Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 860 – 875

doi:10.55730/1300-0632.4022

Recognizing handwritten digits using spiking neural networks with learning algorithms based on sliding mode control theory

Yeşim ÖNİZ*, Mehmet AYYILDIZ
Department of Mechatronics Engineering, Faculty of Engineering and Natural Sciences,
İstanbul Bilgi University, İstanbul, Turkiye

Received: 29.09.2023 • **Accepted/Published Online:** 25.08.2023 • **Final Version:** 29.09.2023

Abstract: In this paper, a spiking neural network (SNN) has been proposed for recognizing the digits written on the LCD screen of an experimental setup. The convergence of the learning algorithm has been ensured by using sliding mode control (SMC) theory and the Lyapunov stability method for the adaptation of the network parameters. The spike response model (SRM) has been utilized in the design of the SNN. The performance of the proposed learning scheme has been evaluated both on the experimental data and on the MNIST dataset. The simulated and experimental results of the SNN structure have been compared with the responses of a conventional neural network (ANN) for which the weight update rules have been also derived using SMC theory. The conducted simulations and experimental studies reveal that convergence can be ensured for the proposed learning scheme and the SNN yields higher recognition accuracy compared to a conventional ANN.

Key words: Digit recognition, spiking neural networks, sliding mode control

1. Introduction

Handwritten digit recognition can be considered as a benchmark problem in the field of pattern recognition, with applications varying from automatic postal letter sorting to bank cheque processing. The challenges associated with this recognition task arise from the fact that the style of handwritten digits differs greatly across people and regions and it highly depends on the writing medium. Furthermore, even for the same person, the style of the digits might vary at different times. To overcome these difficulties and to provide improved recognition accuracy, various machine-learning techniques have been proposed in the literature. Among these, conventional and deep neural networks constitute the most extensively exploited ones [1–3].

In biological neurons, sequences of spikes are used to transmit information among the presynaptic and postsynaptic neurons. In the postsynaptic neurons, the incoming spikes give rise to alterations in the chemical and electrical properties of the membrane potential. Different mathematical models referred to as artificial neuron models, have been proposed to model the relationships among inputs and outputs of biological neurons [4]. The perception model forms the first generation of artificial neural networks, and in this relatively simple neuron model, a threshold function has been utilized as the activation function resulting in an all-or-none response as the output. In the second generation, the threshold function has been replaced by a continuous activation function enabling the outputs of neurons to lie within the interval $[0, 1]$. The typical network

*Correspondence: yesim.oniz@bilgi.edu.tr

structure of this generation of neural networks is the multilayer perceptron model (MLP). For both generations, the outputs can be considered as the normalized firing rate of the neurons, and hence this coding scheme has been generally called rate coding, which has been the prevailing approach in machine learning for many years. Several successful results have been reported on the application of this scheme for digit recognition problems [5]. Currently, convolutional neural networks (CNNs) have become the prevalent approach for recognition applications. CNNs are deep-learning neural networks composed of multiple successive convolutional layers followed by pooling layers. They constitute a powerful tool for pattern recognition tasks, as the number of hidden layers can be as many as 30, whereas in a traditional neural network, this number is limited to 3.

Despite the fact that breakthrough progress has been reported in the literature for both MLP and CNN, their information-processing process is biologically inaccurate and fails to reflect the actual working mechanism of biological neurons. On the contrary, spiking neural networks (SNNs) have gained popularity in recent years due to their ability to better model the temporally dynamic nature of biological neural systems. SNNs use spikes that are used to convey information, rather than continuous activation values as in traditional ANNs. This makes SNNs more energy efficient and biologically realistic [6, 7]. Analogous to the biological neurons, in SN models, the data to be transmitted among neurons are encoded in the precise timing information of the spikes whereas all spikes have the same magnitude, i.e. unlike the MLP, the magnitude of the spikes does not provide any information. Hence, SNNs are equipped with a more powerful information representation ability, which is essential for handling temporal and spatiotemporal data. Furthermore, the use of discrete spike trains instead of analog signals, lower energy requirement due to the asynchronous operating principle, and parallel processing capability make SNNs more suitable for hardware implementations than the second-generation networks [8].

The above-mentioned advantages along with the fast adaptation capability of SNNs result in an increasing number of applications of SNNs in diverse areas including the digit recognition problem [9–11]. In most of the research articles in this area, the backpropagation method with gradient-decent algorithm has been used [12–14]. Despite the reported successful results, the conventional methods used for the adaptation of the network parameters suffer from the possibility of convergence to local minima. Furthermore, the robustness of these algorithms cannot be guaranteed. This study aims to develop novel robust learning algorithms that can be used in the new generation of neural networks for machine learning applications. For this purpose, the weight adaptation rules have been derived by making direct use of the sliding mode control theory and Lyapunov stability theorem to guarantee the robustness of the learning process. The efficacy of the proposed learning scheme has been assessed on the data obtained from an experimental setup and also on the benchmark digit recognition dataset, the MNIST dataset.

This paper is organized as follows: The general structure of a spiking neural network has been explained in Section 2. In Section 3, the novel learning algorithm based on the sliding mode control theory has been detailed. The results of the simulations have been presented in Section 4. The details of the experimental setup and the results of the experimental studies have been given in Section 5. Concluding remarks have been presented in Section 6.

2. Spiking neural networks

There is a close resemblance between a spiking neuron and its biological counterpart in the sense that both the inputs and outputs of a spiking neuron have been described by the temporal information of the incoming and outgoing spikes. Additionally, multiple synaptic connections exist between a presynaptic neuron and a postsynaptic neuron, each having different weight values with varying delay times. The weight values between

the neurons indicate the strength of the connections. The spike generation process of a postsynaptic neuron is demonstrated in Figure 1. Spike trains, which can be described as the sequence of the firing times, constitute the spiking neuron inputs and outputs. Each firing time in the spike train denotes the time instant at which a pulse has been generated in the presynaptic neuron and then emitted to the postsynaptic neuron.

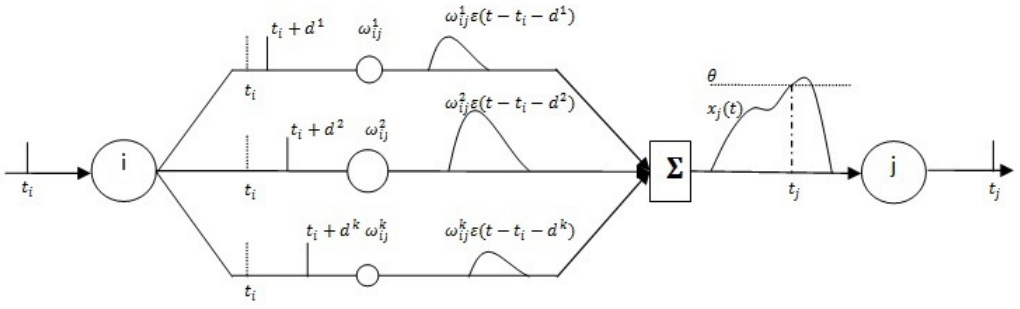

Figure 1. A spike generated at $t = t_i$ has been transmitted to neuron j by k different synapses, each with its own synaptic weight and delay time. If the aggregated sum of the incoming spikes exceeds the threshold Θ at time t_j, the neuron j will fire a spike at $t = t_j$

The membrane potential of a spiking neuron can be modeled as a leaky integrator, where the contribution of the newer spikes to the neuron's potential is greater than the former spikes. If the resulting neuron potential exceeds a predefined threshold, a pulse will be generated and then propagated to the postsynaptic neurons through multiple synapses. This scheme can be considered biologically plausible since, like the biological neurons, all spikes have the same amplitude, which indicates that the magnitude of the spikes is ineffective in information encoding, and the transmission of information is provided by the precise timing of the pulses.

In the literature, several mathematical formulations have been extensively used to describe the spike generation and information transmission processes among spiking neurons [9]. In this study, the spike response model (SRM) has been preferred over the alternatives because of its mathematical simplicity. Furthermore, to eliminate the need for a refractory term, it has been assumed that each neuron throughout the network can fire only a single pulse in each step of the simulation. As illustrated in Figure 1, a spike generated by presynaptic neuron i at $t = t_i$ will be propagated to neuron j by multiple synapses. Each synapse can have a different synaptic weight and delay time. The following spike response function describes the postsynaptic neuron potential as a result of the incoming spikes:

$$x_j(t) = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ij}^k \varepsilon(t - t_i - d^k) = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ij}^k y_i^k(t) \tag{1}$$

with N indicating the number of the presynaptic neurons of neuron j whereas K stands for the number of synaptic connections. t_i corresponds to the spike time of presynaptic neuron i with d^k being the time delay of the k-th synapse. w_{ij}^k denotes the weight value among neurons i and j for synapse k, with the former index indicating the presynaptic and the latter the postsynaptic neuron.

The following formulation for the spike response function, $\varepsilon(t)$, has been adopted in this study[12]:

$$\varepsilon(t) = \begin{cases} \frac{t}{\chi} e^{1-\frac{t}{\chi}} & t > 0 \\ 0 & t \leq 0 \end{cases} \tag{2}$$

with χ being the decay time constant. For neuron j, the firing time t_j corresponds to the time instant at which the membrane potential, i.e. weighted sum of the incoming pulses, becomes greater than the threshold value Θ, that is $x_j(t) \geq \Theta$.

3. Derivation of the learning algorithm for SNN

Similar to the multilayer perceptron model (MLP) of the second-generation neural networks, the parameter update rules for SNNs greatly rely on the gradient descent-based approaches [15, 16]. Even though several promising results have been attained with these learning algorithms, they can frequently result in suboptimal performance with respect to convergence speed and robustness. Moreover, as the error surface may include a number of local minima along with a global minimum, the learning algorithms employing gradient-based approaches fail to ensure the stability of the learning process as the system can easily get stuck in one of the local extrema [17].

To alleviate these issues, evolutionary methods can be applied [18, 19]. The main disadvantage of these approaches is that the stability of the convergence is also indeterminate and it is rather hard to determine the optimal values of the stochastic operators. As an alternative, learning algorithms relying on the sliding mode control (SMC) theory have been proposed to update the parameters of the neural network [20–22]. The main advantages of these learning algorithms are robust system response, faster convergence speed than the conventional learning schemes, and stability of the learning process.

SMC aims to direct the system trajectories into a plane called the sliding surface, on which the predefined error function is zero, and then keep them moving along this surface for all succeeding times. The sliding surface is generally specified by the difference of the outputs of the system from their target values and the derivatives of these differences. Such a formulation is given below:

$$S(x(t)) = \left(\frac{d}{dt} + \lambda\right)^{(n-1)} (y - y^d) = 0 \tag{3}$$

with λ being a strictly positive constant. In this formulation, y and y^d stand for the actual and target output states of the system, respectively, and n indicates the order of differentiation.

To secure the attractiveness of the sliding surface $S(x)$, the following conditions should be met:

- Any trajectory should remain on the sliding surface if it starts on this surface
- Any trajectory should converge to the sliding surface if it starts outside this surface

Regarding the first condition, if the system states start on the sliding surface, i.e. $S(x(t)) = 0$, then they should keep moving on this surface, which requires $\dot{S}(x(t)) = 0$. The second condition, which is also called the reachability condition, requires the following inequalities to be satisfied:

$$\lim_{S \to 0^+} \dot{S} < 0 \quad \text{and} \quad \lim_{S \to 0^-} \dot{S} > 0 \tag{4}$$

To satisfy this condition, the Lyapunov stability method should be employed [23].

Consider the Lyapunov function candidate stated as:

$$V(x,t) = \frac{1}{2}S(x(t))^2 \geq 0 \tag{5}$$

for which the equality only holds if $S = 0$. Recalling the Lyapunov stability theorem, to be able to guarantee the reachability condition, the derivative of the Lyapunov function should be negative definite, which requires the following condition to be met:

$$\dot{V}(x,t,S) = S(x(t))\frac{\partial S(x(t))}{\partial t} \leq 0 \tag{6}$$

with equality being only applicable if $S = 0$. If a positive definite Lyapunov function candidate fulfilling the reachability condition can be determined, then it will be guaranteed that the states of the system converge to the sliding surface and remain on this surface for the subsequent time instances.

In this study, a novel learning algorithm relying on the sliding mode control theory has been derived for multi-input multi-output (MIMO) systems. Considering that there are J output neurons in the given network structure, J independent sliding surfaces S_j $(j = 1, \ldots, J)$ have been designed. In the design of the sliding surfaces, the order of the differentiation has been set to $n = 1$, and the differences of the actual times/values of the output neurons from their target times/values have been employed. To satisfy the reachability condition, the parameter adaptation rules have been derived such that always a negative value has been attained for the product of each sliding surface S_j with its derivative $\dot{S}_j$. Thus, unlike the gradient approaches, the convergence of the algorithm has been guaranteed.

Consider a spiking neural network with a single hidden layer as illustrated in Figure 2. In this network structure, H neurons exist in the input layer, there are I neurons in the hidden layer and the output layer consists of J neurons. Furthermore, $w_{hi}^k(t)$ denotes the synaptic weight between the neuron h of the input layer and the neuron i of the hidden layer connected by the synapse k, whereas $w_{ij}(t)$ corresponds to the weight value among the neuron j of the output layer and neuron i of the hidden layer. To overcome the need for decoding and thus decrease the computational load, the neurons in the output layer are assumed to be with a linear activation function.

The learning error $e_j(t)$ can be defined as the scalar quantity obtained by $e_j(t) = \tau_j(t) - \tau_j^d(t)$, $\tau_j(t)$ and $\tau_j^d(t)$ being the actual and the desired outputs for the output neuron j, respectively. In accordance with the SMC approach, a time-varying sliding surface can be defined as follows:

$$S_j(e_j(t)) = e_j(t) = \left(\tau_j(t) - \tau_j^d(t)\right) = 0 \tag{7}$$

which provides the output of the neural network, $\tau_j(t)$, to coincide with the network's target output signal $\tau_j^d(t)$ for all succeeding times once the sliding surface is reached. The Lyapunov function candidates have been defined as follows:

$$V_j(S_j(t)) = \frac{1}{2}S_j^2(t) = \frac{1}{2}e_j^2(t) = \frac{1}{2}\left(\tau_j(t) - \tau_j^D(t)\right)^2 \tag{8}$$

The reachability condition requires that $V_j(S_j(t))\dot{V}_j(S_j(t)) \leq 0$. The derivative of the Lyapunov function candidate $V_j(S_j(t))$ can be expressed as:

$$\dot{V}_j(S_j(t)) = S_j(t)\dot{S}_j(t) = e_j\dot{e}_j = e_j(\dot{\tau}_j - \dot{\tau}_j^D) \tag{9}$$

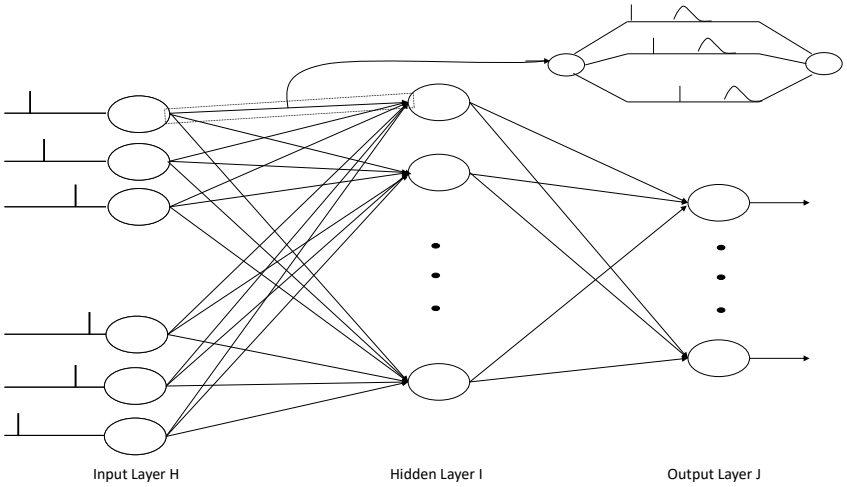

Figure 2. Structure of a three-layered SNN.

The chain rule has been utilized to determine the time derivative of the output signal, $\dot{\tau}_j$, as follows:

$$\frac{d\tau_j}{dt} = \frac{\partial \tau_j}{\partial w_{ij}} \frac{dw_{ij}}{dt} + \frac{\partial \tau_j}{\partial w_{hi}^k} \frac{dw_{hi}^k}{dt} = \frac{\partial \tau_j}{\partial w_{ij}} \dot{w}_{ij} + \frac{\partial t_i}{\partial x_i(t_i)} \frac{\partial x_i(t_i)}{\partial w_{hi}^k} \sum_{j=1}^{J} \frac{\partial \tau_j}{\partial t_i} \dot{w}_{hi}^k \tag{10}$$

Assuming that the desired outputs of the network are constants, $\dot{\tau}_j^D = 0$, then the reachability condition requires that

$$\frac{d\tau_j}{dt} e_j = \frac{\partial \tau_j}{\partial w_{ij}} e_j \dot{w}_{ij} + \frac{\partial t_i}{\partial x_i(t_i)} \frac{\partial x_i(t_i)}{\partial w_{hi}^k} \sum_{j=1}^{J} \frac{\partial \tau_j}{\partial t_i} e_j \dot{w}_{hi}^k \leq 0 \tag{11}$$

where the equality holds if and only if $e_j = 0$. The first term in 10 yields:

$$\frac{\partial \tau_j}{\partial w_{ij}} = \frac{\partial \left\{ \sum_{i=1}^{I} w_{ij} t_i \right\}}{\partial w_{ij}} = t_i \tag{12}$$

The derivative of the output of the spiking neural network with respect to the hidden layer neuron's firing times results in:

$$\frac{\partial \tau_j}{\partial t_i} = \frac{\partial \left\{ \sum_{i=1}^{I} w_{ij} t_i \right\}}{\partial t_i} = w_{ij} \tag{13}$$

The direct computation of the derivative of $\frac{\partial t_i}{\partial x_i(t_i)}$ is impossible as t_i is not continuous. Therefore, a linear function of t has been utilized to approximate the function x_i. For a small enough region around t_i, the threshold function has been approximated by $\delta t_i(x_i) = -\delta x_i(t_i)/\gamma$, with γ corresponding to the partial derivative of $x_i(t)$ with respect to t. Hence $\frac{\partial t_i}{\partial x_i(t_i)}$ evaluates to:

$$\frac{\partial t_i}{\partial x_i(t_i)} = \left[\frac{\partial t_i(x_i)}{\partial x_i(t)} \right]_{x_i = \Theta} = \frac{-1}{\frac{\partial x_i(t_i)}{\partial t}} = \frac{-1}{\sum_{h=1}^{H} \sum_{k=1}^{K} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \tag{14}$$

The last term $\frac{\partial x_i(t_i)}{\partial w_{hi}^k}$ in 10 can be computed as:

$$\frac{\partial x_i(t_i)}{\partial w_{hi}^k} = \frac{\partial \left\{ \sum_{i=1}^{I} \sum_{k=1}^{K} w_{hi}^k y_h^k(t_i) \right\}}{\partial w_{hi}^k} = y_h^k(t_i) \tag{15}$$

Using partial derivatives of 12-15, the time derivative of the neural network output becomes:

$$\dot{\tau}_j = t_i \dot{w}_{ij} - \sum_{j=1}^{J} \frac{w_{ij} y_h^k(t_i)}{\sum_{h=1}^{H} \sum_{k=1} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \dot{w}_{hi}^k \tag{16}$$

Using 16, the derivative of the Lyapunov function candidate yields:

$$\dot{V}_j = t_i e_j \dot{w}_{ij} - \frac{\sum_{j=1}^{J} w_{ij} e_j y_h^k(t_i)}{\sum_{h=1}^{H} \sum_{k=1}^{K} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i}} \dot{w}_{hi}^k \tag{17}$$

If the parameter update rules for the weights w_{ij} and w_{hi}^k are chosen as

$$\dot{w}_{ij} = -\alpha t_i sgn(e_j) \tag{18}$$

$$\dot{w}_{hi}^k = \beta y_h^k(t_i) sgn\left(\left(\sum_{j=1}^{J} w_{ij} e_j \right) \sum_{h=1}^{H} \sum_{k=1}^{K} w_{hi}^k \frac{\partial y_h^k(t_i)}{\partial t_i} \right. \tag{19}$$

then the derivative of the Lyapunov function candidate can be written as:

$$\dot{V}_j = -\alpha \left(t_i\right)^2 |e_j| - \beta \left| \sum_{j=1}^{J} w_{ij} e_j \right| y_h^k(t_i)^2 \tag{20}$$

which implies

$$\dot{V}_j \leq 0 \tag{21}$$

with the equality holding if and only if $e_j = 0$, and hence the convergence is guaranteed.

4. Simulation studies

The performance of the developed learning algorithm has been tested first on the MNIST dataset, which is still popular in testing various machine learning algorithms because of its accessibility and benchmarking capabilities. Especially for the studies utilizing spiking neurons for recognition tasks, the MNIST data set can be considered among the most widely used data sets [24–26], as the work on this new generation of neural networks is rather limited compared to second generation networks, and the use of the MNIST data set enables to compare and evaluate the performance of the new developed algorithms with the results reported in the literature. To assess the performance of the proposed learning scheme, learning algorithms based on the sliding mode control theory have also been adapted to a conventional MLP.

The MLP model used in this study also consists of 3 layers as shown in Figure 3. Similar to the proposed SNN structure, the neurons in the output layer have a linear activation function, whereas the neurons in the

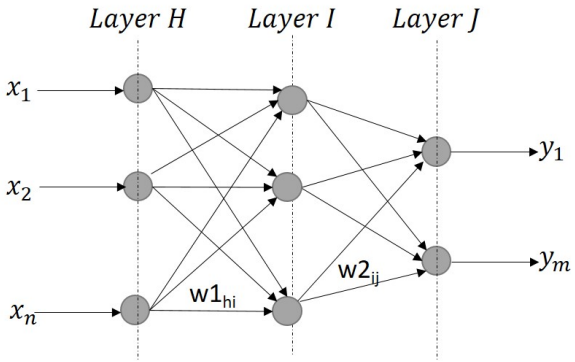

Figure 3. Multilayer perceptron model with a single hidden layer.

hidden layer have been equipped with a sigmoid activation function. For the given network structure, y_h^H, y_i^I and y_j^J correspond to the output values of the h-th neuron in the input layer H, i-th neuron in the hidden layer I and j-th neuron in the output layer J, respectively. For the output neuron j, the error function has been defined as $e_j(t) = y_j^J - y_j^D$, with $y_j^D(t)$ being the target output for the corresponding neuron.

Defining a different sliding surface for each output as $S_j = e_j$ and selecting the Lyapunov function candidate for neuron j as $V_j = \frac{1}{2}S_J^2$, the derivative of the Lyapunov function candidate can be computed as:

$$\dot{V}_j = y_i^I e_j \dot{w2}_{ij} + y_i^I(1 - y_i^I) y_h^H \sum_{j=1}^{J} w2_{ij} e_j \dot{w1}_{hi} \tag{22}$$

The following parameter update rules have been selected:

$$\dot{w2}_{ij} = -\alpha y_i^I sgn(e_j) \tag{23}$$

$$\dot{w1}_{hi} = -\beta y_h^H sgn\left(\sum_{j}^{J} w2_{ij} e_j\right) \tag{24}$$

then

$$\dot{V}_j = -\alpha (y_i^I)^2 |e_j| - \beta y_i^I \left(1 - y_i^I\right) (y_h^H)^2 \left|\sum_{j=1}^{J} w2_{ij} e_j\right| < 0, \forall e_j \neq 0 \tag{25}$$

In the simulated studies, the training of the proposed network structures has been accomplished on the first 60,000 samples of the MNIST dataset, and the testing phase has been carried out on the remaining 10,000 samples. Each sample in the original dataset comprises of gray-scale images of digits from 0 to 9 with a size of 28-by-28 pixels.

The MNIST dataset has been used in two different types of simulations: In the first one, it is used for both training and testing, and in the second study the testing is accomplished on the binary data acquired from the experimental setup. To provide consistency between training and testing data, the gray-scale images of the MNIST dataset have been transformed into binary images using scaling and threshold functions. Selected samples are shown in Figure 4.

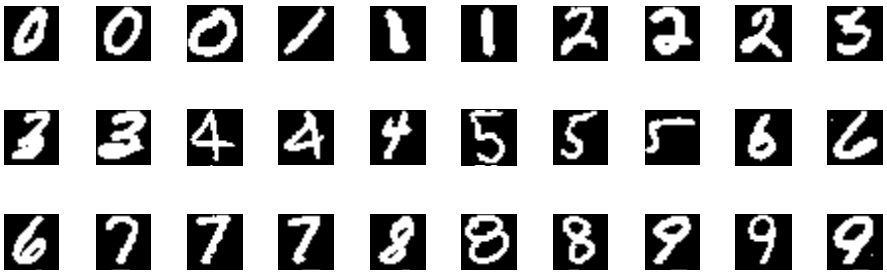

Figure 4. Binary images generated from the MNIST dataset.

Each image has been converted into a column vector with a dimension of 784 elements, and each neuron in the input layer receives one of the entries of the vector as an input. The hidden layer consists of 50 neurons. Each neuron in the output layer corresponds to one of the digits.

The input vector obtained from the black-and-white images can not be directly fed to the input layer of the SNN network. First the corresponding spike times for each input value should be computed. Delay coding can be called among the extensively used approaches for encoding data into precise timing of the spikes. The underlying idea of the delay coding is that the neurons subjected to higher input signals tend to generate a pulse earlier than the same neuron exposed to weaker signals. In this study, for the h-th element of the input vector with a value of 0, the corresponding firing time has been set as $t_h = 6 \ \ ms$, whereas a spike time of $t_h = 1 \ \ ms$ has been assigned to the input value of 1. As linear activation functions are utilized in the output layer, the need for the decoding operation has been omitted. 3 synapses have been assumed between each presynaptic and postsynaptic neuron to lessen the computational burden, as the increased number of synapses leads to exponentially growing computational time.

The same training and test sets have been used in the simulations of both network structures. The values of the coefficients α and β have been determined as 0.35 empirically. The maximum epoch number has been set to 500. Table 1 summarizes the obtained results.

Table 1. Results obtained with SNN and MLP networks for the MNIST dataset.

	Training set	Test set
MLP with gradient-based learning	96.46%	95.25%
MLP with SMC-based learning	96.87%	95.43%
SNN with gradient-based learning	97.41%	95.29%
SNN with SMC-based learning	97.99%	95.61%

Table 2 includes the accuracy results obtained in different studies for the MNIST data set using SNNs. It should be noted that in most of the studies, the number of hidden layer neurons is relatively large (several hundred), whereas only 50 hidden neurons have been used in the current study to lower computational burden. Even for this small value of hidden neurons, the network was able to attain an acceptable response compared to the results reported in the literature for larger network architectures.

Table 2. Accuracy of different learning algorithms on MNIST data set.

Reference	Network Structure	Test Accuracy
Oniz et al. (this study)	One hidden layer with 50 hidden neurons	95.6%
Mostafa [27]	One hidden layer with 800 hidden neurons	97.2%
Mostafa [27]	Two hidden layers each with 400 hidden neurons	97.0%
Diehl et al. [28]	One hidden layer with 1600 hidden neurons	95.0%
Khan et al. [29]	500 hidden neurons	91.9%
Zhao et al. [30]	Each image generates about 200 events	91.3%
Beyeler et al. [31]	3136 neurons in orientation layer	91.9%
Mirsadeghi et al. [32]	One hidden layer with 500 hidden neurons	97.4%
Qu et al. [33]	Two hidden layers each with 400 hidden neurons	92.0%
Zheng et al. [34]	Two hidden layers (300 neurons, 100 neurons)	97.8%

(a) **(b)**

Figure 5. (a) Schematic diagram of the experimental setup, (b) GUI used in experiments.

5. Experimental setup

The performance of the proposed learning algorithm has been assessed via the data acquired from an experimental setup consisting of a 21.5-inch LCD screen (HP, ProDisplay P223), a 21.5-inch multi-touch IR frame (Xintai Touch), and a PC. The LCD screen was placed horizontally on top of a desk, and the IR frame was positioned on top of it, which was for the detection and recording of the position of the participant's index. The PC and IR frame communication has been established via a USB interface. The illustrations of the setup used in the experiments are given in Figure 5 and Figure 6. A GUI on MATLAB has been developed to provide instructions to the participants and to record handwritten digits on the screen. For the recording operation, the sampling time has been set to 140 Hz, which enables capturing sufficient data points during the trials while avoiding delays due to the processing.

The experiments have been conducted with 20 subjects. Among the applicants, ten males and ten females have been selected as participants to avoid gender bias, if there is any. All participants were college students with an average age of 22 $\pm$ 2.4. The consent form, which was approved by the Ethical Committee for Human Participants of İstanbul Bilgi University, has been signed by all participants before the experiments.

The participants attended two sessions (i.e. the training session and the actual experiment). In the training session, subjects familiarized themselves with the experimental setup. They received verbal and

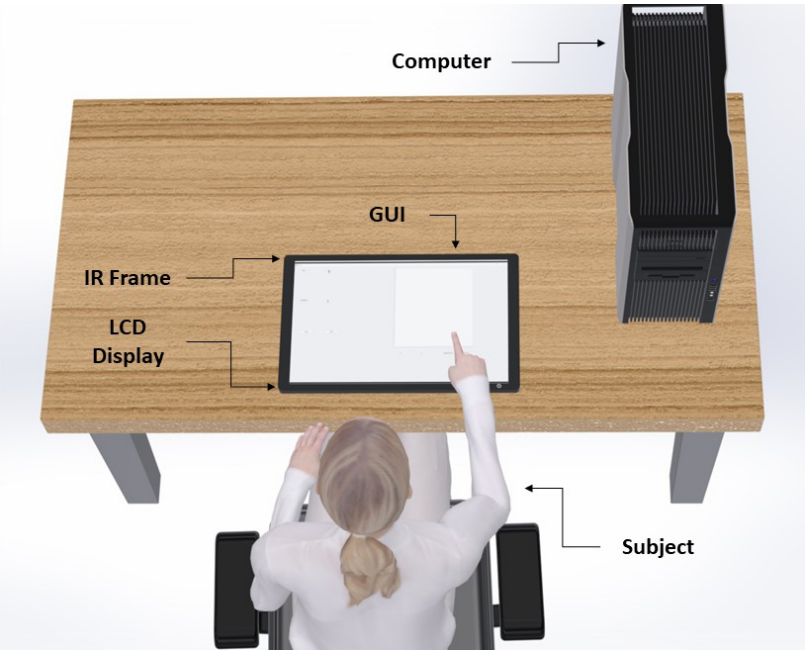

Figure 6. The illustration of the experimental setup.

visual guidance regarding the experimental procedure. The questions of the participants were answered by the experimenter during the training session, which was not permitted in the actual experiment.

Once the training session had been completed, the participants were allowed to perform the actual experiment. Their task has been stated as to write the digits shown on the LCD screen with their index finger according to the instructions provided by the GUI. Each experiment consisted of 100 trials, which corresponds to 10 repetitions for each digit. During the experiment, the digits were displayed in random order to the subjects. A keyboard has been used by the experimenter to initialize the session and then to switch between the trials. For each participant, the experiments have been conducted in a single session, and no time limit has been set for finishing the trial. Each experiment has been completed in around 10 min.

5.1. Preprocessing

In the experiments, the requested digits shown by the GUI have been drawn by the subjects on the screen. During the course of drawing, the position of the mouse cursor has been synchronized with the subjects' fingertips. The data, which were recorded before the session started and during finger lift when subjects were connecting curves to form the requested digits, resulted in redundant data points at the beginning and in the middle of the trial. A preprocessing algorithm has been employed to remove these redundant data points and create the digits without additional input.

Three columns comprising the x- and y-positions of the fingertip along with the time information when the point has been touched by the subject have been generated from the recorded data. The first two columns have been used to create a matrix. To eliminate the data points at the beginning of a trial before the subject actually started drawing the digit, the repeated rows of this matrix have been replaced with zeros, as these correspond to the redundant data points recorded at the starting position.

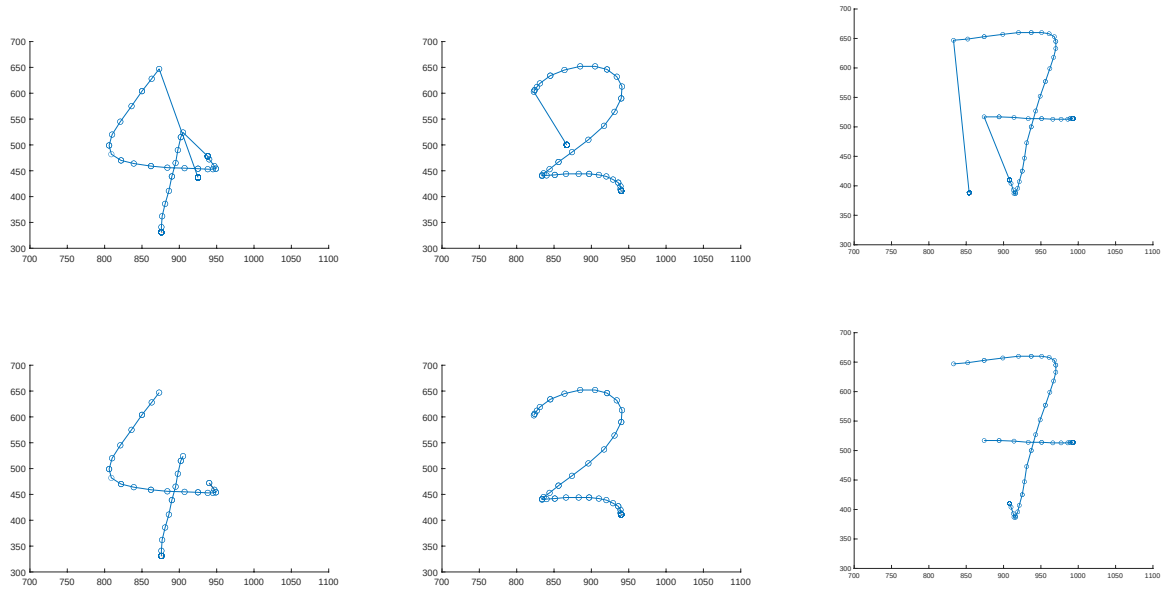

Figure 7. The figures in the upper row illustrate the digits drawn by the subjects before any preprocessing operation. The figures in the lower row correspond to the same digits after removal of redundant data point.

A preliminary inspection has been performed to recognize the duplicated data points occurring in the middle of a trial when drawing digits such as 4, 5, and 7, which generally require discontinuous movement of the fingertip on the screen. It has been observed that continuous handwriting could contain up to five repeated data points, and the presence of more repeating points indicates that the finger has been lifted from the screen (i.e. the contact between the finger and the screen has been interrupted). In accordance with the observations, the rows repeated more than ten times have been removed from the matrix. The final form of the matrix includes only the data points which correspond to the actual positions of the fingertip used to draw the digit (See Figure 7).

The area of the screen occupied by pixels from 700 to 1100 on the x-axis, and from 300 to 700 on the y-axis has been designated for drawing the digits. To provide consistency with the MNIST dataset, a $28x28$ grid structure has been generated by dividing these intervals into 28 equal parts. After eliminating the repeated data points from the data, a value of 1 has been assigned to the cells, on top of which the finger moved, whereas for the untouched cells the value has been set to 0. With this approach, a $28x28$ pixels binary image has been generated for each recording. Following the dilation operation with a $3x3$ kernel, bounding boxes around the generated patterns have been formed, and the areas outside these boxes have been eliminated to end up with the best fit of the pattern to the given area. Next, the image has been rescaled to provide a $28x28$ grid outline. Figure 8 illustrates the steps of the preprocessing operations on sample data. Sample patterns after the preprocessing operations have been presented in Figure 9.

5.2. Implementation of the models

Following the simulation studies on the MNIST dataset, a new approach has been pursued, in which the testing phase has been carried out on the experimental data. The motivation behind this approach was to test the

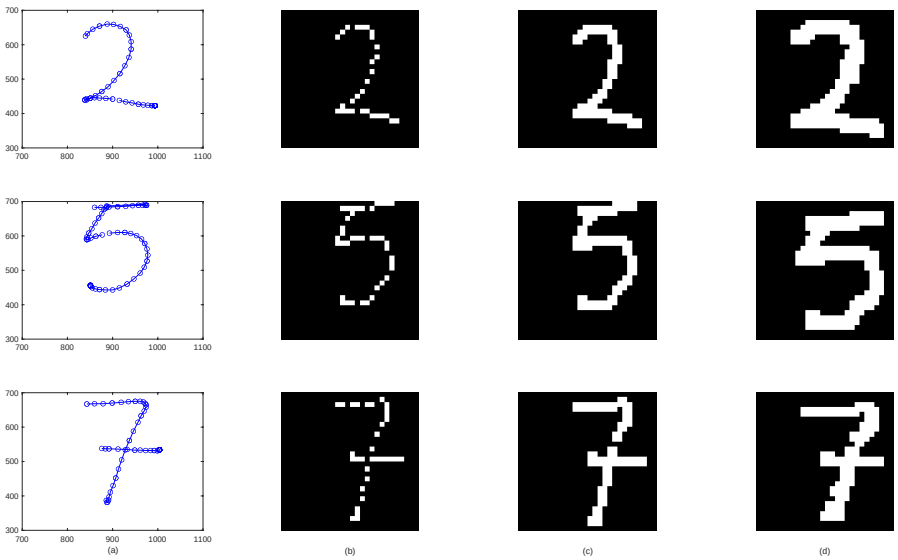

Figure 8. (a) Image after the repeated data points have been eliminated; (b) binary image; (c) dilated image; (d) scaled image.

abilities of both trained networks to recognize patterns in a new, yet similar dataset.

For the test data, the conventional MLP network was able to correctly classify 86.1%, whereas the performance of the SNN structure is 86.5%. The same tests have been repeated for the networks with sliding mode control theory-based learning algorithms. The accuracy achieved for the MLP is increased to 87.3 and for SNN to 88.1. The confusion matrix for the SNN with the proposed learning scheme is presented in Table 3. Regarding the confusion matrix, "1", "4" and "9" are the most misclassified patterns.

In the subsequent part of the studies, both training and testing were performed on the experimental data. The training dataset has been generated by randomly selected data from 8 male and 8 female subjects, and the test dataset has been formed using the data of the remaining subjects. Table 4 demonstrates the accuracy of training and testing phases, which indicate that the SMC-based learning algorithms can enhance the convergence characteristics, and for both types of learning algorithms, SNN can attain at least equal or

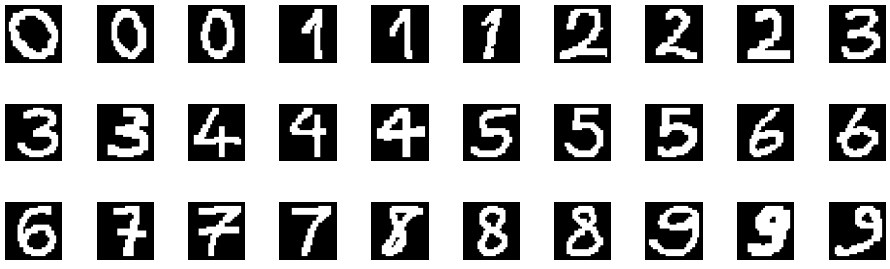

Figure 9. Sample patterns generated after preprocessing operations.

Table 3. Confusion matrix for the SNN with the proposed learning scheme (training on MNIST dataset).

	Target Digit										
Output Digit		0	1	2	3	4	5	6	7	8	9
	0	183	0	0	0	2	1	4	1	0	0
	1	0	162	1	1	2	0	2	5	1	0
	2	3	3	192	1	8	6	0	0	0	3
	3	0	3	1	194	0	18	4	9	14	15
	4	0	9	0	0	167	0	0	3	1	2
	5	4	4	0	3	0	169	8	0	2	3
	6	3	0	3	0	10	1	177	0	0	0
	7	0	6	1	0	3	0	0	177	0	9
	8	7	4	2	1	0	4	5	2	179	6
	9	0	9	0	0	8	1	0	3	3	162

better performance than the conventional neural networks. The confusion matrix for the SNN with the proposed learning scheme is presented in Table 5.

Table 4. Results of SNN and MLP on the dataset obtained from experiments.

	Training set	Test set
MLP with gradient-based learning	97.88%	90.25%
MLP with SMC-based learning	100%	93.75%
SNN with gradient-based learning	99.06%	92.00%
SNN with SMC-based learning	100%	94%

Table 5. Confusion matrix for the SNN with the proposed learning scheme (training on experimental data).

	Target Digit										
Output Digit		0	1	2	3	4	5	6	7	8	9
	0	40	0	0	0	0	1	0	0	0	0
	1	0	37	0	0	1	0	0	1	0	2
	2	0	0	39	0	0	0	0	0	0	1
	3	0	0	0	35	0	1	0	0	1	0
	4	0	1	1	0	39	0	1	0	0	0
	5	0	0	0	2	0	37	0	0	1	2
	6	0	0	0	0	0	0	39	0	1	0
	7	0	2	0	0	0	0	0	38	0	0
	8	0	0	0	3	0	1	0	0	37	0
	9	0	0	0	0	0	0	0	1	0	35

6. Conclusion

In this study, new learning rules based on the Lyapunov stability theorem have been derived for spiking neural networks. The efficacy of the proposed learning algorithms has been assessed on the handwritten digit

recognition problem both for MNIST dataset and experimental data. An experimental setup has been set up using a multi-touch IR frame on top of an LCD screen to record the fingertip positions of the subjects. Several preprocessing operations have been applied to convert the recorded position data into binary images. For the SNN structure, delay coding has been performed to transform pixel values into spike times. Sliding mode control theory-based learning algorithms have been also derived for a conventional MLP network, for which the same simulated and experimental studies have been carried out as well. The obtained results reveal that the proposed learning algorithm can enhance the convergence characteristics of the neural networks, and spiking neural networks are capable of providing better or at least equal performance than the second-generation neural networks.

Acknowledgement

The authors would like to acknowledge the support of Bilgi Research Fund (Project no: AK 850890000) during the course of this work.

References

[1] Ghosh MM, Maghari AY. A comparative study on handwriting digit recognition using neural networks. In: 2017 International Conference on Promising Electronic Technologies (ICPET); Deir El-Balah, Palestine; 2017. pp. 77-81.

[2] Ramzan M, Khan HU, Awan SM, Akhtar W, Ilyas M et al. A survey on using neural network based algorithms for hand written digit recognition. International Journal of Advanced Computer Science and Applications. 2018; 9 (9).

[3] Ahlawat S, Choudhary A, Nayyar A, Singh S, Yoon B. Improved handwritten digit recognition using convolutional neural networks (CNN). Sensors. 2020;20 (12): 3344.

[4] Haykin S, Network N. A comprehensive foundation. Neural networks. 2004; 2 (2004): 41.

[5] Saeed AM. Intelligent handwritten digit recognition using artificial neural network. Int. Journal of Engineering Research and Applications. 2015; 5 (5): 46-51.

[6] Kampakis S. Improved Izhikevich neurons for spiking neural networks. Soft Computing. 2012; 16 (6): 943-953.

[7] Yang S, Wang J, Deng B, Liu C, Li H et al. Real-time neuromorphic system for large-scale conductance-based spiking neural networks. IEEE transactions on cybernetics. 2018; 49 (7): 2490-2503.

[8] Farsa EZ, Ahmadi A, Maleki MA, Gholami M, Rad HN. A low-cost high-speed neuromorphic hardware based on spiking neural network. IEEE Transactions on Circuits and Systems II: Express Briefs. 2019 ;66 (9): 1582-1586.

[9] Yamazaki K, Vo-Ho VK, Bulsara D, Le N. Spiking neural networks and their applications: A Review. Brain Sciences. 2022; 12 (7): 863.

[10] Ponulak F, Kasinski A. Introduction to spiking neural networks: Information processing, learning and applications. Acta neurobiologiae experimentalis. 2011; 71 (4): 409-433.

[11] Abiyev RH, Kaynak O, Oniz Y. Spiking neural networks for identification and control of dynamic plants. In: 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM); Kaohsiung, Taiwan; 2012. pp. 1030-1035

[12] Bohte SM, Kok JN, La Poutre H. Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing. 2002; 48 (1-4): 17-37.

[13] Schrauwen B, Van Campenhout J. Improving SpikeProp: Enhancements to an error-backpropagation rule for spiking neural networks. In: Proceedings of the 15th ProRISC workshop; Veldhoven, the Netherlands; 2004. pp. 301-305.

[14] Mirsadeghi M, Shalchian M, Kheradpisheh SR, Masquelier T. STiDi-BP: Spike time displacement based error backpropagation in multilayer spiking neural networks. Neurocomputing. 2021; 427: 131-140.

[15] Lillicrap TP, Santoro A. Backpropagation through time and the brain. Current opinion in neurobiology. 2019; 55: 82-89.

[16] Whittington JC, Bogacz R. Theories of error back-propagation in the brain. Trends in cognitive sciences. 2019; 23 (3): 235-250.

[17] Xu Y, Yang J, Zhong S. An online supervised learning method based on gradient descent for spiking neurons. Neural Networks. 2017;93:7-20.

[18] Abusnaina AA, Abdullah R, Kattan A. Supervised training of spiking neural network by adapting the E-MWO algorithm for pattern classification. Neural Processing Letters. 2019; 49 (2): 661-682.

[19] Belatreche A, Maguire LP, McGinnity M, Wu QX. Evolutionary design of spiking neural networks. New Mathematics and Natural Computation. 2006; 2 (03): 237-253.

[20] Parma GG, Menezes BR, Braga AP. Sliding mode algorithm for training multilayer artificial neural networks. Electronics Letters. 1998; 34 (1): 97-98.

[21] Yu X, Kaynak O. Sliding-mode control with soft computing: A survey. IEEE transactions on industrial electronics. 2009;56 (9): 3275-3285.

[22] Oniz Y, Kaynak O. Variable-structure-systems based approach for online learning of spiking neural networks and its experimental evaluation. Journal of the Franklin Institute. 2014; 351 (6): 3269-3285.

[23] Utkin V, Guldner J, Shi J. Sliding mode control in electro-mechanical systems. CRC Press; 2017.

[24] Kheradpisheh SR, Mirsadeghi M, Masquelier T. BS4NN: binarized spiking neural networks with temporal coding and learning. Neural Processing Letters. 2022; 54 (2):1255-73.

[25] Javanshir A, Nguyen TT, Mahmud MP, Kouzani AZ. Advancements in Algorithms and Neuromorphic Hardware for Spiking Neural Networks. Neural Computation. 2022;34 (6):1289-328.

[26] Tang H, Kim H, Kim H, Park J. Spike counts based low complexity SNN architecture with binary synapse. IEEE Transactions on Biomedical Circuits and Systems. 2019;13 (6):1664-77.

[27] Mostafa H. Supervised learning based on temporal coding in spiking neural networks. IEEE transactions on neural networks and learning systems. 2017;29 (7):3227-35.

[28] Diehl PU, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. Frontiers in computational neuroscience. 2015;9:99.

[29] Khan MM, Lester DR, Plana LA, Rast A, Jin X et al. SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence); 2008. pp. 2849-2856.

[30] Zhao B, Ding R, Chen S, Linares-Barranco B, Tang H. Feedforward categorization on AER motion events using cortex-like features in a spiking neural network. IEEE transactions on neural networks and learning systems. 2014;26(9):1963-78.

[31] Beyeler M, Dutt ND, Krichmar JL. Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule. Neural Networks. 2013 Dec 1;48:109-24.

[32] Mirsadeghi M, Shalchian M, Kheradpisheh SR, Masquelier T. STiDi-BP: Spike time displacement based error backpropagation in multilayer spiking neural networks. Neurocomputing. 2021; 427:131-40.

[33] Qu L, Zhao Z, Wang L, Wang Y. Efficient and hardware-friendly methods to implement competitive learning for spiking neural networks. Neural Computing and Applications. 2020;32 (17):13479-90.

[34] Zheng N, Mazumder P. Online supervised learning for hardware-based multilayer spiking neural networks through the modulation of weight-dependent spike-timing-dependent plasticity. IEEE transactions on neural networks and learning systems. 2017;29 (9):4287-302.