Evolving Spiking Neural Networks for Predicting Transcription Factor Binding Sites

Heike Sichtig, Member, IEEE, J. David Schaffer and Alberto Riva

Abstract—Interdisciplinary problem solving requires a fundamental understanding of complex biological adaptive systems, from cellular to molecular level in order to tackle challenging problems such as neurodegenerative diseases. In this work we present a description and an initial evaluation of a Spiking Neural Network-Genetic Algorithm (SNN-GA) system we are developing for the computational prediction of transcription factor binding sites (TFBS). The SNN-GA approach is based on modeling information processing of biological neurons through evolutionary processes. The goal of our work is to reduce the number of false positives in the predicted TFBSs, through a more precise modeling of information contained in the alignments in the training data. We show an evaluation of four proposed network topologies that represent TFBS data. We use real TFBS data from the TRANSFAC® database and appropriately generated negative samples. We evaluated the network topologies one three well-known models for transcription factors: RSRFC4, ZID and p53. Benchmark performances for these models are given using MAPPER and MATCHTM. The results show that our method has the potential to attain very high classification accuracy.

I. INTRODUCTION

The identification of novel putative binding sites for a Transcription Factor (TF) of interest in a DNA sequence under analysis is one of the most difficult challenges in computational biology. Complexity arises with traditional approaches to computational modeling of the binding sites because only limited samples of actual sites are available and patterns are difficult to extract due to variability, possible interdependence and human error. Variability refers to changes of nucleotides at various positions in the samples of actual sites. Interdependence relationships of two or more nucleotides may arise in samples of actual sites, contributing to the challenge of finding an accurate predictor. The third factor, human error, refers to many complications that can arise when dealing with TFBSs, starting with experimental verification of an actual sample of a site to a misspelled data entry for a nucleotide. We report on the application of Evolving Spiking Neural Networks, a sophisticated machine-learning framework based on the combination of Neural Networks and Genetic Algorithms, to the computational prediction of Transcription Factor Binding Sites (TFBSs).

We used a SNN-GA platform that we developed to create models for representation and detection of binding sites for a TF of interest. Spiking Neural Networks (SNNs) are a variant of artificial neural networks. Spiking neurons are more biologically realistic than earlier models being based on precise spike timing [1], [2]. Hence, SNNs are able to represent complex spatial-temporal patterns in data and are advantageous models for complex dynamics governing many real world problems that require modeling of signals that vary over time. Optimization of SNNs is achieved through evolutionary computation that relies on Genetic Algorithms (GAs), a form of global search heuristic. Each TFBS model is a SNN that is trained using available examples of real binding sites and appropriately defined negative examples, and can be used to find new putative binding sites.

II. BACKGROUND

Common methods for the identification of TFBSs based on the probability distribution of the four nucleotides at each position [3] and the extent to which they are conserved by evolution [4] typically suffer from poor specificity resulting in an excessive number of false positives. Recent work has demonstrated the importance of taking dependencies among different positions into account when modeling a TFBS, instead of treating all positions as independent [3], [5], [6], [7].

Genetic Algorithms have been previously applied to the prediction of transcription factor binding sites [8], [9], [10]. The SiteGA web-based tool was used by Levitsky et al. [8] to discover dinucleotide interactions that are most significant to distinguish real sites from false positives. According to the authors, SiteGA performs close to, and sometimes better than, probability weight matrices (PWMs). The web-based tool supports only a limited number of TFs. Another algorithm described by Hawkins et al. [11] exploits evolutionary information using a phylogenetic tree and a model of evolution.

III. METHODS

Our approach to the detection of TFBS is based on the hypothesis that binding sites are characterized by complex patterns of nucleotides that are not limited to the frequencies of nucleotides at each location and to simple correlations, but may include complex relationships among nucleotides at different positions. We are therefore interested in testing whether an adaptive modeling framework that is able to learn
these patterns from the data may be successfully applied to the TFBS detection problem.

Our work relies on an extension of the SNN-GA approach introduced by Sichtig et al. [12], [13], [14]. The SNN-GA framework was developed to perform realistic simulations of natural and artificial neuronal networks, but its ability to accurately represent and simulate complex systems makes it useful for a wider range of applications.

The current system, consists of a Spiking Neural Network (SNN) simulator and a Genetic Algorithm (GA). The simulator is an event-driven form of Sichtig’s Spiking Neural Network Simulator (SSNNS), previously developed by Sichtig [12], while the GA is a variant of the CHC family of genetic algorithms [15]. CHC is an acronym for cross-generational elitist selection, heterogeneous recombination and cataclysmic mutation. After the SNN is initialized, the GA heuristically adjusts some of the network parameters until a termination criterion is reached. The network parameters that are adjusted by the GA are called evolvable parameters. The user generally sets the criterion to be either a maximum generation or a specific target fitness.

A SNN consists of at least one spiking neuron, inputs (receiving input spikes) and synapses (connecting inputs to neurons or neurons to other neurons). When spikes arrive at a synapse, they are converted to post-synaptic potentials (PSP) which are then summed at the downstream neuron. Synapse properties include weights, delays and time constants, and neuron properties include time constants, and refractory period parameters.

A. Spiking Neural Networks (SNNs)

As mentioned before, this work is an extension of previous work on SNNs by [12], [13], [14] with Gerstner’s Spike Response Model (SRM) for leaky integrate and fire neurons. Artificial leaky neurons are biological analogs containing a decay term that allows the membrane potential to leak out any change in potential in case the input does not result in threshold crossing, i.e., firing a spike. Biological membranes are not perfect insulators, as previously thought, as ions diffuse through the membrane if no cell equilibrium is reached over time. It has been shown experimentally that leaky integrate and fire neurons are capable of mimicking the behavior of biological neurons [1], [2], [16].

The SNNs used in this work are composed of such leaky neurons connected by synapses. The state of each neuron is represented by its membrane potential, which changes over time in response to spikes arriving at its input synapses, and by its refractoriness following its own previous spikes. Equation (1) introduces the mathematical model underlying spiking neurons in our system.

\[ u_j(t) = \sum_{t_j^{(j)} \in F_j} \eta(t-t_j^{(j)}f_j) + \sum_{i} \sum_{t_i^{(s)} \in F_i} w_{ji} \times \epsilon(t-t_i^{(s)}d_{ji}) \]  

(1)

The total membrane potential of neuron j, \( u_j[t] \), is the sum of all the post-synaptic potentials (PSPs) caused by pre-synaptic firings of neurons i and the refractory effect of a negative reset potential. The total membrane potential of neuron j is composed of the \( \eta \)-function, modeling its own refractoriness, and the \( \epsilon \)-function, describing the effect of the pre-synaptic spikes of neurons i, where the weights (\( w_{ji} \)) and delays (\( d_{ji} \)) describe the dynamics of the synaptic connections. Once neuron j generates a spike at time \( t_j \) the membrane potential is reset by the spike after-potential, described by \( \eta \). Let \( \Gamma_j \) be the set of all neurons that make synaptic contact with the dendritic tree of neuron j. Then, \( F_j \) represents the spike train from pre-synaptic neuron i and \( F_j \) the spike train from the current neuron j.

For every neuron, the refractoriness \( \eta \) is calculated as a sudden drop followed by an exponential recovery, as shown in equation (2).

\[ \eta = (\eta_0 - U_R) \times \exp{-(\Delta t) \over \tau_r} \]  

(2)

\( \eta_0 \) is the negative after-spike potential, \( U_R \) is the resting potential (usually taken to be zero) and \( \tau_r \) is the refractory time constant. \( \eta_0 \) is set to -70mV and \( U_R \) is set to 0ms in subsequent simulations. \( \tau_r \) is an evolvable parameter.

The spike response function for the computation of PSPs using the SRM model is shown in equation (3).

\[ \epsilon = \frac{1}{1 - \frac{\tau_s}{\tau_m}} \times (\exp{-(\Delta t) \over \tau_m} - \exp{-(\Delta t) \over \tau_s}) \]  

(3)

This function is an exponential growth and decay function with two parameters specifying rise time (\( \tau_s \)) and length of PSPs adjustment (\( \tau_m \)). \( \tau_s \) and \( \tau_m \) are evolvable parameters. The time difference \( \Delta t \) is computed using the spike times of pre-synaptic neurons up to the current computation time. The double-exponential spike response function \( \epsilon \) allows flexibility to account for various forms of spike responses, i.e. short/long rise time and short/long duration. When the neuron’s cumulative membrane potential reaches a preset static threshold and the neuron is outside its refractory period, the neuron fires a spike and resets to its highest refractory value: \( \eta_0 \).

B. Genetic Algorithms (GAs)

SNN modeling is still an art with a wide range of parameters that need to be attuned to the problem at hand. Heuristic search algorithms can aid in the understanding of complex problem solving when the solution space is large. One such method is a GA that can discover good solutions even in the unknown territory of fitness landscapes. Our SNN-GA framework uses a GA to tune the SRM network parameters mentioned in the previous section.

The GA used in this work is a variation of the CHC algorithm developed by Eshelman [15], entirely rewritten by the authors. CHC uses survival competition between parents and offspring and vigorous crossover, but only when two individuals are sufficiently distant (incest prevention that virtually eliminates genetic drift [17]). CHC also provides soft restarts to mitigate against convergence to a local maximum: if the organisms in the population become too similar to
each other, the organism with the highest fitness is retained, while the rest of the population is replaced by new organisms obtained by mutating the survivor’s chromosome. It was shown that CHC can outperform the traditional GA as a function optimizer over a wide range of fitness functions [15].

C. SNN Parameter (Chromosome) Encoding

The simulator is controlled by a set of parameters that govern the activities of each of its elements (e.g. weights, delays, refractory periods and time constants). These control parameters are tuned by the CHC algorithm, described in the previous section, which enables the system to learn a desired emergent pattern, using a fitness function that depends on the closeness between the generated spike trains and given target spike trains. We use a reflected Gray code to encode the set of parameter values as a chromosome. The prominent feature of Gray code is that two successive values of this binary numeral system differ in only one bit, e.g. 00⇒01⇒11⇒10. These Hamming distance properties are often used in GAs.

D. General Fitness Measure

The definition of the fitness function is a critical aspect in the use of GAs: organisms are ranked and selected on the basis of their fitness, and the purpose of GAs is to efficiently discover an organism with the best possible fitness.

The next section describes our application of the SNN-GA framework to a well known computational biology problem and defines the fitness function as the model’s ability to discriminate between positive and negative examples in the training set.

IV. APPLICATION: PREDICTING TFBS

This section describes how we applied the SSN-GA framework to the prediction of TFBSs. The basic steps of our method are the following:

1) We start with a set of known binding sites for a TF of interest (obtained from experiments, or from a database like TRANSFAC® [18]) and an equal number of negative examples (see below for a description of how the negative examples were generated). The sequences are converted into spike trains according to an appropriate encoding.

2) We create a population of networks with the desired topology, with randomly initialized parameters.

3) Each network is fed a spike train corresponding to a TFBS sequence and one corresponding to a non-TFBS sequence. The resulting output spikes are compared by an appropriate fitness function that measures the network’s ability to distinguish real TFBS sequences from non-TFBS ones. This is repeated for each pair of TFBS / Non-TFBS sequences.

4) The GA is used to evolve a set of network parameters that maximizes the separation between the positive and the negative TFBS examples.

A. TFBS: Generation of Negative Samples

In order to effectively train a classification model, one would need a set of negative examples (sequences that are known not to be binding sites) in addition to the positive ones. Unfortunately, good negative examples are not easily available. The number of possible sequences of the same length as a typical binding site is so large that screening them experimentally is unfeasible. On the other hand, generating a set of completely random sequences without knowledge of known TFBS sites and treating them as negative examples, would not provide sufficient discriminating power. One of the caveats with generating a negative set using this method is the potential to include actual TFBS.

Instead, we could use the characteristics of known TFBS sequences, such as their probability weight matrix (PWM), to generate new sequences that are similar to the known sites, but at the same time different enough to be suitable as negative examples for training. A PWM is a matrix that describes the frequency distribution of the four nucleotides at each position in the TFBS sequence. In our procedure, we first compute the average pair-wise similarity of the real TFBS sequences, and their PWM. We then generate a new sequence by sampling from the PWM distributions, and we compute the average similarity between this sequence and the true TFBSs. If the average similarity is higher than the average pair-wise similarity, the new sequence is considered a negative example.

The rationale of this method is to produce sequences that are as close as possible to the real TFBSs (being generated by the same nucleotide distribution at each position) while being sufficiently different from them to assume that they would not function as binding sites in reality. This procedure can be repeated to generate the desired number of negative examples.

B. TFBS: Sample Transcription Factors to be Tested

RSRFC4 is an isoform of the Human myocyte enhancer factor 2A (MEF2A), a member of the MEF2 family of DNA-binding regulatory proteins. The database we used contains 38 known binding sites for this factor and the sequences have a length of 16 nucleotides. The consensus sequence is: attcTAtttTAgtct (uppercase letters indicate highly conserved positions). Figure 1(a) is a graphical representation of the frequencies with low-quality positions trimmed at both ends. The relative entropy (y-axis) of the sequence logo is a measure of the amount of information that a nucleotide at a specific position (x-axis) carries. If a position is highly conserved (an uppercase letter in the consensus sequence), the information content at that position is low, meaning that the same nucleotide appears most of the time at that position. If it is less conserved, the information content will be higher.

ZID is an isoform of the Human Cys2Hys2 zinc finger domain class. The database we used contains 36 known binding sites and the sequences have a length of 12 nucleotides. The consensus sequence is: ggCTCtacacc. The sequence logo is shown in Figure 1(b).
p53 is an isoform of the tumor suppressor protein, important in multicellular organisms and encoded by the TP53 gene in human. p53 is involved in cell cycle regulation and functions as a tumor suppressor and hence is involved in the prevention of cancer. The database we used contains 17 known binding sites and the sequences have a length of 20 nucleotides. The consensus sequence is: ggaCATGcCGgCATGtcc. The sequence logo is shown in Figure 1(c).

C. TFBS: Network Structures

One of the main benefits of our method is that it allows us to experiment with different alternative network topologies, reflecting different hypotheses about the internal structure of binding sites. While traditional TFBS detection methods treat each base position as independent of all the others, there is increasing evidence that this assumption is, in general, not verified. The networks used in our approach have the ability to represent interactions and dependencies among nucleotides at different positions in the alignment (even non-contiguous ones). The specific structure, that we can assume will be different for each TF, is learned on the basis of the available binding site data in each case. We defined two different types of model structures: nucleotide-based and position-based, and present two network topologies for each, resulting in four networks.

The nucleotide-based networks have five inputs corresponding to the A, T, C, G and N nucleotides, and the order in which they spike is determined by the nucleotide occurring at each consecutive position in the TFBS sequence.

The position-based networks have one input for each nucleotide in the alignment; each input spikes only once at a time depending on the specific nucleotide it contains (in our case: A, T, C, G and N correspond to arbitrarily selected spike times of 10ms, 20ms, 30ms, 40ms and 50ms respectively). Following are graphical representations of the four selected network topologies and their descriptions.

The Nucleotide-Based-Simple network (NBS) in Figure 2 consists of five inputs (corresponding to A, T, C, G and N) and one output neuron (OUT). The inputs are directly connected to the output neuron with synapses (small circles). Input spikes are encoded based on the TFBS sequence and spaced 10ms apart. A sample sequence (selected from the ZID TF) is depicted in Figure 2 along with the structure of NBS-type networks. We set the target spike for a TFBS sequence to 40ms after the last input. For ZID sequences the length is 12 nucleotides, so the last spike will occur at 120ms and the target spike will therefore be at 160ms. The fitness for this network is based on the temporal distance between the spike generated by TFBS sequences and by NON-TFBS sequences. The further away the NON-TFBS sequence fires from the TFBS target, the higher the fitness. If the NON-TFBS sequence does not produce a spike at all after the target, it is considered a valid classification and rewarded with a spike difference of 10ms. In this way, the absence of an output spike is not rewarded too much to drive evolution to a spike/no-spike network, but at the same time it allows for such cases when they take place. In case the output neuron fires multiple times, only the first spike after the target is considered.

The Nucleotide-Based-Complex network (NBC) in Figure 3 shows the structure of NBC-type networks (similar to NBS-type networks). The inputs for NBC-type networks are connected to additional neurons (HA, HT, HC, HG, HN), also called hidden neurons. Hence, each nucleotide has its own hidden neuron; the hidden neurons are fully connected to allow interaction, and are then connected to a single output neuron (OUT). We use the same fitness function as introduced previously for the NBS-type networks. Hidden neurons are allowed to spike arbitrarily. Figure 3 shows that each hidden neuron in NBC-type networks receives
input from its corresponding nucleotide, is connected to all other hidden neurons and the output neuron, resulting in 30 connections or synapses (small circles) for this network. In contrast, NBS-type networks have only 5 connections from the inputs to the output neuron. The network structure for NBC-type networks is much more complex and neurons are expected to fire at a greater rate due to connectedness of neurons.

The Position-Based-1-Neuron network (PB1N) in Figure 4 has one input for each position in a TFBS sequence (12 for TF ZID) and a single output neuron (OUT). The nucleotides found at each position (A, T, C, G or N) are converted into spike times (10ms, 20ms, 30ms, 40ms and 50ms respectively) for the corresponding input. The fitness for PB1N-type networks is computed by setting the target of firing a spike close to a specified firing time if it is a TFBS sequence length of the model.

The Position-Based-2-Neuron network (PB2N) in Figure 5 shows the structure of PB2N-type networks with its inputs corresponding to each TFBS sequence position (12 for TF ZID) and two output neurons (OUT1, OUT2). As in the previous case, the base at each position (A, T, C, G or N) is converted into a spike time (10ms, 20ms, 30ms, 40ms or 50ms respectively). For the fitness function of PB2N-type networks, we require one of the two output neurons to spike as close as possible to a set target time of 100ms and the other one to spike as far away as possible from that time (or not spike at all) when the sequence is a real TFBS. For a NON-TFBS sequence the opposite should happen, with the second neuron firing close to 100ms and the first neuron spiking at a later time or not at all. In this way, simply checking which neuron fires a spike shortly after 100ms can easily identify TFBS and NON-TFBS.

D. Methods

As described in the previous section, the fitness of a network model in the SNN-GA framework is computed based on the desired target of firing a spike close to a specified firing time if it is a TFBS.

- We select a population size of 50 and a maximum generation criterion of 2000 for the genetic algorithm.
- The nucleotide-based networks have a target spike time of 200ms for RSRFC4, 160ms for ZID and 240ms for p53, chosen because of their last input spike at 160ms, 120ms and 200ms respectively. The target firing time for nucleotide-based networks changes based on the sequence length of the model.
- The position-based networks have a target spike time of 100ms, because their last spike time is always set at 50ms (inputs are seen simultaneously, not depending on sequence length).

Each SRM model parameter is encoded as a binary number consisting of the appropriate number of bits, using Gray coding; the chromosome is the bit vector obtained by concatenating the binary representations of all parameters.
Table I gives an overview of the encoding characteristics of tunable network parameters for this application. A relatively simple SNN network with a single spiking neuron and two synapses will require a chromosome of bit length 56 with the parameter encoding given in Table I. Hence, the resulting search space has already a complexity of $2^{56}$ and requires a heuristic algorithm to search for solutions rather than performing an exhaustive search.

The following tables provide an overview of the characteristics for the four networks to be tested for each binding site, Table II for the RSRFC4 binding site, Table III for the ZID binding site and Table IV for the p53 binding site. The search space for nucleotide-based networks is static due to their characteristic of processing input for each nucleotide separately. This makes predictions for computation time more feasible. The position-based networks are more complex and dynamic, for example p53 (with sequence length of 20 nucleotides) and the PB2N network with the largest search space of size $2^{224}$.

### Table I

<table>
<thead>
<tr>
<th>PARAM</th>
<th>NBS</th>
<th>NBC</th>
<th>PB1N</th>
<th>PB2N</th>
</tr>
</thead>
<tbody>
<tr>
<td>neuron $\tau_R$</td>
<td>11</td>
<td>200</td>
<td>0 – 10</td>
<td></td>
</tr>
<tr>
<td>neuron refrac</td>
<td>11</td>
<td>100</td>
<td>0 – 120</td>
<td></td>
</tr>
<tr>
<td>synapse weight</td>
<td>10</td>
<td>5</td>
<td>0 – 200</td>
<td></td>
</tr>
<tr>
<td>synapse delay</td>
<td>7</td>
<td>1</td>
<td>0 – 128</td>
<td></td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>CHARACTERISTICS OF NETWORKS FOR RSRFC4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTERISTICS</td>
</tr>
<tr>
<td>neurons</td>
</tr>
<tr>
<td>synapses</td>
</tr>
<tr>
<td>parameters</td>
</tr>
<tr>
<td>chrom length</td>
</tr>
<tr>
<td>search space</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>CHARACTERISTICS OF NETWORKS FOR ZID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTERISTICS</td>
</tr>
<tr>
<td>neurons</td>
</tr>
<tr>
<td>synapses</td>
</tr>
<tr>
<td>parameters</td>
</tr>
<tr>
<td>chrom length</td>
</tr>
<tr>
<td>search space</td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>CHARACTERISTICS OF NETWORKS FOR p53</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTERISTICS</td>
</tr>
<tr>
<td>neurons</td>
</tr>
<tr>
<td>synapses</td>
</tr>
<tr>
<td>parameters</td>
</tr>
<tr>
<td>chrom length</td>
</tr>
<tr>
<td>search space</td>
</tr>
</tbody>
</table>

### E. Results and Discussion

First, an overview of the four network simulation runs (for each TF) is given in Table V. The results are the percentages of correctly classified TFBS sequences versus NON-TFBS sequences from the training data set. The total number of sequences for each model is also provided (number in parentheses immediately following the TF name).

<table>
<thead>
<tr>
<th>TF</th>
<th>NBS (%)</th>
<th>NBC (%)</th>
<th>PB1N (%)</th>
<th>PB2N (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSRFC4(38)</td>
<td>38(92%)</td>
<td>38(100%)</td>
<td>37(97%)</td>
<td>38(100%)</td>
</tr>
<tr>
<td>ZID(36)</td>
<td>27(75%)</td>
<td>31(86%)</td>
<td>33(92%)</td>
<td>36(100%)</td>
</tr>
<tr>
<td>p53(17)</td>
<td>11(65%)</td>
<td>17(100%)</td>
<td>17(100%)</td>
<td>17(100%)</td>
</tr>
</tbody>
</table>

For nucleotide-based networks, the NBS-type network and the NBC-type network show similar classification results for both TFs. However, the average run time for NBC-type networks was 3 days on a Mac OS X 2.93 GHz Intel Core 2 Duo processor, compared to NBC-type networks that take on average for 2 hours only. The NBC-type networks have significant higher computational cost due to their complex network structure and display little benefit over NBC-type networks. It would be favorable to choose the NBS network for further testing.

The training data results suggest that position-based networks generate better classification results than nucleotide-based networks. The average run time for the PB1N-type networks was 3 hours, whereas the PB2N-type networks ran for about 9 hours on average. The evolved PB1N networks demonstrate high classification accuracies for all three binding site models and run within an acceptable time range.

A sample of evolved parameter values for a PB2N network for the ZID TF is given in Table VI.

Looking at Table VI, the evolved synapse parameter for the delay of the first synapse (1OUT) is 10ms. This synapse will most likely loose its influence on the membrane potential at 100ms. The next synapse (2OUT) has a delay of 127ms and results in a synaptic influence to neuron OUT after 127ms. This suggests that the first and second positions in the sequence (corresponding to 1OUT and 2OUT) are not as important as other positions given the target of output neuron OUT to spike as close to 100ms as possible. We know that positions 3-5 are important for the ZID TF because the
The evolved weight for synapses 3, 4 and 5 are very high (146.4, 170 and 140.6 in magnitude) with evolved delays of 60ms, 72ms and 1ms respectively (i.e. a C at synapse 3 and a T at synapse 4 will influence the membrane potential of neuron OUT and is most likely to contribute to a spike at 100ms). Further analysis and experiments are needed to detect a specific underlying pattern in the sequences.

The network evolution and performance is further examined for the PB2N network by looking at the fitness curve for the ZID and p53 TFs. Figures 6(a) ZID TF and (b) p53 TF show the progression of best (red), average (blue) and worst (black) fitness of an organism over 2000 generations in a simulation run. Each generation generates a SNN with a different set of evolved parameters (see table VI for a sample PB2N evolved network for ZID TF). The fitness for the evolution of the PB2N network for ZID includes three restarts due to divergence, suggesting that organisms (chromosomes) in the GA become too similar, but the organisms attain a higher fitness over time for the best PB2N network. Note that the best PB2N network is always kept (red curve). The best PB2N network for the p53 network is found at about 1000 generations and does not show divergence or a better fitness over the next 1000 generations.

The following two tables are measures of sensitivity of the four networks that were tested with SNN-GA and are compared to MAPPER and MATCH, two well-known methods to predict TFBS. Table VII shows the counts of true positives (true binding sites). Table VIII shows the count of false positives (negative binding sites that are classified as true binding sites).

### Table VII: Comparison with MAPPER and MATCH (True Positives)

<table>
<thead>
<tr>
<th>TF (#s)</th>
<th>NBS</th>
<th>NBC</th>
<th>PB1</th>
<th>PB2</th>
<th>MAP</th>
<th>MA*</th>
<th>MA**</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSR (38)</td>
<td>35</td>
<td>.98</td>
<td>37</td>
<td>38</td>
<td>.85</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>ZID (36)</td>
<td>27</td>
<td>.31</td>
<td>33</td>
<td>.36</td>
<td>.36</td>
<td>26</td>
<td>8</td>
</tr>
<tr>
<td>p53 (17)</td>
<td>11</td>
<td>.17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

MA* MATCH with minsum filter
MA** MATCH with minFP filter

### Table VIII: Comparison with MAPPER and MATCH (False Positives)

<table>
<thead>
<tr>
<th>TF (#s)</th>
<th>NBS</th>
<th>NBC</th>
<th>PB1</th>
<th>PB2</th>
<th>MAP</th>
<th>MA*</th>
<th>MA**</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSR (38)</td>
<td>11</td>
<td>.19</td>
<td>4</td>
<td>4</td>
<td>.20</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ZID (36)</td>
<td>8</td>
<td>.23</td>
<td>4</td>
<td>2</td>
<td>15</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>p53 (17)</td>
<td>3</td>
<td>.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

MA* MATCH with minsum filter
MA** MATCH with minFP filter

Figure 7(a) - ZID and (b) - p53 show the ROC curves with performances for SNN-GA (black), MAPPER (red), MATCH - minsum (blue) and MATCH-minFP (green). The SNN-GA PB2N network emerges as a predictor with 100% accuracy for the ZID and p53 TFs.

The comparison shows that the SNN-GA PB2N network is a favorable network for the prediction of binding sites for the tested factors. The SNN-GA PB2N network attains high classification accuracy with a very low false positive rate. The results are a first exploration with three TFs and more tests are needed to make a definite decision on network structure. The TFs were selected based on sequence length of the binding site to make training more reliable. Further testing will involve TFs with shorter binding sites to make a more realistic prediction in the future.

Lastly, we generated a random sequence of nucleotides and embedded samples of actual binding sites of the p53 TF at every 100 nucleotide for further testing of accuracy of the SNN-GA PB2N network. Sensitivity was again 100% accurate with all true binding sites detected and a low false positive rate of 6.4%. Hence, we show that the network is able to detect all known true binding sites within a randomly generated sequence with a minimal detection of false positives. It must be noted that the false positives detected are at least 50% different than the true binding sites and that they may be true binding sites that are not experimentally verified yet.

### V. Conclusion

Classification of TFBS and NON-TFBS sequences is a very difficult and an inherently complex task. The generation of negative samples from a limited set of actual binding sites proves to be a challenging task. Our paradigm is able to detect an underlying pattern in the TFBS model with consistent accuracies of 100 percent for the three TFs tested.
with the PB2N network. It should be noted that the results obtained may exhibit over-fitting due to training and testing with the same limited set of samples from the TFs. Cross validation and testing on more TFs will be necessary to support our claim of having developed an accurate predictor for TFBS. Moreover, the SNN-GA approach aided network structure selection and network parameter evolution.

Classification accuracies also depend on human errors in TFBS sequences and the generated set of negative examples. Manual inspection of the TFBS sequences and removal of suspicious sequences may help.

The major limitation of our method is currently its high computational cost: training a single network on a relatively few binding site examples can take several hours on a medium-sized workstation. Future work on improving speed of algorithms and parallelizing the GA will lower the computational cost. Once the network parameters that provide optimal classification ability have been identified, they can be hard-coded into a specialized network that can then be used to scan new sequences very efficiently.

Other directions for future research include generating negative examples with different methods, defining and evaluating more alternative network topologies, increasing the number of network parameters tuned by the GA, and exploring long- and short-term learning in synapses. Finally, we will work towards making our system for the identification of TFBS publicly available through a user-friendly interface.

REFERENCES