C_{cost}, but we found that increasing the value of δt more than 0.1 ms affects training the model (inaccurate training happens). When our proposed model's C_{cost} value is compared with the state-of-the-art, it outperforms all. For the WBC data set, the value of L_{syn}, and C_{cost} for the proposed model is found to be 28, and 5.6×10^5 respectively; those are the best results out there in Table 3. WOLIF shows better C_{cost}, which is 2.8×10^5 (since WOLIF runs for only 500 iterations). However, WOLIF has lower accuracy and is biologically less plausible than the proposed model in the case of the binary WBC dataset.

The topology and test accuracy values for SpikeProp, SWAT, OSNN, and SRESN in the case of ION, LIV, and PID dataset as mentioned in Table 3 were taken from [37], where these models were experimented on the aforementioned datasets. For the ION dataset, the value of L_{syn}, and C_{cost} for the proposed model is found to be 100 and 2.0×10^6 respectively; those are also the best results compared to the others except WOLIF which has 1.0×10^6. However, biological properties such as axonal noise and spontaneous firing are not considered in WOLIF. The testing accuracy for the proposed model is much better than others for the ION data set, which is 93.0±0.6%. The value of L_{syn} is 19, which is very good and also equal to WOLIF [45]. For the LIV dataset, the test accuracy value is 88.9±0.1%, which is much better than any other algorithm given in Table 3. For the PID dataset, the test accuracy value is 90.4±0.2%, which is much better than any other algorithm given in Table 3. Also, the value of L_{syn} is 25, which is much better than other algorithms and equal to WOLIF [45].

Table 3 Performance comparison of the proposed model with the state-of-the-art algorithms

Dataset	Model	Topology	L_{syn}	δt	T	Accuracy(%)	C_{cost}
WBC	SpikeProp [10]	55:15:2	855	0.01	50	97.0±0.6	6.4×10^9
	SWAT [36]	54:702:2	39,312	–	–	96.7±2.3	–
	OSNN [11]	54:(10–16):2	560–896	–	–	90.4±1.8	–
	SRESN [35]	54:(5–8)	270–432	–	–	94.0±2.6	–
	SEFRON [37]	55:1	55	0.01	4	96.4±0.7	2.2×10^6
	Evolutionary SNN [43]	–	–	–	–	95.9±0.8	–
	SpiFoG [44]	37:12:1	456	1	20	96.8±0.8	9.1×10^6
	WOLIF [45]	28:1	28	0.1	2	97.0±0.2	2.8×10^5
	Proposed	28:1	28	0.1	2	**98.4±0.3**	5.6×10^5
ION	SpikeProp [10]	199:25:2	5025	0.01	4	86.5±7.2	6.0×10^9
	SWAT [36]	198:2574:2	514,800	–	–	90.0±2.3	–
	OSNN [11]	198:(4–11):2	800–2200	–	–	76.6±4.8	–
	SRESN [35]	198:(6–13)	1188–2574	–	–	79.3±3.0	–
	SEFRON [37]	199:1	199	0.01	4	88.9±1.7	8.0×10^6
	Evolutionary SNN [43]	–	–	–	–	90.2±0.0	–
	WOLIF [45]	100:1	100	0.1	2	90.6±1.4	1×10^6
	Proposed	100:1	100	0.1	2	**93.0±0.6**	2.0×10^6
LIV	SpikeProp [10]	37:15:2	585	0.01	4	65.1±4.7	7.0×10^8
	SWAT [36]	36:468:2	17,784	–	–	60.9±3.2	–
	OSNN [11]	36:(4–7):2	152–266	–	–	56.7±1.8	–
	SRESN [35]	36:(5–8)	180–288	–	–	57.4±1.1	–
	SEFRON [37]	37:1	37	0.01	4	67.7±1.3	1.5×10^6
	Evolutionary SNN [43]	–	–	–	–	67.2±0.0	–
	WOLIF [45]	19:1	19	0.1	2	80.3±0.2	1.9×10^5
	Proposed	19:1	19	0.1	2	**88.9±0.1**	3.8×10^5
PID	SpikeProp [10]	49:20:2	1020	0.01	4	76.2±1.8	1.2×10^9
	SWAT [36]	48:624:2	31,200	–	–	72.1±1.8	–
	OSNN [11]	48:(8–18):2	400–900	–	–	63.5±3.0	–
	SRESN [35]	48:(6–12)	288–576	–	–	66.1±1.4	–
	SEFRON [37]	49:1	49	0.01	4	74.0±1.2	2.0×10^6
	Evolutionary SNN [43]	–	–	–	–	73.9±0.0	–
	WOLIF [45]	25:1	25	0.1	2	83.3±0.7	2.5×10^5
	Proposed	25:1	25	0.1	2	**90.4±0.2**	5.0×10^5

Bold values indicate the best results

5 Conclusion

This paper mainly explores the synapse model of a spiking neuron and the learning method from the example patterns efficiently and effectively, which is experimentally proven. Using the noisy synapse is biologically realistic and provides robustness to the model. The handling of noise is a challenging task that is efficiently implemented with the proposed model. When the biological properties of neurons come into the picture, the first and most important thing is managing the excitatory and inhibitory neurons properly; otherwise, there is always a tendency of not-firing spikes even after the increase in the total simulation time. In a biological neuron, the GABA switch manages this phenomenon. However, computationally, it is less explored since most of the SNN deals with either all excitatory or some small portion of inhibitory neurons with the remaining excitatory neurons. However, the sign-changing phenomenon is random in the biological neuron. Therefore, we have provided a 50–50 chance for the neurons to be either excitatory or inhibitory and appropriately trained, which converges to the optimum solutions. Convergence is the criterion that is hard to achieve when a mixture of excitatory and inhibitory neurons is used, so most SNN models prefer to use something other than it. Another attractive property of the biological neuron is its spontaneous firing activity, also used in our proposed model. This activity of spontaneous firing is much less explored in the case of the SNN. In our model, this property and the aforementioned properties are considered and experimentally proven to have better results than state-of-the-art, keeping the computational cost as minimal as possible (due to the absence of hidden layer(s)) and biological plausibility as high as possible.

Moreover, a kernel function by neglecting the mean term of RBF kernel and replacing the standard deviation with τ_{mem} is added with a double-decaying kernel function to handle the noise properly. Due to the noise, a distortion happens while exchanging information between a pre and post-synaptic neuron. There is a need for kernel function, which can rapidly increase the PSP of a post-synaptic neuron. It helps keep the total simulation time value as small as possible, which is advantageous towards the computational cost. Finally, it can be summarized that the proposed model outperforms all other aforementioned methods in terms of accuracy, L_{syn}, and C_{cost} for all data sets used for benchmarking.

It is possible to extend this work by allowing multiple spikes to fire from the readout LIF neuron. However, multiple spike firing activity demands an increase in total simulation time, which is computationally costly. As mentioned, the architecture will be explored in future work to cope with the classification problems where multiple spikes are necessary, as in time-series analysis. Due to properties such as robustness to noise, computationally inexpensive, and biological plausibility, this model can easily experiment with the time-series data where noise is a devil factor.

Data availability The datasets WBC, ION, LIV, and PID used in this paper are downloaded from the following repositories. The link for the WBC dataset: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original), for the ION dataset: https://archive.ics.uci.edu/dataset/52/ionosphere, for the LIV dataset: https://archive.ics.uci.edu/dataset/60/liver+disorders, and for the PID dataset: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

Declarations

Conflict of interest Both authors declare that there are no Conflict of interest.

References

1. Gerstner W, Kistler WM (2002) Spiking neuron models: single neurons, populations, plasticity. Cambridge university press
2. Natschläger T, Ruf B (1998) Spatial and temporal pattern analysis via spiking neurons. Netw: Comput Neural Syst 9(3):319–332
3. Gerstner W, Kistler WM, Naud R, Paninski L (2014) Neuronal dynamics: From single neurons to networks and models of cognition. Cambridge University Press
4. Maass W (1997) Noisy spiking neurons with temporal coding have more computational power. In: Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference, vol 9, p 211. MIT Press
5. Bialek W, Rieke F, Van Steveninck RDR, Warland D (1991) Reading a neural code. Science 252(5014):1854–1857
6. Bohte SM (2004) The evidence for neural information processing with precise spike-times: a survey. Natural Comput 3(2):195–206
7. Maass W (1997) Networks of spiking neurons: the third generation of neural network models. Neural Netw 10(9):1659–1671
8. Davies M, Srinivasa N, Lin T-H, Chinya G, Cao Y, Choday SH, Dimou G, Joshi P, Imam N, Jain S et al (2018) Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro 38(1):82–99
9. Cassidy A, Sawada J, Merolla P, Arthur J, Alvarez-lcaze R, Akopyan F, Jackson B, Modha D (2016) Truenorth: a high-performance, low-power neurosynaptic processor for multi-sensory perception, action, and cognition. In: Proceedings of the Government Microcircuits Applications & Critical Technology Conference, Orlando, FL, USA, pp 14–17
10. Bohte SM, Kok JN, La Poutre H (2002) Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing 48(1–4):17–37
11. Wang J, Belatreche A, Maguire L, McGinnity TM (2014) An online supervised learning method for spiking neural networks with adaptive structure. Neurocomputing 144:526–536
12. Stein RB (1965) A theoretical analysis of neuronal variability. Biophys J 5(2):173–194
13. Stein RB (1967) Some models of neuronal variability. Biophys J 7(1):37–68
14. Lapicque L (1907) Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. Journal de Physiologie et de Pathologie Generalej 9:620–635
15. Hodgkin AL, Huxley AF (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol 117(4):500–544
16. Kistler WM, Gerstner W, Hemmen JLV (1997) Reduction of the Hodgkin–Huxley equations to a single-variable threshold model. Neural Comput 9(5):1015–1045
17. Izhikevich EM (2003) Simple model of spiking neurons. IEEE Trans Neural Netw 14(6):1569–1572
18. Zeng N, Wu P, Zhang Y, Li H, Mao J, Wang Z (2024) Dpmsn: a dual-pathway multiscale network for image forgery detection. IEEE Transactions on Industrial Informatics
19. Faisal AA, Selen LP, Wolpert DM (2008) Noise in the nervous system. Nature Rev Neurosci 9(4):292–303
20. Katz B, Miledi R (1965) The measurement of synaptic delay, and the time course of acetylcholine release at the neuromuscular junction. In: Proceedings of the Royal Society of London, Series B. Biological Sciences. 161(985):483–495
21. Kerschensteiner D (2014) Spontaneous network activity and synaptic development. Neurosci 20(3):272–290
22. Häusser M, Raman IM, Otis T, Smith SL, Nelson A, Du Lac S, Loewenstein Y, Mahon S, Pennartz C, Cohen I et al (2004) The beat goes on: spontaneous firing in mammalian neuronal microcircuits. J Neurosci 24(42):9215–9219
23. Ganguly K, Schinder AF, Wong ST, Poo M-M (2001) Gaba itself promotes the developmental switch of neuronal gabaergic responses from excitation to inhibition. Cell 105(4):521–532
24. Lee SW, Kim Y-B, Kim JS, Kim WB, Kim YS, Han HC, Colwell CS, Cho Y-W, Kim YI (2015) Gabaergic inhibition is weakened or converted into excitation in the oxytocin and vasopressin neurons of the lactating rat. Mol Brain 8(1):1–9

25. Shrestha SB, Song Q (2015) Adaptive learning rate of spikeprop based on weight convergence analysis. Neural Netw 63:185–198
26. Booij O, Nguyen H (2005) A gradient descent rule for spiking neurons emitting multiple spikes. Inform Process Lett 95(6):552–558
27. McKennoch S, Liu D, Bushnell LG (2006) Fast modifications of the spikeprop algorithm. In: International Joint Conference on Neural Network Proceedings, pp. 3970–3977, IEEE
28. Xu Y, Zeng X, Han L, Yang J (2013) A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. Neural Netw 43:99–113
29. Ghosh-Dastidar S, Adeli H (2009) A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection. Neural Netw 22(10):1419–1431
30. Xu Y, Yang J, Zhong S (2017) An online supervised learning method based on gradient descent for spiking neurons. Neural Netw 93:7–20
31. Taherkhani A, Belatreche A, Li Y, Maguire LP (2018) A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks. IEEE Trans Neural Netw Learn Syst 29(11):5394–5407
32. Mostafa H (2017) Supervised learning based on temporal coding in spiking neural networks. IEEE Trans Neural Netw Learn Syst 29(7):3227–3235
33. Qu L, Zhao Z, Wang L, Wang Y (2020) Efficient and hardware-friendly methods to implement competitive learning for spiking neural networks. Neural Comput Appl 32(17):13479–13490
34. Toğaçar M, Ergen B, Cömert Z (2021) Detection of weather images by using spiking neural networks of deep learning models. Neural Comput Appl 33(11):6147–6159
35. Dora S, Subramanian K, Suresh S, Sundararajan N (2016) Development of a self-regulating evolving spiking neural network for classification problem. Neurocomputing 171:1216–1229
36. Wade JJ, McDaid LJ, Santos JA, Sayers HM (2010) Swat: a spiking neural network training algorithm for classification problems. IEEE Trans Neural Netw 21(11):1817–1830
37. Jeyasothy A, Sundaram S, Sundararajan N (2018) Sefron: a new spiking neuron model with time-varying synaptic efficacy function for pattern classification. IEEE Trans Neural Netw Learn Syst 30(4):1231–1240
38. Kasiński A, Ponulak F (2006) Comparison of supervised learning methods for spike time coding in spiking neural networks. Int J Appl Math Comput Sci 16(1):101–113
39. Lobo JL, Del Ser J, Bifet A, Kasabov N (2020) Spiking neural networks and online learning: an overview and perspectives. Neural Netw 121:88–100
40. Wang X, Lin X, Dang X (2020) Supervised learning in spiking neural networks: a review of algorithms and evaluations. Neural Networks
41. Taherkhani A, Belatreche A, Li Y, Cosma G, Maguire LP, McGinnity TM (2020) A review of learning in biologically plausible spiking neural networks. Neural Netw 122:253–272
42. Ahmed FY, Shamsuddin SM, Hashim SZM (2013) Improved spikeprop for using particle swarm optimization. Math Probl Eng 1:257085
43. Saleh AY, Shamsuddin SM, Hamed HNA (2017) A hybrid differential evolution algorithm for parameter tuning of evolving spiking neural network. Int J Comput Vision Robot 7(1–2):20–34
44. Hussain I, Thounaojam DM (2020) Spifog: an efficient supervised learning algorithm for the network of spiking neurons. Sci Rep 10(1):1–11
45. Hussain I, Thounaojam DM (2021) Wolif: an efficiently tuned classifier that learns to classify non-linear temporal patterns without hidden layers. Appl Intell 51(4):2173–2187
46. Haupt RL, Ellen Haupt S. (2004) Practical genetic algorithms. Wiley Online Library
47. Kuo B-C, Ho H-H, Li C-H, Hung C-C, Taur J-S (2013) A kernel-based feature selection method for svm with rbf kernel for hyperspectral image classification. IEEE J Select Top Appl Earth Observ Remote Sens 7(1):317–326
48. Minneci F, Kanichay RT, Silver RA (2012) Estimation of the time course of neurotransmitter release at central synapses from the first latency of postsynaptic currents. J Neurosci Methods 205(1):49–64
49. Holland JH, et al. (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press
50. Deb K, Gulati S (2001) Design of truss-structures for minimum weight using genetic algorithms. Finite Elem Anal Design 37(5):447–465
51. Lim SM, Sultan ABM, Sulaiman MN, Mustapha A, Leong K (2017) Crossover and mutation operators of genetic algorithms. Int J Mach Learn Comput 7(1):9–12
52. Baluja S, Caruana R (1995) Removing the genetics from the standard genetic algorithm. In: Machine Learning Proceedings 1995, pp 38–46. Elsevier
53. Wolberg WH, Mangasarian OL (1990) Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proceed Nat Acad Sci 87(23):9193–9196
54. Sigillito VG, Wing SP, Hutton LV, Baker KB (1989) Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Tech Digest 10(3):262–266
55. McDermott J, Forsyth RS (2016) Diagnosing a disorder in a classification benchmark. Pattern Recogn Lett 73:41–43
56. Pima Indians Diabetes Database (2017). https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database