SpikeComp: An Evolving Spiking Neural Network with Adaptive Compact Structure for Pattern Classification

Jinling Wang[1(✉)], Ammar Belatreche[1], Liam P. Maguire[1], and T. Martin McGinnity[1,2]

[1] Intelligent Systems Research Centre (ISRC), Faculty of Computing and Engineering, University of Ulster, Magee Campus, Northland Road, Derry, BT48 7JL, UK
Wang-J1@email.ulster.ac.uk,
{A.Belatreche,LP.Maguire}@ulster.ac.uk

[2] School of Science and Technology, Nottingham Trent University, Nottingham, UK
TM.McGinnity@ulster.ac.uk

Abstract. This paper presents a new supervised learning algorithm (SpikeComp) with an adaptive compact structure for Spiking Neural Networks (SNNs). SpikeComp consists of two layers of spiking neurons: an encoding layer which temporally encodes real valued features into spatio-temporal spike patterns, and an output layer of dynamically grown neurons which perform spatio-temporal pattern classification. The weights between the neurons in the encoding layer and the new added neuron in the output layer are initialised based on the precise spiking times in the encoding layer. New strategies are proposed to either add a new neuron, or update the network parameters when a new sample is presented to the network. The proposed learning algorithm was demonstrated on several benchmark classification datasets and the obtained results show that SpikeComp can perform pattern classification with a comparable performance and a much compact network structure compared with other existing SNN training algorithm.

Keywords: Spiking neurons · Supervised learning · Adaptive structure · Classification

1 Introduction

Artificial neural networks (ANNs) mimic the brain's ability to solve complex tasks, such as face recognition and object identification. The frequency of the spikes is used to encode information. However, existing studies have shown that biological neurons use precise spiking times to encode information [1]. Spiking neural networks (SNNs) use computational neural models which capture the firing dynamics of biological neurons [2]. SNNs offer a more biologically plausible model compared with their classical counterparts and are capable of handling spatio-temporal data. SpikeProp is an adaptation of the classical backpropagation algorithm which minimize the error between the predicted and desired spiking times. It can achieve comparable performance on several benchmark datasets to rate-coded networks [3]. However, it is slow when used in an online setting.

S. Arik et al. (Eds.): ICONIP 2015, Part II, LNCS 9490, pp. 259–267, 2015.
DOI: 10.1007/978-3-319-26535-3_30

Belatreche et al. [4] proposed a derivative-free supervised learning algorithm where an evolutionary strategy (ES) is used to minimise the error between the output firing times and the corresponding desired firing times. This algorithm achieves a better performance than SpikeProp. However, an ES-based iterative process makes the training procedure extremely time-consuming and is not suitable for online learning. It is important to note that fixed network architectures were employed in the above-mentioned approaches. For maximum adaptability, which is needed in a changing environment or when dealing with streams of data, it is desirable that both the network structure and its weights adapt to new data.

In [5] an evolving approach for SNNs (eSNN) has been presented. The learning rule is based on the rank order of the spiking times. The eSNN has successfully been used for on-line spatio-and spectro-temporal pattern recognition. We have developed an RBF-like evolving leaning algorithm for SNN in [6], the centre of each added hidden RBF spiking neuron is represented by its time to first spike, the adaptive SNN is able to classify spike-based spatio-temporal inputs after just one presentation of the training set. Recently, a Self Regulating Evolving SNN has been proposed for handling classification problems in [7]. The evolving algorithm utilizes the association between the spiking outputs of differing classes during training. These associations take a vital role in pattern classification tasks.

This paper presents a new supervised learning algorithm (called SpikeComp) with an adaptive compact structure for Spiking Neural Networks (SNNs). Extended from [6], SpikeComp assumes that each added output neuron has a centre which is represented by its time to first spike; there is an accommodation field around the centre. New strategies are proposed for adding a new neuron, updating the network weights, or just updating the neurons centre when a new sample is presented to the network. These strategies depend on whether the spiking time of a sample lies inside the accommodation field of a neuron centre and whether the winner output neurons are associated or not with the same class as the incoming input sample. SpikeComp has been validated on several benchmark datasets; the results show that using a much compact network, SpikeComp can achieve comparable performance using compared with other machine learning algorithms and the existing eSNN.

The remainder of this paper is structured as follows: Sect. 2 describes the employed neural model, the employed temporal encoding scheme, the SNN structure design and the proposed learning strategies. Section 3 presents experimental results for training SNNs on selected benchmark datasets from the UCI Machine Learning Repository. Finally Sect. 4 concludes the paper.

2 Learning and Structural Adaptation

2.1 Spiking Neural Model

The proposed SpikeComp employs the Integrate-and-Fire (*IF*) neurons in the output layer. The postsynaptic potential (*PSP*) of an output neuron i at time t relies on the spike times received from neurons in the encoding layer and can be described as:

$$PSP(i,t) = \sum_{j\epsilon[1N]} \mathrm{W}_{ji} \exp\left(-\frac{t_j}{\tau}\right) \quad (1)$$

Where $j \in [1, \mathrm{N}]$ represents the j^{th} incoming connection, and N is the total number of incoming connections between the encoding layer and the output neuron i; t_j represents the precise spiking time of the j^{th} encoding neuron; τ is a time constant and determines the range for which synaptic strengthening occurs; W_{ji} is the synaptic weight associated with the synaptic connection between output neuron i and encoding neuron j. If $PSP(i, t)$ is greater than the firing threshold of neuron i, denoted by, $PSP_{th}(i)$, then an output spike is produced at neuron i in the output layer, and the simulation for the current input sample is terminated and the PSP of firing output neuron is reset.

2.2 Information Encoding

Gaussian Receptive Field population encoding scheme, proposed by Bohte et al. [3], is used to encode continuous input variables into spike times. An example is shown in Fig. 1 where a real-valued feature of 5.1 (illustrated by a vertical dashed red line) is converted into spike times using eight Gaussian receptive fields. The resulting eight response values (y) are 0.0228, 0.4578, 0.9692, 0.2163, 0.0051, 0, 0 and 0 respectively. These values are then mapped linearly to spike times (see Eq. 2):

$$t = -9 * \mathrm{y} + 9 \quad (2)$$

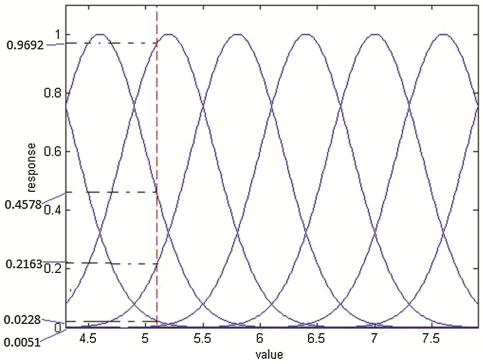

Fig. 1. Encoding of a real valued feature of 5.1 using 8 Gaussian receptive fields neurons.

If the resulting spike time is equal to 9 ms, this neuron is treated as 'silent', and is represented by a value of '−1'. The resulting spiking times are therefore represented by the following series of spiking times: 8.79 ms, 4.88 ms, 0.28 ms, 7.05 ms, 8.95, -1(silent), -1(silent) and -1(silent).

Time-to-first spike decoding is employed at the output layer where an input sample is considered to be correctly classified if the first spike is produced at an output neuron whose class label matches the class label of the current input sample; otherwise an input sample is considered to be incorrectly classified.

2.3 Network Topology

Figure 2 presents the proposed network topology that consists of a layer of encoding neurons and a layer of output neurons. Each neuron in the encoding layer provides a spike time which is fed to the next layer, and is fully connected to the neurons in the output layer. The number of neurons in this layer is determined by the dimensionality of the dataset (m) and the number of Gaussian receptive fields q, i.e., given by $m * q$. The set of spiking times represented by these neurons in the encoding layer is in the range of the time coding interval [0,T_{ref}] and $T_{ref} = 9$ ms.

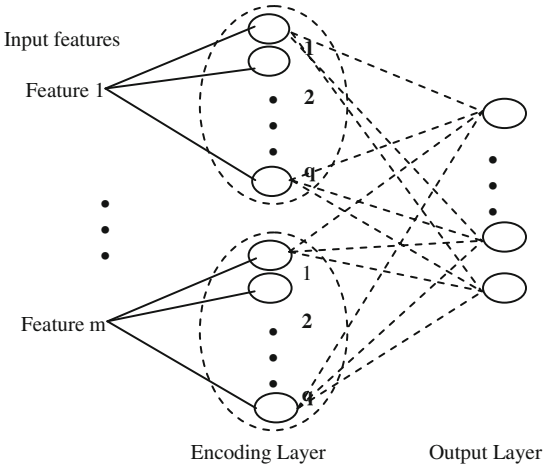

Fig. 2. A two layer feedforward SNN with adaptive structure.

The output layer has no output neurons at the beginning of the training process. A new output neuron is added dynamically when an incoming sample is received and is fully connected to the neurons in the encoding layer. Only one synaptic connection exists between every encoding neuron and every output neuron. In this study the simulation interval per sample is [0 10] ms.

2.4 Learning and Structure Adaptation

As a new sample is presented to the network, SpikeComp finds the two winner neurons which have the same class label (i) and differing class label (k) as the incoming sample, and their first spiking times t_{cc} and t_{mc}, respectively (as illustrated in Fig. 3). Every added output neuron has a centre which is set based on the time of the first spike of the added output neuron ($T_c(i)$, i represents the i^{th} added output neuron). In addition, we use two radius parameters, R_{cc} and R_{mc}, which represent accommodation boundaries when the neuron is a winner and has the same class label (R_{cc}) or a differing class label (R_{mc}) as the new incoming input sample, respectively. When added to the neuron centre value, i.e. $T_c + R_{cc} = Th_{cc}$, or $T_c + R_{mc} = Th_{mc}$, the resulting sums determine the neuron adding

thresholds (Th_{cc} and Th_{mc}) when the output neuron has the same class label or different class label as the incoming sample, respectively. Based on the associations among the first spiking times (t_{cc} and t_{mc}) of the two winner neurons and the neuron adding thresholds (Th_{cc} and Th_{mc}), SpikeComp adds a new neuron (neuron addition strategy), update the network weights (weights update strategy) or just update the centre (update centre strategy).

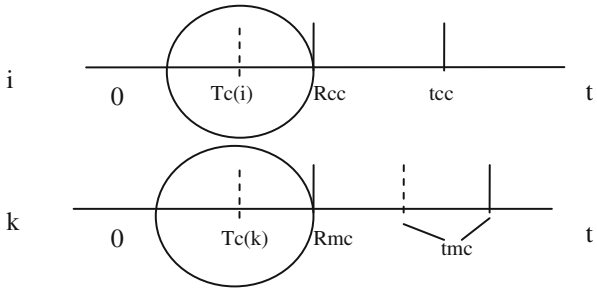

Fig. 3. Description of neuron addition with regard to classification or misclassification.

When a new output neuron (i) is created and its weights W_{ji} are initialised using Eq. 3;

$$w_{ji} = \exp\left(-\frac{t_j}{\tau}\right) \tag{3}$$

The maximum postsynaptic potential for this added output neuron ($PSP_{max}(i)$) is set when the training sample is propagated into the network, and is calculated using Eq. 1 when all potential input spikes have been used. The firing threshold of this added output neuron $PSP_{th}(i)$ for the postsynaptic potential in the network is determined by Eq. 4 as follows:

$$PSP_{th}(i) \leftarrow con * (PSP_{max}(i)) \tag{4}$$

Where $con \in [0, 1]$ and is set to 0.5 in this paper. The class label of this new added output neuron is set the same as that of this new sample.

Output Neuron Adding Strategy. SpikeComp adds a new output neuron during the training stage under three situations: (a) if an input sample is presented from a class that is different from those previously learned so an output neuron is added to cater for the new class. (b) As shown in Fig. 3, the output neuron i fires first ($t_{cc} < t_{mc}$, solid line for t_{mc} in neuron k), the spiking time (t_{cc}) of neuron i is greater than the adding threshold of neuron i, i.e., $t_{cc} < t_{mc}$ & $t_{cc} > Th_{cc}(i)$ or the output neuron k fires first ($t_{cc} > t_{mc}$, dashed line for t_{mc} in neuron k), the spiking time (t_{mc}) of neuron k is greater than the adding threshold of neuron k, i.e., $t_{mc} < t_{cc}$ & $t_{mc} > Th_{mc}(k)$. In these situations, a new output neuron is added. (c) if the winner output neuron (k) fires first, and its spiking time (t_{mc}) is less than the adding threshold of neuron k, i.e., $t_{mc} < t_{cc}$ & $t_{mc} < Th_{mc}(k)$, Hence, a new

neuron needs to be added to represent this new incoming sample. The weights of this winner neuron k are updated so that $t_{mc} > Th_{mc}(k)$ using Eqs. (5) and (6). Where η is the learning rate and is set to 0.025.

$$W_{jk} = W_{jk} + \eta * \left(\exp\left(-\frac{t_j}{\tau}\right) - W_{jk}\right) \quad j\varepsilon\,[1N]\,\&\forall\,(j) \in \psi_k \tag{5}$$

$$\psi_k = \{(j) \mid \exp\left(-\frac{t_j}{\tau}\right) \leq W_{jk}\} \tag{6}$$

It should be noted that only synapses between the neurons in the encoding layer and the neuron k with weights that are higher than a certain value, in this paper this value is set to $\exp\left(-\frac{t_j}{\tau}\right)$, are updated. This should reduce the computational cost as weight update is limited to a subset of synaptic connections. As a result, the corresponding high weights between the neurons in the encoding layer and the neuron k are decreased, leading to late firing of the neuron k.

Weights Update Strategy. If the criteria for a new output neuron addition are not met, then relevant weights are considered to be updated. (a) if the spiking time difference between the neuron i and the neuron k is sufficiently large, i.e., $M/2 < (t_{mc} - t_{cc})/t_{cc} < M$. Furthermore, the neuron i fires first and the spiking time (t_{cc}) of neuron i is less than the adding threshold of neuron i, i.e., $t_{cc} < t_{mc}$ & $t_{cc} < Th_{cc}(i)$, M represents the separation constant, determines when a neuron should be added or just weights are updated, then only the weights for the winner neuron (i) are updated based on Eqs. (7) and (8). It should be noted that only synapses between the neurons in the encoding layer and the winner neuron i with weights that are lower than a certain value (set to $\exp\left(-\frac{t_j}{\tau}\right)$ in this paper) are updated. As a result, the corresponding low weights between the neurons in the encoding layer and the neuron i are increased, leading to earlier spiking of the neuron i.

$$W_{ji} = W_{ji} + \eta * \left(\exp\left(-\frac{t_j}{\tau}\right) - W_{ji}\right) \quad j\varepsilon\,[1N]\ \&\ \forall\,(j) \in \psi_i \tag{7}$$

$$\psi_i = \{(j) \mid W_{ji} \leq \exp\left(-\frac{t_j}{\tau}\right)\} \tag{8}$$

or (b). In this case where $|t_{mc} - t_{cc}|/t_{cc} <= M$, the weights between the neurons in the encoding layer and the neuron i, and neuron k are updated to avoid misclassification. The weights between the neurons in the encoding layer and the neuron i are updated using (7) and (8), the weights between the neurons in the encoding layer and the neuron k are updated using (5) and (6). As a result, the weights connecting to neuron i are strengthened, while those connecting to neuron k are weakened. This will increase the difference between the first spike times of the winner neuron i and the winner competing neuron k. It should be noted that neuron addition strategy and weights update strategy are inspired from [7].

Neuron Centre Update Strategy. If the learning strategies for neuron addition and weights update mentioned above are not satisfied, the centre of neuron i needs to be updated using Eq. (9). The total number of samples used to train the neuron i, represented by N_i^{sample}, is then incremented. So the neuron adding threshold Th_{cc} and Th_{mc} for the neuron i are also updated to accommodate the current input sample.

$$T_c(i) \leftarrow \frac{T_c(i) * N_i^{sample} + t_{cc}}{N_i^{sample} + 1}. \tag{9}$$

3 Performance Evaluation of SpikeComp

In this section, the performance of the proposed SpikeComp is evaluated on the IRIS, Wisconsin Breast Cancer (*WBC*) and Ionosphere (*Ion*) benchmark datasets from the UCI machine learning repository. The datasets were each divided into two sets as outlined in Table 1, where the first *Tr* samples were used for training and the remaining *Ts* samples were used for testing. *T* represents the total number of samples, *c* is the number of classes. Table 1 also lists the values of the parameters *con*, τ, *M*, R_{cc} and R_{mc} that are related to the network parameter and structure learning for each dataset.

Table 1. Description of different datasets and their parameter values.

Dataset	T/Tr/Ts	m	c	q	N	con	τ	M	R_{cc}/R_{mc}
IRIS	150/30/120	4	3	10	40	0.5	25	0.1	2.5/2.0
WBC	683/40/643	9	2	8	72	0.5	30	0.2	2.5/2.5
Ion	351/100/251	33	2	7	231	0.5	30	0.1	2.5/2.5

The dynamic changes of the number of output neurons against the number of epochs during training for the IRIS, WBC and Ionosphere datasets are illustrated in Fig. 4. From Fig. 4 it could be seen that the network has grown a total of seven, fourteen and sixty-two neurons for IRIS, WBC and Ionosphere datasets, respectively.

Table 2 lists the classification accuracies obtained for each dataset and compares the classification performance of these datasets using SpikeComp algorithm with the Rank Order based eSNN method in [5] and the popular classical methods, namely KNN, SVM and MLP. *Acc_Tr/Acc_Ts* represents the training/testing classification accuracy; *Num_O* represents the total number of neurons in the output layer after training. From Table 2, it can be observed that the results of SpikeComp are comparable to the Rank Order based eSNN method [5] and other machine learning algorithms. In addition, these results clearly show that SpikeComp adds much less output neurons than the Rank Order based eSNN method.

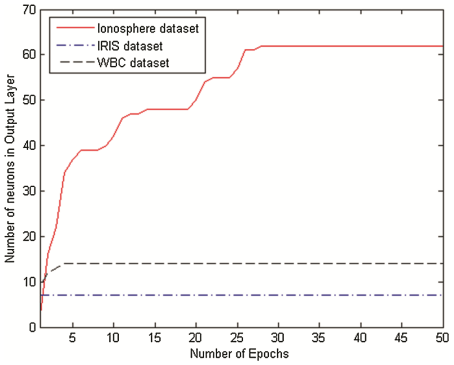

Fig. 4. Evolution of the output layer structure against the number of epochs during training for the IRIS, WBC and Ionosphere datasets.

Table 2. Comparison of the classification accuracy of SpikeComp with other approaches,

	SpikeComp/ (Num_o)	eSNNs/(Num_o)	MLP	KNN (k = 3)	SVM
Dataset	Acc_TR/Acc_TS (%)				
IRIS	100/**97.5**/(7)	100/95.0/(84)	100/92.7	96.0/92.0	100/96.0
WBC	97.5/96.7/(14)	99.6/**98.7**/(280)	98.0/88.5	97.3/98.5	96.6/98.5
Ion	96.0/91.2/(62)	81.6/74.4/(213)	100/78.3	88.6/**93.2**	100/90.3

4 Conclusion

In this paper, an Evolving Spiking Neural classifier, called SpikeComp, has been presented. SpikeComp employs a two layers feedforward network topology, and the Gaussian receptive fields population encoding is used to temporally encode real valued feature vectors into spatio-temporal spike patterns in the encoding layer. Output neurons that process spatio-temporal inputs from the encoding layer, are dynamically grown as new spatio-temporal spiking patterns are presented to the spiking neural network based on differing learning strategies. Evaluated on several benchmark tasks, the classification performance of SpikeComp has been compared with eSNN and other machine learning methods. The results clearly indicate that SpikeComp achieves comparable performance with much smaller network structure, but some datasets need more training epochs than other evolving SNNs methods. Future work will demonstrate these learning strategies on other neural models and a wider range of datasets.

References

1. Rullen, R., Van Guyonneau, R., Thorpe, S.J.: Spike times make sense. Trends Neurosci. **28**(1), 1–4 (2005)
2. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Netw. **10**(9), 1659–1671 (1997)
3. Bohte, S.M., Kok, J.N., Poutre, H.L.: Error-backprogation in temporally encoded networks of spiking neurons. Neurocomputing **48**, 17–37 (2002)
4. Belatreche, A., Maguire, L.P., McGinnity, T.M., Wu, Q.: Evolutionary design of spiking neural networks. J. New Math. Nat, Comput. **2**(3), 237–253 (2006)
5. Kasabov, N., Dhoble, K., Nuntalid, N., Indiveri, G.: Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. Neural Netw. **41**, 188–201 (2013)
6. Wang, J., Belatreche, A., Maguire, L.P., McGinnity, T.M.: An online supervised learning method for spiking neural networks with adaptive structure. Neurocomputing **144**, 526–536 (2014)
7. Dora, S., Subramanian, K., Suresh, S., Sundararajan, N.: Development of a self regulating evolving spiking neural network for classification problem. Neurocomputing (2015, in press)