Received: 14 April 2025 | Revised: 4 June 2025 | Accepted: 22 August 2025 | Published online: 13 October 2025

RESEARCH ARTICLE

Artificial Intelligence and Applications
2026, Vol. 4(2) 248–255
DOI: 10.47852/bonviewAIA52025930

BON VIEW PUBLISHING

Incorporating Fuzzy ARTMAP Network with Spiking Neural Networks

Issam Dagher[1,*]

[1] *Department of Computer Engineering, University of Balamand, Lebanon*

Abstract: In this paper, we have incorporated the Fuzzy Adaptive Resonance Theory Mapping (Fuzzy ARTMAP) classification module into the Spiking Neural Networks (SNNs). This new network can have multiple hidden layers, a supervised corrected layer, and an ARTMAP module. The objective of the supervised layer is to correct the spikes given by the Spike-timing-dependent plasticity (STDP) layer in a supervised manner. Then the resulting corrected spikes are fed into an ARTMAP network. When an output spiking train code is similar to one of the stored ART codes. It modifies that code to incorporate the new input into that code category. This combination required a new training algorithm. The hidden layers are trained using the popular STDP. This unsupervised learning is followed by a supervised layer where new supervised weight update formulas are presented. This new combination is compared to other spiking algorithms using datasets from the UCI repository. It is shown that SNN_ARTMAP provides better recognition results than other similar spiking supervised learning algorithms.

Keywords: SNN, ARTMAP, STDP, supervised learning

1. Introduction

Spiking Neural Networks (SNNs) are a type of artificial neural network that more accurately mimics the structure of natural neural networks. Transmission of information is done using action potentials (neuron spikes). Spikes are impulses or discrete events that appear at certain positions in a time interval. A first-order differential equation can serve to represent the membrane potential of a neuron. This equation is capable of forecasting whether a spike will happen or not. Because these spikes are non-differentiable, different SNN training techniques were developed. Indirect methods were used by first training the network in the continuous domain and then converting it into the discrete domain. In the work of Luo et al. [1], a learning algorithm utilizing gradient descent is introduced to optimize synaptic delays, aiming to improve the sequential learning ability of an individual spiking neuron.

Lower latency paired with substantial activation sparsity results in significant enhancements in computational efficiency [2]. Efficient implementation of SNN using FPFA is presented in the work of El Maachi et al. [3].

Different SNN training techniques were developed. Direct training of Deep SNN is shown in the work of Zhou et al. [4].

Indirect methods were used by first training the network in the continuous domain and then converting it into the discrete domain.

A new framework based on spikes, utilizing entropy theory, was proposed in the work of Guo et al. [5]. Yang et al. [6] designed an efficient learning mechanism with spiking dendrites. New training techniques for SNN were presented in the work of Saranirad et al. [7], Meng et al. [8], and Hao et al. [9]. Gouda et al. [10] applied the surrogate gradient method, which is a very efficient method to overcome the discontinuity of the spike. It is based on continuous relaxation of the real gradient. Hardware implementation of SNN is presented in the work of Benismon et al. [11] and Frenkel et al. [12]. Converting Deep ANN to SNN with a small number of time steps is shown in the work of Meng et al. [13]. In the work of Li et al. [14], Differentiable Spike (Dspike) functions were introduced. The optimal smoothness for the gradient was determined using the difference gradient.

Adaptive resonance theory (ART) was first proposed in 1976 by Grossberg. As shown in the work of Tashiro et al. [15], the theory was developed to address the stability/plasticity problem. A key difficulty in incremental (or online) learning is the "stability-plasticity dilemma," commonly referred to as "catastrophic forgetting." The system strives to retain and remember as much information as possible because it is uncertain about which information is truly significant to the user and may consequently lose (forget) crucial data. When new content is delivered gradually, the learning is referred to as online learning. Information will only be received once by the system, after which it will modify its model description and get ready for fresh input. Given the restricted amount of memory storage capacity, information that was previously saved during learning may vanish since fresh knowledge has replaced it.

Fuzzy ART neural network is a clustering technique based on Adaptive Resonance Theory. A rectangle serves as the geometric representation of each cluster. Two fuzzy ARTs make up the architecture of fuzzy ARTMAP. Supervised learning characterizes the Fuzzy ARTMAP. It includes two stages: the training stage, during which labeled clusters (rectangles) are created, and the testing stage, in which any unlabeled test data is categorized.

The presentation order of patterns has a significant impact on this architecture's generalization performance. The max–min clustering algorithm was used in the work of Dagher et al. [16] to achieve an efficient order of pattern presentation.

2. Spiking Neural Networks – ARTMAP Prototype

SNNs are a form of artificial neural networks that imitate the organization of biological neural networks. Unlike Multi-Layer-Perceptrons (MLP), SNN integrates the aspect of time into its operational model. In an SNN, neurons transmit information when their membrane potential reaches a certain threshold level. Once the membrane potential surpasses this threshold, the neuron produces a signal or spike that is

***Corresponding author:** Issam Dagher, Department of Computer Engineering, University of Balamand, Lebanon. Email: dagheri@balamand.edu.lb

sent to neighboring neuron. The output of many spikes is called a spike train. The communication and thus learning between neurons is based on the generated spike trains. Only at spike occurrences does a neuron communicate with other neurons. This communication takes place via a synapse between two neurons. Every neuron has the ability to obtain signals from numerous other neurons. To form the output spike train, the currents from all synapses are combined and integrated at the receiving end of the neuron. The activity of pre- and post-synaptic neurons affects the conductance of a synapse, also known as synaptic weight. This activity is thought to be responsible for neurons' ability to learn. Typically, spikes in the pre- and/or post-synaptic neuron cause weight changes and eventually learning.

The postsynaptic current is given by:

$$PS(t) = e^{-t/t1} - e^{-t/t2} \tag{1}$$

where t1 and t2 are time constants representing the steepness of rise and decay times.

2.1. Leaky integrate-and-fire neuron (LIF)

Leaky Integrate-and-Fire (LIF) is the simplest model of a neuron. It is an electrical circuit consisting of a "leaky" resistor with conductance g in parallel with a capacitor C and a current synaptic source I. The membrane potential V(t) follows the differential equation:

$$C\frac{dV}{dt} = -g(V - E) + I \tag{2}$$

When V exceeds a threshold, a spike is obtained, and V is reset to E.

2.2. Coding schemes

Converting input pixels into spikes is done using one of the following coding schemes: rate coding, TTFS coding, phase coding, and burst coding.

We have used the rate coding scheme, where each value is converted into a Poisson spike train.

2.3. Spike-time-dependent plasticity learning

STDP learning [17] is a method used for unsupervised learning. It is based on the behavior noted in biological neurons: When a postsynaptic spike follows a presynaptic spike, the synaptic weight is strengthened. In contrast, if a postsynaptic spike precedes a presynaptic spike, the synaptic weight is weakened.

$$W_{new} = W_{old} + \Delta W \tag{3}$$

$$where\ \ \Delta W = \begin{cases} Aexp^{\frac{\Delta t}{t1}} & if\ \ \Delta t\ negative \\ Bexp^{\frac{\Delta t}{t2}} & if\ \ \Delta t\ positive \end{cases} \tag{4}$$

$$\Delta t\ negative:\ \ A\ postsynaptic\ spike\ happens\ prior\ to\ the\ presynaptic\ spike. \tag{5}$$

$$\Delta t\ positive:\ \ postsynaptic\ spike\ occurs\ after\ the\ presynaptic\ spike. \tag{6}$$

The weight adjustments are not based on whether the actual output is equal to the desired output. But over time, every neuron will be trained to recognize certain data patterns.

For the implementation of the STDP, the weight update formula is represented by:

$$\Delta W = \eta(x_{\text{pre}} - x_{\text{tar}})(W_{\max} - x_{\text{pre}})^{\mu} \tag{7}$$

Where:

$\tau_{x\text{pre}}\frac{dx_{\text{pre}}}{dt} = -x_{\text{pre}}$ which can be approximated by:

$$\tau_{\text{xpre}}(x_{\text{pre}}(t) - x_{\text{pre}}(t-1)) = -x_{\text{pre}}(t) \tag{8}$$

This is a nonlinear learning rule for updating a synaptic weight W, which depends on the presynaptic activity x_pre, a target value x_tar, and a maximum weight W_max.

η: Learning rate controls how quickly the weight changes.

x_pre: Presynaptic activity represents firing rate or signal strength from a presynaptic neuron.

x_tar: Target presynaptic activity at the desired or homeostatic level.

W_max: Maximum allowed weight or upper limit.

μ: Exponent controlling nonlinearity determines how sharply the update is influenced as x_pre approaches W_max.

If x_pre > x_tar, the weight may increase (assuming W_max − x_pre > 0).

The term (W_max − x_pre)^μ prevents the weight from growing indefinitely.

If x_pre < x_tar, the weight may decrease, depending on the rest of the expression.

2.4. Simple SNN architecture with supervised learning

Figure 1 shows the supervised version of a simple SNN. The weights are adjusted based on the difference between the actual output and the target spikes (impulses).

$$E = \left(\sum \delta_{actual} - \sum \delta_{desired}\right)^2 \tag{9}$$

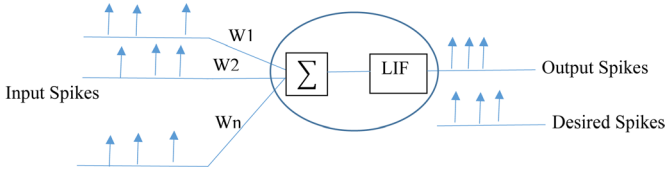
2.5. Simple SNN architecture with supervised learning using backpropagation

The backpropagation algorithm uses the following gradient descent rule for the weight update.

$$W_{\text{new}} = W_{\text{old}} - \alpha\frac{\partial E}{\partial W} \tag{10}$$

This weight update formula requires the calculation of the derivative of the error, which in turn requires the calculation of the derivative of impulses. Because the impulses are practically non-differentiable, different techniques were used to estimate this calculation. The surrogate gradient technique is one of the most common ones.

Figure 1
Supervised version of a simple SNN



2.6. Fuzzy ARTMAP's hyper-rectangle

The ARTMAP architecture consists of three components: two Fuzzy ART networks, ARTa and ARTb, along with an inter-ART operator. Training the network entails inputting data into ARTa and their corresponding targets into ARTb. In ARTa, the data is clustered, and these clusters receive labels from ARTb. Similar data points with the same target group are grouped to form a rectangle that acts as the prototype generated by ARTa. Fuzzy ART is a clustering technique designed for processing continuous data. The system features an input field F1 where data is entered and an output field F2 where the results of the clustering are displayed. Each node *j* in the F2 field is linked to all nodes *i* in the F1 field via a top-down weight represented as *wji*. The collection of top-down weights from node *j* in the F2 field is referred to as W_j, acting as the prototype for cluster *j*. For *p* input data points I1, I_2,…, I_p that are part of cluster *j*, W_j can be defined as follows:

$$w_j = I_1 \wedge I_2 \wedge \cdots I_p \tag{11}$$

It is important to highlight that before being fed into Fuzzy ART, certain preprocessing of the input patterns occurs. The initial preprocessing stage receives a Ma-dimensional input pattern related to the task of pattern clustering and transforms it into an output vector a = (a1,..., aMa), where each element is limited to the range [0, 1]. In the next preprocessing phase, the output a from the first stage is utilized to produce an output vector I in a manner that

$$I = (a, a^c) = \left(a_1, \ldots a_{Ma}, a_1^c, \ldots, a_{Ma}^c\right) \tag{12}$$

Where:

$$a_i^c = 1 - ai \; 1 \le i \le Ma \; ; \tag{13}$$

For *p* patterns:

$$I_1 = (a_1, a_1^c), \ldots, I_p = \left(a_p, a_p^c\right) \tag{14}$$

W_j can be written as follows:

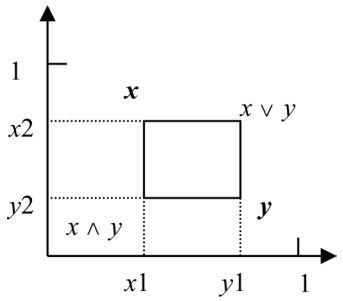
$$w_j = \left(\wedge_{i=1}^P a_i, \wedge_{i=1}^P a_i^c\right) = \left(\wedge_{i=1}^P a_i, \left\{\vee_{i=1}^p a_i\right\}^c\right) w_j = \left(u_j, \{v_j\}^c\right) \tag{15}$$

where $u_j = \wedge_{i=1}^P a_i$ and $v_j = V_{i=1}^P a_i$

The operand $\wedge$ is referred to as the fuzzy-min, and the operand $\vee$ is referred to as the fuzzy-max. The process is shown in Figure 2 for two 2-dimensional vectors, $x = (x1, x2)$ and $y = (y1, y2)$.

$I1 = (x1, x2, 1 - x1, 1 - x2)$; $I2 = (y1, y2, 1 - y1, 1 - y2)$.
$W_j = I1 \wedge I2$;
$W_j = (\min(x1, y1), \min(x2, y2), \max(1 - x1, 1 - y1), \max(1 - x2, 1 - y2))$
$W_j = (x1, y2, 1 - y1, 1 - x2) = (x \wedge y, 1 - (x \vee y))$

Figure 2
Fuzzy minimum and fuzzy maximum operations



The weight vector W_j can be expressed through two vectors u_j and v_j, which act as the vertices of a rectangle R_j.

In Figure 3, there is a rectangle defined by endpoints u_j and v_j that encloses the datapoints.

Given a test input $I_r = (a_r, (a_r)^c)_)$

The following equation is used to determine the bottom-up inputs:

$$T_j(I_r) = \frac{|I_r \wedge w_j|}{\alpha_\alpha + |w_j|} \tag{16}$$

Calculate the value of T_j for all the output nodes *j*. The node jmax which will give the maximum value of T_j will be considered the winner.

$$T_{j\max}(I_r) = \max_{1 \le j \le N} T_j(I_r) \tag{17}$$

$$N \; : \; \text{number of clusters formed.} \tag{18}$$

The label of the data I_r will correspond to the label that node *j*max was assigned following the training phase.

2.6.1. Simple illustration of SNN-ART

In the training phase, the basic idea is that inputs with the same Desired Class will produce a collection of output spike trains. These output spike trains will be mapped to one or more output ART categories corresponding to that Desired Class. A mapping will be formed between each desired Class with a hyper-rectangle containing all the output spike trains. In the test phase, any input with the same desired class will produce an output spike train belonging to that hyper-rectangle.

For illustration given 4 inputs belonging to 2 classes.

*I*1 and *I*2 belong to Class1 with desired spikes 00
*I*3 and *I*4 belong to Class2 with desired spikes 01

After applying these 4 inputs to the SNN-ART network and suppose we get the following output spikes for each input.

1110 with desired spikes 00
1101 with desired spikes 00
1001 with desired spikes 01
1010 with desired spikes 01

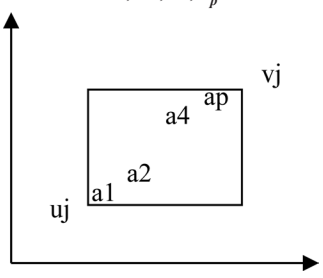
Doing complement coding on the output spikes we get:

11100001 with desired spikes 00
11010010 with desired spikes 00
10010110 with desired spikes 01
10100101 with desired spikes 01

After training, the ART categories are characterized by the following weights

*W*1 = (1,1,0,0,0,0,1,1) and *W*2 = (1,0,0,0,0,1,1,1)

Figure 3
The rectangle R_j, defined by its endpoints u_j and v_j, encompasses *I*1, *I*2,…, I_p



$W1$ is mapped to the desired spike 00 and $W2$ is mapped to the desired spike 01

For testing, applying the first input $I1$ to SNN-ART will give an output spike: 1110

After complement coding we get 11100001

Calculating

$$T_j(I_r) = \frac{|I_r \wedge w_j|}{\alpha_\alpha + |w_j|} \tag{19}$$

for $j = 1$ and $j = 2$, and for $\alpha = 0.001$

$$T1 = \frac{3}{4.001} \quad \text{and } T2 = \frac{2}{4.001} \tag{20}$$

Because $T1 > T2$ then the output of the ART will be the desired spike of $W1$ which is 00.

The same procedure can be applied to all the test inputs.

3. SNN-ARTMAP Network

3.1. Our weight updates of a simple SNN architecture

Figure 4 illustrates a basic Spiking Neural Network (SNN). Every input corresponds to a Poisson Spike train, with the frequency of the spikes being directly related to its magnitude. Each produced spike train is scaled by the synaptic weight. The total of the adjusted spike trains is computed and sent to the Leaky Integrate-and-Fire (LIF) neuron, which produces the output spike train.

$$\text{Out}(t) = \sum_{i=1}^{n} W_i \sum_{j=0}^{t} \delta(t - j) \tag{21}$$

$$\text{Output}(t) = \begin{pmatrix} 1 & \text{if Out}(t) \geq \text{Thres} \\ 0 & \text{else} \end{pmatrix} \tag{22}$$

These two equations show that increasing the weights W_i has the effect of increasing Out(t), which will make Output(t) have a value of 1 (Output spike). And decreasing W_i has the opposite effect.

Our output weight update strategy is the following:

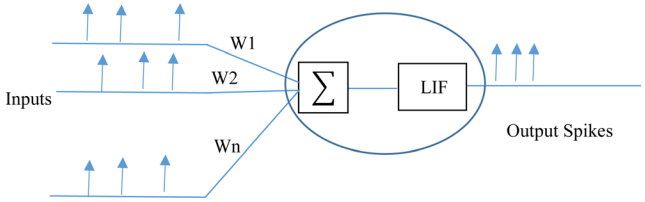
1) If the desired output is 1, then increase W_i (α is a small positive number).

Increasing W_i will increase Out(t) until it becomes greater than or equal to Thresh. A spike is generated

$$W_i(t) = W_i(t) + \alpha . \max(0,\ \text{Thresh} - \text{Out}(t)) \tag{23}$$

We want the neuron to spike, that is, have an output ≥ Thresh. If the current output is too low, i.e., Out(t) < Thresh, then the update amount is positive. This rule increases the weight Wi, moving the neuron's output closer to the threshold. α is a small positive learning rate. The use of max(0, ...) ensures no change if the output already exceeds the threshold.

Figure 4
Simple SNN network



2) If the desired output is 0, then decrease W_i (away from Thresh).

$$W_i(t) = W_i(t) + \alpha . \text{Out}(t) \tag{24}$$

We want the neuron not to spike, so we want the output to stay below the threshold. If the output is high, this update reduces the weight Wi, decreasing the neuron's response. The more the current output Out(t), the larger the weight reduction. α is a learning rate that regulates the magnitude of the update.

3.2. Simple SNN architecture followed by an ART-MAP module

Figure 5 shows a simple SNN-ARTMAP network. It consists of 3 inputs and 1 output.

Any input from the same Desired Class will produce an output spike train that falls within that hyper-rectangle in the training algorithm. Figure 6 illustrates this concept.

Output spikes from C12 are mapped to R3, while those from C2 are mapped to R2. Both R1 and R3 correspond to the same Desired Class.

3.3. SNN-ARTMAP network

Figure 7 shows a more complex SNN-ARTMAP network. It consists of 2 inputs, 3 hidden neurons and 2 outputs.

The hidden weights W_h are updated using the following STDP rule.

$$W_{\text{hnew}} = W_{\text{hold}} + \Delta W \tag{25}$$

Figure 5
Simple SNN-ARTMAP network

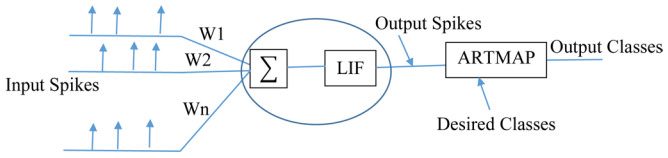


Figure 6
ART rectangles

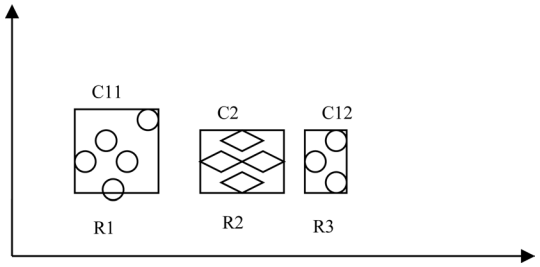
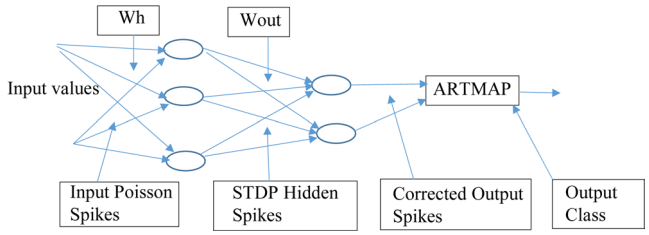


Figure 7
2-3-2 SNN-ARTMAP network



$$\text{where } \Delta W = \begin{cases} A\exp^{\frac{\Delta t}{t1}} & \text{if } \Delta t \text{ negative} \\ B\exp^{\frac{\Delta t}{t2}} & \text{if } \Delta t \text{ positive} \end{cases} \quad (26)$$

The output corrected weights Wout are updated using the following rule:

If the desired output is 1, then

$$W_{\text{out}}(t) = W_{\text{out}}(t) + \alpha.\max(0,\ \text{Thresh} - \text{Out}(t)) \quad (27)$$

Else

$$W_{\text{out}}(t) = W_{\text{out}}(t) - \alpha.\text{Out}(t) \quad (28)$$

It is important to highlight that the complexity of this network can be enhanced by raising the quantity of hidden neurons and the number of hidden layers.

4. Simulation Results

To evaluate the effectiveness of the SNN-ARTMAP algorithm, simulations are conducted using datasets from the UCI repository. We have used the following 2 datasets:

1) The Iris dataset consists of 50 samples representing three distinct species of Iris: Iris setosa, Iris virginica, and Iris versicolor. For each sample, four characteristics were documented, namely the lengths and widths of the sepals and petals, measured in centimeters. A selection of 10 random samples from each species is used as the training set, resulting in a total of 30 samples, while the remaining 40 samples constitute the test set, amounting to a total of 120 samples.

The structure of the network includes 4 input nodes, 3 output nodes, and a varying number of hidden neurons.

Table 1 compares our algorithm with other similar spiking networks.

2) The Wisconsin breast cancer dataset contains 699 instances. It is categorized into benign and malignant cases. Each instance consists of nine measurements, each represented by an integer ranging from 1 to 10. A group of ten random samples from each category was selected to serve as the training set, with the remaining samples designated for the testing set. As a result, there were 20 samples allocated for training and 679 for testing. The network architecture consists of 9 inputs, 2 outputs, and a different number of hidden neurons.

Table 2 compares our algorithm with other similar spiking networks.

Table 1
Classification results on the *Iris* dataset

Method	% Testing accuracy
SpikeProp [18]	96.1
SWAT [19]	95.3
multi-ReSuMe [20]	94.0
Multi-STIP [21]	96.7
SNN-ARTMAP (10 hidden)	96.1
SNN-ARTMAP (20 hidden)	96.9

Tables 1 and 2 present a comparative analysis of various spiking neural network-based methods regarding testing accuracy. Of the six techniques evaluated, the SNN-ARTMAP with 20 hidden neurons achieves the highest testing accuracy, exceeding that of all other models. This suggests that enlarging the number of hidden neurons within the SNN-ARTMAP structure boosts its performance in pattern recognition, likely due to its enhanced capacity for representation. The Multi-STIP approach closely follows, demonstrating strong performance. Multi-STIP is known for its effectiveness in modeling temporal dynamics, which may explain its robust results. A more modest version of SNN-ARTMAP (with 10 hidden neurons) performs competitively against established methods like SpikeProp, a well-known and effective gradient-based learning algorithm for SNN. The SWAT and multi-ReSuMe approaches show slightly lower performance levels. These techniques might be limited by their learning methodologies or their generalization capabilities compared to the others. Notably, multi-ReSuMe, which employs the ReSuMe learning rule for supervised training of Spiking Neural Networks, underperforms relative to the other models, indicating potential challenges in its ability to map temporal patterns effectively or in scaling up the network. In conclusion, the results suggest that SNN-ARTMAP, especially with an adequate number of hidden neurons, stands out as a promising and scalable framework for achieving high-accuracy classification tasks within spiking neural networks. Its ability to either outperform or match traditional approaches highlights its robustness and flexibility.

Table 2
Classification results on the breast dataset

Method	% Testing accuracy
SpikeProp [18]	97.0
SWAT [19]	96.7
multi-ReSuMe [20]	94.8
Multi-STIP [21]	97.1
SNN-ARTMAP (20 hidden)	97.0
SNN-ARTMAP (50 hidden)	97.8

Table 3
Classification results of our algorithm compared to the two-layer FE using different datasets from the UCI repository

Dataset	Method	% Testing accuracy
Iris	Two layer FE [1]	98
Training Samples: 75	SNN-ARTMAP (10 hidden)	97.6
Testing Samples: 75	SNN-ARTMAP (20 hidden)	98.3
Breast Cancer	Two layer FE [1]	97.5
Training Samples: 349	SNN-ARTMAP (10 hidden)	97
Testing Samples: 350	SNN-ARTMAP (20 hidden)	98
Liver Disorders	Two layer FE [1]	64.8
Training Samples: 172	SNN-ARTMAP (10 hidden)	64.5
Testing Samples: 173	SNN-ARTMAP (20 hidden)	65.3
Pima Diabetes	Two layer FE [1]	72.5
Training Samples: 384	SNN-ARTMAP (10 hidden)	72.3
Testing Samples: 384	SNN-ARTMAP (20 hidden)	72.7

Figure 8
Graphical representations of Table 3

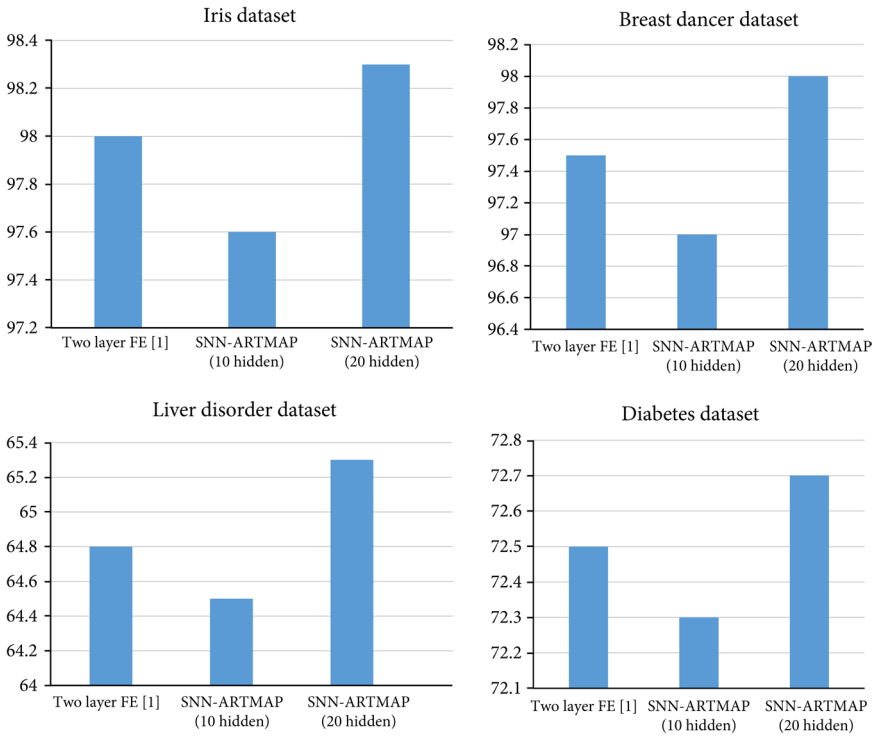


Table 3 shows the classification results of our algorithm compared with those shown in Reference [1]. It can be concluded that the integrated SNN-ARTMAP increases the recognition accuracy of the SNN.

Table 3 can be illustrated graphically in Figure 8.

Figure 8 highlights the superiority of the SNN-ARTMAP in comparison to the two-layer feature extraction (FE) network models that utilize 10 and 20 hidden neurons across four standard datasets: Iris, Breast Cancer, Liver Disorders, and Pima Diabetes. Iris Dataset: With 75 samples for training and another 75 for testing, all three models exhibit remarkable performance. The SNN-ARTMAP with 20 hidden neurons achieves the highest accuracy at 98.3%, slightly outpacing the Two-layer FE at 98% and the SNN-ARTMAP with 10 hidden neurons at 97.6%. These notable results suggest that all models effectively grasp the relatively simple patterns present in the Iris dataset. The benefits of increasing the number of hidden neurons indicate that even a minor architectural change can lead to improved classification outcomes. Breast Cancer Dataset: In this more complex medical dataset, which comprises 349 training samples and 350 testing samples, the performance remains strong. The SNN-ARTMAP with 20 hidden neurons again leads with an accuracy of 98%, surpassing the Two-layer FE at 97.5% and the 10 hidden neuron model at 97%. This improvement reflects the advantages of having greater capacity to handle larger, more varied datasets, allowing the network to better define decision boundaries. Liver Disorders Dataset: In contrast to the earlier two datasets, the classification performance on the Liver Disorders dataset is significantly lower. The SNN-ARTMAP with 20 hidden neurons achieves the highest score of 65.3%, which is slightly above the Two-layer FE at 64.8% and the SNN-ARTMAP with 10 hidden neurons at 64.5%. These minor differences indicate that the models encounter difficulties due to the intrinsic complexity or noise in this dataset, suggesting that enhanced feature engineering or different modeling techniques may be necessary to achieve more dependable results in this instance. Pima Diabetes Dataset: The performance of all three models is fairly similar for this dataset: 72.7% for the SNN-ARTMAP (20 hidden), 72.5% for the Two-layer FE, and 72.3% for the SNN-ARTMAP (10 hidden). These results demonstrate the relatively balanced complexity of the Pima Diabetes dataset and suggest that SNN-ARTMAP models can at least match, if not slightly exceed, traditional FE-based approaches. Once again, the advantage of extra hidden neurons is slight yet consistently observed.

To demonstrate the performance of our algorithm in relation to the leading algorithm, Table 4 presents the classification outcomes of our method against the plastic synaptic weights and delays [22] across various datasets from the UCI repository. Comparable network architectures are illustrated in Table 4.

Table 4
Classification results of our algorithm compared to plastic synaptic weights and delays using different datasets from the UCI repository

Dataset	Method	% Testing accuracy
Iris	25-10-3 network [22]	97
	SNN-ARTMAP (25 hidden)	98.3
Breast Cancer	55-15-2 network [22]	97.9
	SNN-ARTMAP (55 hidden)	98.4
Liver Disorders	37-15-2 network [22]	66.7
	SNN-ARTMAP (37 hidden)	67.3
Pima Diabetes	55-20-2 network [22]	77
	SNN-ARTMAP (55 hidden)	78.7

5. Conclusion

In this paper, a novel training algorithm for Spiking Neural Networks, which we call SNN-ARTMAP, is presented. SNN-ARTMAP is a novel error-based spiking learning algorithm. This network can have multiple hidden layers, a corrected layer, and an ARTMAP module. While the hidden layers are trained according to STDP, the corrected layer is trained according to the target class output. The output spikes are fed into an ART network, which will create spike train hyper-rectangles. Each hyper-rectangle is mapped to the correct target class. It is shown that this algorithm outperforms other similar spiking algorithms. SNN-ARTMAP consistently outperforms or equals both other models across all datasets, highlighting its adaptability and effectiveness.

In challenging datasets like Liver Disorders, none of the models perform exceptionally well, highlighting the necessity for either improved preprocessing techniques or alternative learning approaches.

Future work can be dedicated to integrating SNN-ARTMAP with attention systems [23].

Ethical Statement

This study does not contain any studies with human or animal subjects performed by the author.

Conflicts of Interest

The author declares that he has no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in the UC Irvine Machine Learning Repository at https://archive.ics.uci.edu/.

Author Contribution Statement

Issam Dagher: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

[1] Luo, X., Qu, H., Wang, Y., Yi, Z., Zhang, J., & Zhang, M. (2023). Supervised learning in multilayer spiking neural networks with spike temporal error backpropagation. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(12), 10141–10153. https://doi.org/10.1109/TNNLS.2022.3164930

[2] Rathi, N., & Roy, K. (2023). DIET-SNN: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(6), 3174–3182. https://doi.org/10.1109/TNNLS.2021.3111897

[3] El Maachi, S., Chehri, A., & Saadane, R. (2024). Efficient hardware acceleration of spiking neural networks using FPGA: Towards real-time edge neuromorphic computing. In *2024 IEEE 99th Vehicular Technology Conference*, 1–5. https://doi.org/10.1109/VTC2024-Spring62846.2024.10683049

[4] Zhou, C., Zhang, H., Yu, L., Ye, Y., Zhou, Z., Huang, L., ..., & Tian, Y. (2024). Direct training high-performance deep spiking neural networks: A review of theories and methods. *Frontiers in Neuroscience*, *18*, 1383844. https://doi.org/10.3389/fnins.2024.1383844

[5] Guo, W., Fouda, M. E., Eltawil, A. M., & Salama, K. N. (2023). Efficient training of spiking neural networks with temporally-truncated local backpropagation through time. *Frontiers in Neuroscience*, *17*, 1047008. https://doi.org/10.3389/fnins.2023.1047008

[6] Yang, S., Linares-Barranco, B., & Chen, B. (2022). Heterogeneous ensemble-based spike-driven few-shot online learning. *Frontiers in Neuroscience*, *16*, 850932. https://doi.org/10.3389/fnins.2022.850932

[7] Saranirad, V., Dora, S., McGinnity, T. M., & Coyle, D. (2025). CDNA-SNN: A new spiking neural network for pattern classification using neuronal assemblies. *IEEE Transactions on Neural Networks and Learning Systems*, *36*(2), 2274–2287. https://doi.org/10.1109/TNNLS.2024.3353571

[8] Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., & Luo, Z.-Q. (2022). Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12444–12453.

[9] Hao, Y., Huang, X., Dong, M., & Xu, B. (2020). A biologically plausible supervised learning method for spiking neural networks using the symmetric STDP rule. *Neural Networks*, *121*, 387–395. https://doi.org/10.1016/j.neunet.2019.09.007

[10] Gouda, M., Abreu, S., & Bienstman, P. (2024). Surrogate gradient learning in spiking networks trained on event-based cytometry dataset. *Optics Express*, *32*(9), 16260–16272. https://doi.org/10.1364/OE.518323

[11] Bensimon, M., Greenberg, S., Ben-Shimol, Y., & Haiut, M. (2021). A new SCTN digital low power spiking neuron. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *68*(8), 2937–2941. https://doi.org/10.1109/TCSII.2021.3065827

[12] Frenkel, C., Lefebvre, M., Legat, J.-D., & Bol, D. (2019). A 0.086-mm^2 $12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS. *IEEE Transactions on Biomedical Circuits and Systems*, *13*(1), 145–158. https://doi.org/10.1109/TBCAS.2018.2880425

[13] Meng, Q., Yan, S., Xiao, M., Wang, Y., Lin, Z., & Luo, Z.-Q. (2022). Training much deeper spiking neural networks with a small number of time-steps. *Neural Networks*, *153*, 254–268. https://doi.org/10.1016/j.neunet.2022.06.001

[14] Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., & Gu, S. (2021). Differentiable spike: Rethinking gradient-descent for training spiking neural networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, *34*, 23426–23439.

[15] Tashiro, K., Masuyama, N., & Nojima, Y. (2024). A growing hierarchical clustering algorithm via parameter-free adaptive resonance theory. In *2024 International Joint Conference on Neural Networks*, 1–6. https://doi.org/10.1109/IJCNN60899.2024.10650253

[16] Dagher, I., Georgiopoulos, M., Heileman, G. L., & Bebis, G. (1999). An ordering algorithm for pattern presentation in fuzzy ARTMAP that tends to improve generalization performance. *IEEE Transactions on Neural Networks*, *10*(4), 768–778. https://doi.org/10.1109/72.774217

[17] Iakymchuk, T., Rosado-Muñoz, A., Guerrero-Martínez, J. F., Bataller-Mompeán, M., & Francés-Víllora, J. V. (2015). Simplified spiking neural network architecture and STDP learning algorithm applied to image classification. *EURASIP Journal on Image and Video Processing*, *2015*(1), 4. https://doi.org/10.1186/s13640-015-0059-4

[18] Bohte, S. M., Kok, J. N., & la Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*(1–4), 17–37. https://doi.org/10.1016/S0925-2312(01)00658-0

[19] Wade, J. J., McDaid, L. J., Santos, J. A., & Sayers, H. M. (2010). SWAT: A spiking neural network training algorithm for classification problems. *IEEE Transactions on Neural Networks*, *21*(11), 1817–1830. https://doi.org/10.1109/TNN.2010.2074212

[20] Sporea, I., & Grüning, A. (2013). Supervised learning in multilayer spiking neural networks. *Neural Computation*, *25*(2), 473–509. https://doi.org/10.1162/NECO_a_00396

[21] Lin, X., Wang, X., & Hao, Z. (2017). Supervised learning in multilayer spiking neural networks with inner products of spike trains. *Neurocomputing*, *237*, 59–70. https://doi.org/10.1016/j.neucom.2016.08.087

[22] Wang, J. (2024). Training multi-layer spiking neural networks with plastic synaptic weights and delays. *Frontiers in Neuroscience*, *17*, 1253830. https://doi.org/10.3389/fnins.2023.1253830

[23] Zhou, Z., Niu, J., Zhang, Y., Yuan, L., & Zhu, Y. (2025). Spiking transformer with spatial-temporal spiking self-attention. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1–5. https://doi.org/10.1109/ICASSP49660.2025.10890026

How to Cite: Dagher, I. (2026). Incorporating Fuzzy ARTMAP Network with Spiking Neural Networks. *Artificial Intelligence and Applications*, *4*(2), 248–255. https://doi.org/10.47852/bonviewAIA52025930