Analysis of using an ensemble of Evolutionary Algorithms to train Spiking Neural Networks

Anirudh Shankar
Graduate Student
Purdue University
shanka19@purdue.edu

***Abstract*—The objective of pursuing this project is to analyse a novel method of combining evolutionary algorithms in order to improve the accuracy of a Spiking Neural Network for a classification task. Two experiments are conducted both employing evolutionary algorithms in different settings and the respective results are discussed.**

***Index Terms*—Ensemble Learning, Evolutionary Algorithms, Spiking Neural Networks**

I. Introduction

Ensemble Learning is a popular technique that has been used to construct a set of diverse classifiers for classification using a weighted combination of predictions from various classifiers [1]. Predictions from a diverse set of classifiers are usually combined in a weighted manner with the aim of improving the prediction power on the test data. The three most popular ensemble learning methods are bagging, boosting and stacking. Bagging is an ensemble technique where a single classifier is trained on various bootstrap samples and the predictions are then aggregated by voting [2]. Boosting is a technique which aims to iteratively improve the prediction accuracy by manipulating the training set to include incorrectly classified examples in the training data at every step [3]. Stacking is a technique where a number of classifiers are stacked either using weights or using a meta learner to yield better predictions [4]. Ensemble members/classifiers can be combined using a variety of techniques. Arithmetic measures (like sum or average) or voting can be used to combine results obtained from various classifiers [5]. The different types of combining techniques include the following:

Majority Voting comes in a variety of flavours which include simple majority voting, where more than 50% of classifiers agree on a particular class or unanimous voting where the result is the class that most classifiers agree upon [5] A variation of majority voting is weighted majority voting. The technique is similar to majority voting, except that all the classifiers do not have equal contribution towards the final classification result. Usually the weights are normalized so that they add up to one [5].

If a classifier is able to output probabilities for each class, then these probabilities can be combined to yield a final result. The probabilities maybe combined by using averaging or a weighted combination where the weights add upto 1 to prevent probabilties from exceeding 1. Suppose we have n classifiers and let $P(\omega_c|x)_{\mathrm{i}}$ be the probability of predicting class ω_c by classifier i. Then the averaging and weighted average rules work in the following manner [5]:

Averaging Rule:

$$P(\omega_c|x) = \frac{1}{n}\sum_{j=1}^{n} P(\omega_c|x)_{\mathrm{i}} \tag{1}$$

Weighted Averaging Rule:

$$P(\omega_c|x) = \frac{1}{n}\sum_{j=1}^{n} w_i P(\omega_c|x)_{\mathrm{i}} \tag{2}$$

$$\sum_{j=1}^{n} w_i = 1 \tag{3}$$

These probabilites can then be used to yield the final classification result by means of thresholding. Evolutionary algorithms can be used to determine the weights for each classifier in case of the weighted averaging method. [6] have used Genetic Algorithms to learn the stacking weights for combining various classifiers. Differential Evolution has also been used in the past to learn the weights for each classifier [7]. Besides statisitical models, neural networks have also been extensively used for classification. Neural Networks can be of various types and some of them include Feedforward Networks and Recurrent Networks. Besides these neural networks, there also exist Spiking Neural Networks (SNN) a special class of Artificial Neural Network (ANN) where the neurons communicate using spikes instead of real numbers. They are functionally similar to biological neurons and hence are powerful for analysing neural information processing, plasticity and learning [8]. The vital component of SNN are spiking neurons which communicate by generating and propogating electrical impulses called action potentials or spikes [9]. The dynamics of a spiking neuron is characterized by a state variable called membrane potential which is responsible for generating spikes if the total membrane potential exceeds a certain threshold. A variety of neuron models exist in literature like the LIF Neuron [10]or the Spike Response Model [11].

A. Spiking Neural Networks

Spiking Neural Networks consist of two principal units, a spiking neuron and synapses which connect neurons to each other. Spiking neurons communicate with each other using

Spike sequences/trains. A spike train in mathematical terms can be described as follows [8]:

$$S(t) = \sum_{f} \delta(t - t^f) \tag{4}$$

where, t^f is the firing time time of the f^{th} spike and $\delta(.)$ is a Dirac Delta function with $\delta(t) \neq 0$ for t = 0 and 0 otherwise.

Each input spike to the neuron increases the membrane potential of the neuron by some amount. Spikes from various synapses are integrated to increase the membrane potential of the spiking neuron. When the membrane potential of the neuron crosses a threshold called as the firing threshold it fires a spike. Immediately after a spike is fired by the neuron the membrane potential is reset to a value called as the resting potential. The neuron then enters a refractory period where it fires no spike [8]. In the spike response model, the neuron potential is summarized in terms of a response kernel [11]. The SRM is described by the following equations:

$$u_{\text{i}}(t) = \sum_{t_{\text{i}}^{\text{f}} \in F_i} \eta_i(t - t_{\text{i}}^{(\text{f})}) + \sum_{j \in \Gamma_i} \sum_{t_{\text{j}}^{(\text{f})} \in F_j} w_{\text{ij}} \epsilon_{\text{ij}}(t - t_{\text{j}}^{(\text{f})}) \tag{5}$$

where, $t_{\text{i}}^{(\text{f})}$ is the firing time of the last output spike and $t_{\text{j}}^{(\text{f})}$ is the firing time of the j^{th} pre-synaptic spike. η_i is a kernel function that describes spike emissions after membrane potential has crossed threshold. ϵ_{ij} is a kernel function that describes the response of the post synaptic neuron and w_{ij} is the strength of the synapse between the i^{th} pre-synaptic neuron and j^{th} post-synaptic neuron. An illustration of a simple spiking neuron is given below:

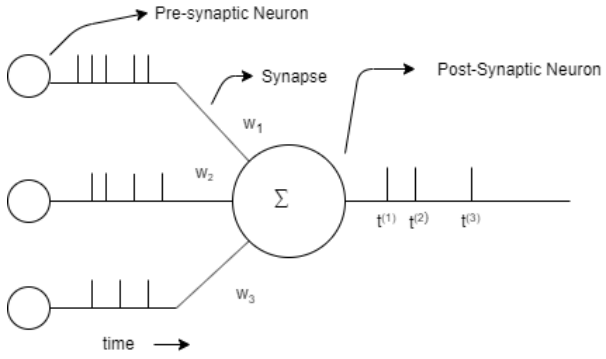

Fig. 1. Schematic Diagram of a simple 2-layer SNN

We can see in Fig.1 that the neuron has 3 pre-synaptic connections. The input spikes from these 3 neurons induce a potential and it keeps rising. As soon as the threshold is crossed from below the neuron fires a spike before coming to rest again.

An important aspect of spiking neural networks is information encoding [12]. Two main approaches of neural information coding are popular. Rate coding and Temporal Coding. Rate Coding approach assumes that information is encoded in the number of spikes or mean firing rate of a neuron. Temporal coding on the other hand assumes that information is encoded in the precise timing of the individual spikes.

To perform a classification task using Spiking Neural Networks the input has to be encoded in the form of spike trains. Various methods exist to encode real-valued inputs into spike trains. Two approaches of encoding are popular in literature. One of them is called rate coding where the real valued input is converted to a poisson spike train with a mean firing rate corresponding to the real valued input. The other approach is called as Population encoding. In population encoding a single real valued input value is represented by a population of neurons spiking at certain times [13]. The outputs are also encoded in a suitable manner. Output encoding maybe present in the form of number of spikes or precise timing of spikes. In the former case, the number of output spikes determine the class while in the latter case different spike times are associated with different classes.

B. Evolutionary Algorithms for Continuous domains

In order to learn the weights of a spiking neural network we need to employ evolutionary algorithms that can function with real valued data. [14] describe real coded genetic algorithms which can be used in cases where the search space is continuous. A modified ACO algorithm suited for optimization in continuous domains has been proposed in [15]. Another very popular evolutionary algorithm used in continuous domains is called Differential Evolution [16].

Real coded genetic algorithms follow a similar structure as the original binary coded genetic algorithms but the crossover and mutation strategies are modified accordingly. [14] give a good review of the various crossover and mutation strategies used in cases of RCGAs. Differential evolution modifies the existing population members by adding a weighted difference of two solutions to a third solution. If the newly obtained solution is better than the existing solution, the new one replaces the existing one and this process continues in an iterative manner [16]. The pseudo code for differential algorithm is given as follows:

Algorithm 1: Differential Evolution Algorithm

```
Result: Local or Global Optimal Solution
Initialize Population;
Compute Fitness;
while While stopping criteria not met: do
    for each member of the population
    Select 3 random solutions a,b,c leaving the current
      member;
    Compute mutant solution as follows: mutant = a +
      rand*(b-c);
    Replace some indexes of current member with
      indexes of mutant solution;
    Check if new solution is better than current
      solution;
    If YES replace existing solution else CONTINUE;
end
```

C. Structure of Report

The report is organized in the following manner. We first describe the two experiments conducted in this report. All the relevant details with respect to the experiments have also been documented in the methodology section of this report. As described above, ensemble learning is a popular technique aimed at improving the predictive power. Lack of supervised learning methods for spiking neural networks naturally promotes the use of evolutionary algorithms. Previous works have demonstrated the use of evolutionary algorithms for spiking neural networks. [12] have used PSO to train a Spiking Neural Network with dynamic synapses. [17] have used a novel method termed as Parallel Differential Evolution to train Spiking Neural Networks. We propose to use an ensemble of evolutionary algorithms with an aim of increasing the accuracy of Spiking Neural Networks for classification.

II. Methodology

A. Experiment Descriptions

We conduct two experiments by using the Iris Dataset. The Iris benchmark dataset consists of 150 samples each having 4 features. The dataset consists of 3 classes Setosa, Virginica and Versicolor out of which two classes are non-linearly separable. Each class consists of 50 instances.

The first experiment is a simple experiment where we use 7 different classifiers for the Iris Dataset. Our aim is to determine the optimal combination of weights that should be assigned to each classifier in order to maximize the prediction accuracy.

The second experiment is aimed at using an ensemble of evolutionary algorithms to train a Spiking Neural Network to classify the instances. Other relevant details about the experiment have been described in the sections below.

B. Experimental Setup - 1

We first do a train-test split of 50% before proceeding towards building the models. Seven base classifiers were trained using the training data set and 10-fold cross validation was carried out in order to tune the hyperparameters for each classifier.We would first like to mention that accuracy was used as the fitness function for all the evolutionary algorithms. In order to save computing time, we compute the predictions on the test data set after the classifiers are trained. These predictions are stored so that they can be used later directly without having to compute it with every iteration. Two different approaches were tried in order to improve the prediction accuracy using ensemble methods.

In the first approach, a binary coded genetic algorithm was used to determine the classifiers which would be part of the ensemble. Later, a majority voting of the predictions obtained from these classifiers was used to yield the final predictions. It was observed that this method could not outperform any of the classifiers that we had tried before.

In the second approach we take a weighted average of the probabilities obtained from each classifier and then classify the example according to the weighted combination of probabilities. Real-Coded Genetic Algorithms and Differential Evolution algorithms was used for this purpose. The results concerning with both the algorithms have been documented in the results section.

C. Experimental Setup - 2

In this experiment, we wish to evaluate the performance of Spiking Neural Networks by using an ensemble of evolutionary algorithms. We use RCGA, Differential Evolution and PSO as our base algorithms. The motivation is that various algorithms may converge to different optima which may lead to diverse classification results. Combining these results may result in a better accuracy as the errors of one algorithm would be compensated by the other. We use the Spike Response model as our neuron model and the SRM used in the experiment is described as follows [18]:

$$u(t) = \eta(t - t_i) \sum_{j|t_j > t_i} w_j \epsilon(t - t_j) \tag{6}$$

where, $u(t)$ is the membrane potential of the neuron at time t, $\eta(t - t_i)$ is the membrane reponse after a spike. $\epsilon(t)$ describes the excitatory post synaptic potential of the neuron. We integrate spikes from all synapses.

$$\eta(t-t_i) = K_1 \exp \frac{-(t-t_i)}{\tau_m} - K_2(\exp \frac{-(t-t_i)}{\tau_m} - \exp \frac{-(t-t_i)}{\tau_s}) \tag{7}$$

$$\epsilon(t - t_j) = K(\exp \frac{-(t-t_j)}{\tau_m} - \exp \frac{-(t-t_j)}{\tau_s}) \tag{8}$$

where $K = 2.1$, $K_1 = 2.0$, $K_2 = 4.0$, τ_m and τ_s are synaptic time constants with values 10.0 and 2.5 respectively.

We use a single layer spiking neural network for this classification task. The network topology used is 4-1. We follow the approach followed in [17]. The inputs are first normalized in the range [0,10]. These inputs are then transformed to uniformly distributed spikes in the simulation period corresponding to the input value. We use a simulation period of $T = 200ms$ with $dt = 1.0ms$. The threshold of the neuron is set to be $0.1mV$. Due to the heavy computation involved we only use a sample size of 30 random instances. No seperate test set is used.

The outputs are encoded in the following manner: If the number of spikes are in the range [0,10] the input is classified as class 1, if the spikes are in the range (10,20] then they are classified as class 2 else they are classified as class 3. We use three different evolutionary algorithms to train the Spiking Neural Network. By training we wish to learn the optimal weights which gives the highest accuracy. An illustration of the behaviour of the spiking neuron for a single instance is given in Fig.2 below :

Fig. 2. Output Spike Train of an Iris Dataset Training Instance

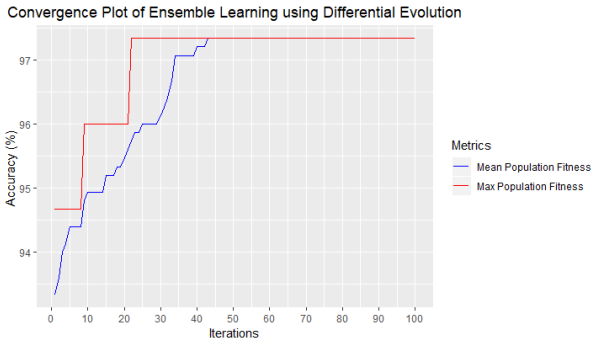

Fig. 3. Convergence Plot for Differential Evolution method

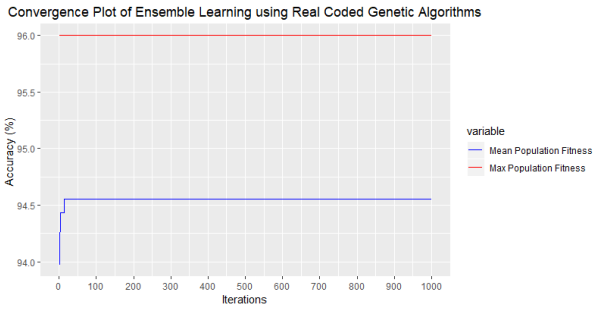

Fig. 4. Convergence Plot for RCGA method

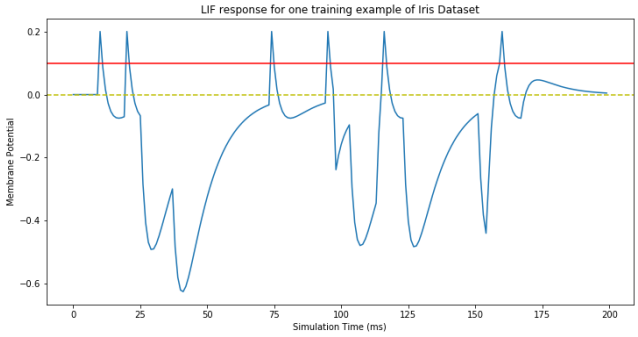

Fig. 5. Output spike train for a training instance before learning

III. Results and Discussions

In case of the first experiment, two evolutionary algorithms have been used to determine the optimal weights for an ensemble setting. We have been successfully able to achieve a test accuracy of 97.33% on the Iris Dataset by combining the predicted probabilities from various classifiers. Differential Evolution was able to converge to the optimal solution in less than 50 iterations and performed significantly better than Real Coded Genetic Algorithms. The convergence plots for both the algorithms have been illustrated in Fig.3 and Fig.4 respectively. The results have been presented below in a tabular form:

Classifiers	Accuracy
Logistic Regression	96%
KNN	94.67%
Naive Bayes	92%
Decision Trees	92%
Random Forests	92%
Gradient Boosting	92%
SVM	96%
Ensemble Classifier	**97.33%**

TABLE I
Experiment-1 Results

The convergence plots for both Differential Evolution and Real Coded Genetic Algorithms have been given below.

In the case of the second experiment, three evolutionary algorithms were used to train a Spiking Neural Network on the Iris dataset. The highest accuracy achieved in SNN is 76.67% using Differential Evolution. Before training the SNN it produced the output as illustrated in Fig. 5.

After training the weights using evolutionary algorithms we can see that the neuron is able to clearly distinguish the three classes based on the number of spikes emitted and this is illustrated in Fig.6, Fig.7 and Fig.8 respectively.

The results of training an SNN for a classification task using Evolutionary Algorithms have been summarized in Table 2.

The predictions from the three evolutionary algorithms were combined using majority voting to see if there was any improvement in the accuracy. There was no improvement in the accuracy and the reason is that all the evolutionary algorithms converged to nearly the same solution hence, classification results were not diverse. Due to the lack of diversity in the classification results combining the various agents did not

Evolutionary Algorithms	Best Accuracy
Real Coded Genetic Algorithm	70%
Differential Evolution	76.67%
Particle Swarm Optimizer	73.34%
Ensemble Classifier	**76.67%**

TABLE II
Experiment-2 Results

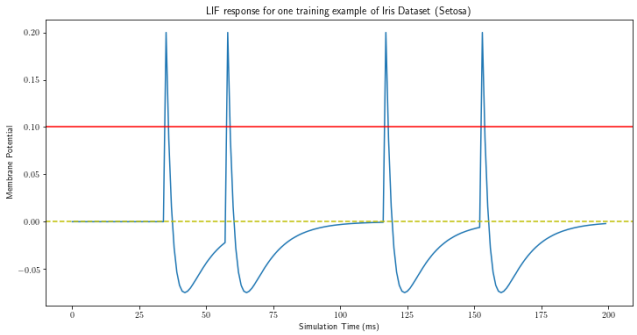

Fig. 6. Output spike train for a Class-1 instance after training

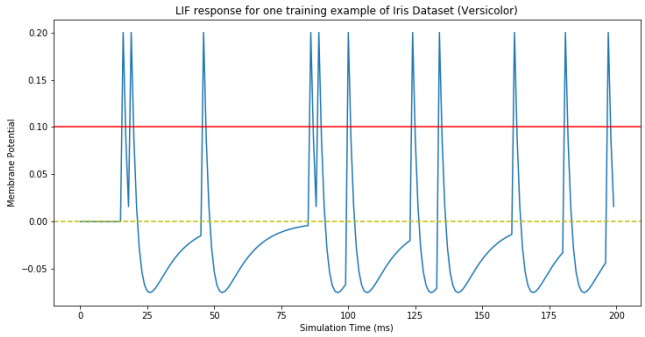

Fig. 7. Output spike train for a Class-2 instance after training

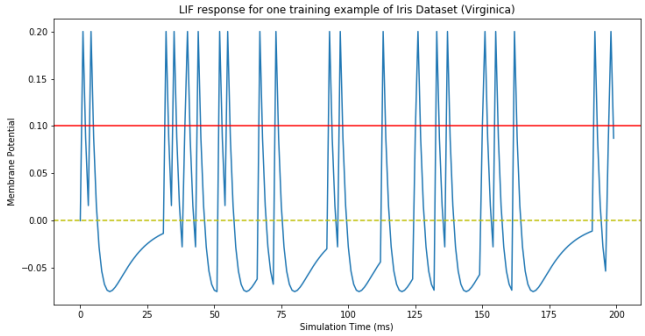

Fig. 8. Output spike train for a Class-3 instance after training

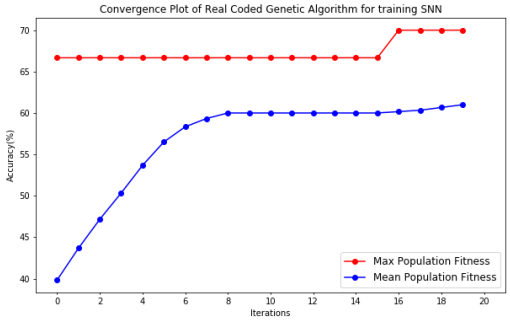

Fig. 9. Convergence Plot of RCGA for training SNN

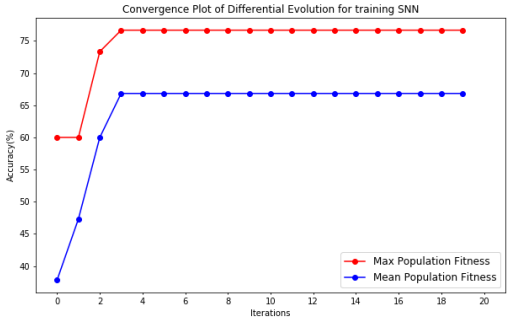

Fig. 10. Convergence Plot of Differential Evolution for training SNN

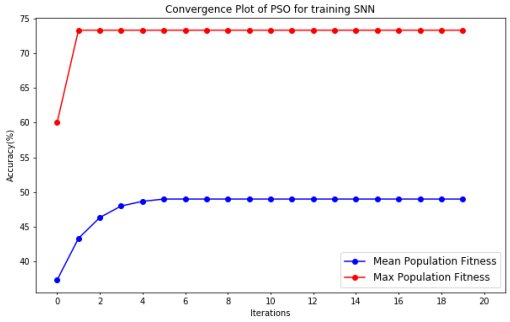

Fig. 11. Convergence Plot of PSO algorithm for training SNN

show any further improvement. The convergence plots of all the three evolutionary algorithm for training SNN have been shown in Fig.9, Fig. 10 and Fig. 11 respectively :

IV. Discussion and Conclusion

In this report a novel way of using an ensemble of evolutionary algorithms to train Spiking Neural Networks (SNN) has been proposed. The motivation behind such an idea was to increase the predictive power by including a diverse set of classifiers. The predictions obtained are not diverse enough to help ensemble learning benefit from it. It is also possible that the number of evolutionary algorithms used is less and including more algorithms as base classifiers may help alleviate the issue. Training all the three algorithms on the entire dataset and then combining them is a significant task that needs to be accomplished. Future approaches include improving the network topology to a more complex one and evaluating the possibility of combining the results from these evolutionary algorithms using another meta heurisitic as it

was done for the first experiment. All the relevant codes for conducting the experiments have been included along with this report.

V. Acknowledgment

I would like to take this opportunity to thank Dr. Mario Ventresca for approving the project and giving me the opportunity to pursue these experiments. It has been an immense learning experience to experiment with evolutionary algorithms for continuous domains in context of training SNNs.

References

[1] Dietterich, T. G. (2000a). Ensemble methods in machine learning. In Multiple classifier systems (pp. 1–15). Springer

[2] Efron, B., Tibshirani, R. (1993). An Introduction to the Bootstrap. Chapman and Hall, New York.

[3] R. E. Schapire, "The strength of weak learnability," Machine Learning, vol. 5, no. 2, pp. 197–227, June 1990

[4] D. H. Wolpert, "Stacked generalization," Neural Networks, vol. 5, no. 2, pp. 241–259, 1992

[5] Valentini G., Masulli F. (2002) Ensembles of Learning Machines. In: Marinaro M., Tagliaferri R. (eds) Neural Nets. WIRN 2002. Lecture Notes in Computer Science, vol 2486. Springer, Berlin, Heidelberg

[6] Sikora, Riyaz and O'la Al-Laymoun. "A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms." Handbook of Research on Organizational Transformations through Big Data Analytics. IGI Global, 2015. 43-53. Web. 9 Dec. 2019. doi:10.4018/978-1-4666-7272-7.ch004

[7] Yong Zhang, Hongrui Zhang, Jing Cai, and Binbin Yang, "A Weighted Voting Classifier Based on Differential Evolution," Abstract and Applied Analysis, vol. 2014, Article ID 376950, 6 pages, 2014. https://doi.org/10.1155/2014/376950.

[8] Ponulak, Filip Kasiński, Andrzej. (2011). Introduction to spiking neural networks: Information processing, learning and applications. Acta neurobiologiae experimentalis. 71. 409-33.

[9] Kandel ER, Schwartz TMJ, Jessel TM (1991) Principles of Neural Sciences. Elsevier, New York, NY.

[10] R B Stein. The frequency of nerve action potentials generated by applied currents. Proc. R. Soc. Lond. B, 167:64–86, 196

[11] Gerstner W. A framework for spiking neuron models - the spike response model. In F. Moss and S. Gielen (Eds.), Handbook of Biological Physics, Volume 4, Chapter 12, pp. 469–516. Amsterdam: Elsevier, 2001.

[12] A. Mohemmed, S. Schliebs, S. Matsuda, N. Kasabov, SPAN: spike pattern association neuron forlearning spatio-temporal sequences. Int. J. Neural Syst. 22(4), 116 (2012)

[13] Qiang Yu, Huajin Tang, Kay Chen Tan, and Haoyong Yu. A brain-inspired spiking neural network model with temporal encoding and learning. Neurocomputing, 138:3–13, 2014.

[14] Herrera, F., Lozano, M. Verdegay, J. Artificial Intelligence Review (1998) 12: 265. https://doi.org/10.1023/A:1006504901164

[15] Socha, K.; Dorigo, M. Ant colony optimization for continuous domains. Eur. J. Oper. Res. 2008, 185, 1155–1173.

[16] Storn, R. Price, K. Journal of Global Optimization (1997) 11: 341. https://doi.org/10.1023/A:1008202821328

[17] Pavlidis, Nicos Tasoulis, O.K. Plagianakos, V.P. Nikiforidis, George Vrahatis, Michael. (2005). Spiking neural network training using evolutionary algorithms. 2190 - 2194 vol. 4. 10.1109/IJCNN.2005.1556240.

[18] Masquelier, T., Guyonneau, R., and Thorpe, S. (2008). Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. PLoSONE 3:e1377. doi: 10.1371/journal.pone.0001377