Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

Article

Flexibility of Boolean Network Reservoir Computers in Approximating Arbitrary Recursive and Non-recursive Binary Filters

Moriah Echlin[1,2], Boris Aguilar[1], Max Notarangelo[1], David L Gibbs[1], Ilya Shmulevich[1,*]

1 Institute for Systems Biology; 401 Terry Ave N, Seattle, WA 98109, USA
2 Molecular & Cellular Biology Program, University of Washington; Seattle, WA 98195, USA
* Correspondence: ilya.shmulevich@systemsbiology.org

Keywords: complex adaptive systems; systems dynamics; dynamical systems; signal processing; reservoir computing; machine learning; Boolean networks; biological modeling

Abstract

Reservoir computers (RCs) are a biology inspired computational framework for signal processing typically implemented using recurrent neural networks. Recent work has shown that Boolean networks (BN) can also be used as reservoirs. We analyze the performance of BN RCs, measuring their flexibility and identifying factors that determine effective approximation of Boolean functions that are applied in a sliding-window fashion over a binary signal, either non-recursively or recursively. We train and test BN RCs of different sizes, signal connectivity, and in-degree to approximate 3-bit, 5-bit and 3-bit recursive binary functions. We analyze how BN RC parameters and function average sensitivity, a measure of function smoothness, affect approximation accuracy as well as the spread of accuracies for a single reservoir. We found that approximation accuracy and reservoir flexibility are highly dependent on RC parameters. Overall, our results indicate that not all reservoirs are equally flexible and RC instantiation and training can be more efficient if this is taken into account. The optimum range of RC parameters opens up an angle of exploration for understanding how biological systems might be tuned to balance system restraints with processing capacity.

Introduction

Fruit flies use their olfactory system to differentiate between odors and in doing so, they map similar odors to similar neural responses. The odor molecules are essentially an input data stream. To process this input, the molecules are captured by odorant receptor neurons that feed a set of 50 projection neurons. The projection neurons, operating in 50-dimensional space, are expanded further into a higher dimension of 2000, stored by so-called Kenyon cells, and followed by the classification (locality sensitive hashing) that takes place in that high dimension [1].

Many biological systems are regarded as non-linear dynamical systems [2] operating in high-dimensional space. For example, proteins, genes, and macromolecules interact in a variety of ways to create the dynamics of cells [3]. Collections of cells interact and coordinate activity, forming cohesive units such as bacterial colonies, simple multicellular organisms, or tissues in more complex multicellular organisms. At each level of organization, 'input' signals (e.g., odors or hormones) are introduced into the system, processed by means of the system's dynamics,

and responded to accordingly, sometimes generating new 'output' signals as a byproduct [4]. Fundamentally, biological systems must process external signals in real time to inform a wide variety of response decisions.

In this vein, reservoir computing (RC) is a form of signal processing used for classification and online learning that uses such high-dimensional systems to process signals. Initially called echo state networks [2] or liquid state machines [5], the computational structure resembles an unorganized recurrent neural network. These networks were inspired by the most classical example of signal processing in biological systems — the brain. While initial research in neural networks was focused on modeling and understanding neural computation [6] applications and directions for research quickly expanded in scope. For reservoir computers, this includes systems for predicting time series from chaotic systems [7,8] as well as multiple real-world applications such as gas detection [9], robotics [10–12], and economic trends [13], Applications in processing human data are also found with human emotion [14] and gesture recognition [15], speech and text processing [16,17], and health care monitoring [18].

In biology, reservoir computers have been used to analyze biological data for identifying anomalous states such as cardiac arrhythmia [19], seizures [20,21], and microsleeps [22] as well as for high-dimensional classification such as cell type detection [23]. In these applications, RCs are used to interpret or classify biological data. RCs have also been used as models of biological systems, serving to explore the mechanisms and dynamics of said systems. For example, the questions of how the brain functions [24-26] and how gene regulatory networks respond to external stimuli [27,28] have been studied in an RC context.

In reservoir computing, an incoming signal is fed into the reservoir, a randomly connected recurrent network where nodes are connected via a variety of coupling functions. Thus, the reservoir transforms the signal into a high-dimensional representation. Finally, an output layer is trained, often with straightforward techniques such as regularized regression [29], to perform a classification or regression task (Fig. 1).

RCs can also be trained to act as signal processors, filtering the input to produce a new output. This can be through approximating a function that is normally applied to a window sliding over the signal. The type of function can be extended to recursive functions, where some elements of the input-window are replaced by values from previously generated outputs. These types of filters have a long history of use which includes filtering biological signals [30–32].

While reservoirs are most typically built using unorganized recurrent neural networks, it has been shown that any non-linear dynamical system that exhibits a fading memory property can theoretically be used to compute a time-invariant function on a signal [33]. In fact, DNA, carbon nanotubes, memristor networks, and even buckets of water have been shown to work as reservoirs [34–36]. One type of dynamical system that has only recently been explored as a reservoir are Boolean networks [37]. Boolean networks (BNs) are well-suited to reservoir computing since they are one of the simplest modeling approaches that 1) can capture the heterogeneity, both in wiring and coupling functions, that characterizes self-assembled systems, and 2) can exhibit non-trivial dynamical behavior required for computation [38]. Additionally, the fading memory property can easily be achieved in Boolean networks by adjusting two key parameters, the average in-degree of the Boolean functions, $\bar{K}$, and the bias, p, which is the probability that a function produces an output of 1 for any given set of inputs [39,40].

Implementation of Boolean networks, both in software and in hardware, is also considerably easier and faster than more traditional approaches [41].

Similar to neural networks, Boolean networks were first used as models of biological systems, namely gene regulatory networks [37,42,43]. While often researched to understand cellular behavior in terms of protein/gene interaction networks in isolation [44–46], BNs have also been used to understand how external signals impact biological systems [47–49]. These systems rely on their ability to process external signals (relevant to survival or population function) and make decisions based on them in real time, thereby performing a similar computation that a reservoir does [4]. Just as the Boolean network framework has provided insight into how biological systems function, studying BN reservoir computers (BN RCs) may further our understanding of biological signal-response.

Previous work on BNs as reservoirs is still limited. Snyder *et al*. [38,50] evaluated BN RCs with different numbers of nodes (N) and average node input degrees ($\bar{K}$). Using a measure of reservoir quality, it was found that, compared to networks with homogeneous in-degree, networks with heterogeneous in-degree better separated inputs by class, the so-called separability property [5]. Also, it was found that computational ability - when the difference in separability and fading memory is greatest - was maximized when the average in-degree was $\bar{K}$ = 2 (out of $\bar{K}$=1,2,3) [50]., and that higher $\bar{K}$ led to worse performance. This value for $\bar{K}$ is notable, as Boolean networks with $\bar{K}$ = 2 are known to be dynamically critical (Lyapunov exponent = 0). Critical dynamics, which lie at the border between order and chaos, have been shown to be important in computation and information processing, biological and otherwise [51-55].

While previous work [38,50] has laid the foundation showing that it is possible to use Boolean networks as reservoirs, the main focus has been on testing a small number of well-known functions, such as the median and parity functions, as benchmark measures. We are extending the body of research in three ways: (1) by performing a wider examination of how well these systems perform across different functions, including recursively defined operators, (2) by analyzing how flexible a single reservoir is for repurposing (i.e., retraining only the output layer and keeping the same network reservoir), and (3) characterizing the functions that are easier or harder for the reservoir to implement in terms of the function's average sensitivity, which is a measure of its smoothness. We also evaluate reservoirs under different dynamical regimes: ordered, critical, chaotic.

Materials and Methods

Reservoir Computer

A reservoir computer consists of three components: the input layer, the reservoir network, and the output layer (Fig. 1). The input represents a temporally changing signal that perturbs the reservoir, which performs computations on the signal in real time. The output node is the readout of the reservoir's computation, estimating a given function operating on the signal. The value of the output node is given by a linear combination of the states of the reservoir nodes that are continuously being driven, either directly or indirectly (via other nodes), by the input signal. The weights of the linear regression are trained to be specific to a given objective

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

function, which captures the error between the reservoir output and the function to be approximated.

Reservoir

In our implementation, the reservoir is constructed as a random Boolean network (RBN) [37]. RBNs are networks with N binary-valued nodes with states

$$X_t = \{x_1{}^t, x_2{}^t, \ldots, x_N{}^t\} \qquad x^t{}_i \in \{0,1\}, i = 1, \ldots, N, t > 0.$$

The state of the network at any time, X_t, is a function of the state at the previous time, X_{t-1}, given by

$$X_t = F(X_{t-1})$$

where

$$F = \{f_1, f_2, \ldots, f_N\}$$

is the set of Boolean updating functions corresponding to each node. The node in-degree need not be constant, such that each function $f_i$has an independent number of arguments, k_i, so that

$$x_i{}^t = f_i(x_{j_1}{}^{t-1}, x_{j_2}{}^{t-1}, \ldots, x_{j_{k_i}}{}^{t-1}).$$

The identity of each of the k_i arguments is chosen randomly from the N nodes, without replacement. The in-degree of the whole network is characterized by the average in-degree across its nodes, $\bar{K}$. The other primary descriptor of the network is the bias, p, which is the probability that a function outputs a 1. Together, p and $\bar{K}$ can be varied to adjust the dynamics of the network - from ordered, in which perturbations die out, to chaotic, in which perturbations are amplified [39,40]. For this paper, we will fix p= 0.5 and tune the dynamics of the networks by varying $\bar{K}$. Generally we tune the dynamics using $\bar{K}$< 2 for ordered, $\bar{K}$= 2 for near critical, $\bar{K}$> 2 for chaotic. To construct a network with a specific $\bar{K}$, we take $\bar{K} \cdot N$ edges uniformly distributed amongst all pairs of nodes.

Input

In order for the RBN to act as a reservoir computer, it must be driven by an external signal, represented by a temporal sequence of binary values, u^t. Some percentage (expressed as a fraction) of reservoir nodes, L, are directly connected to the input signal. If a node,i, is connected to the input signal, then the input becomes an additional argument to its function, i.e.,

$$x_i{}^t = f_i(x_{j_1}{}^{t-1}, x_{j_2}{}^{t-1}, \ldots, x_{j_{k_i}}{}^{t-1}, u^{t-1}).$$

The $L \cdot N$ nodes that have the input as an additional argument are chosen uniformly from the N nodes of the reservoir without replacement.

Output

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

At each time step, the reservoir produces a binary output value, $\hat{y}^t$, defined as:

$$\hat{y}^t = \Theta(\sum_{j=1}^{N} w_j x_j^t + b),$$

where Θ is the function that maps values greater than 0.5 to 1, everything else to 0; w_j is a weight for each node of the reservoir and b is a constant. The parameters w_j and b are trained to approximate the output, y^t, of a given Boolean function operating on a moving window over u^t, under some error criterion.

Objective Functions

In this work, the computational performance of the reservoir computer is assessed by approximating Boolean functions of 3 and 5 arguments evaluated over a temporally changing input signal. Moreover, we considered two types of functions

- Non-recursive functions, defined as $y^t = g(u^{t-\tau}, u^{t-\tau-1}, \ldots, u^{t-\tau-(M-1)})$
- Recursive functions defined as $y^t = g(u^{t-\tau}, u^{t-\tau-1}, \ldots, u^{t-\tau-(M-2)}, y^{t-1})$

where M is the length of the sliding window on the input signal for which a function is approximated and τ is a delay between when the signal perturbs the reservoir and when the corresponding output is computed. In this work we explore M= 3,5 for non-recursive functions and M= 3 for recursive functions. τ is kept fixed to τ = 1. We have tested all the 256 3-bit recursive and non-recursive functions, and 1000 randomly sampled 5-bit functions.

Training and Testing algorithm

To train a BN RC for approximating a single function, we use random binary sequences as input streams, u^t, and compare the reservoir's output value, $\hat{y}^t$, to the function output, y^t, evaluated over the input stream. Specifically, we use a set of 150 random binary sequences of length $10 + M - 1$ (e.g. 12 for 3-bit functions) to generate a set of 1,500 different $(\hat{y}, y)$pairs. With the 1,500 $(\hat{y}, y)$ pairs, a regression model is fit in order to generate the coefficient weights on the output layer. We used Scikit-learn to perform lasso regression [56] with an α= 0.1 for fitting.

To test a BN RC for approximating a single function, we again use a set of 150 random binary sequences to generate 1,500 $(\hat{y}, y)$ pairs. We then measure the accuracy of the BN RC for that function, defined as

$$a_{ij} = 1 - \rho / |y|,$$

where j indexes the Boolean function, and i is a particular instantiation of an RC, ρ is the Hamming distance $\sum_t |\hat{y}^t - y^t|$, which is scaled by the length of y, denoted by $|y|$. It can be seen that the accuracy a_{ij} is thus between 0 and 1. This process of training and testing is repeated for a set of different functions for each BN RC, thereby creating multiple estimation accuracies a_{ij}.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

Overall Strategy

The goals of this work center around exploring how the approximation accuracy of RBN RCs varies across many different functions, providing a sense of 'flexibility'. To do that, one reservoir is constructed and applied to estimating the output of a set of Boolean functions, such as all 3-bit functions. This is done for a total of 100 RC instances for each combination of different values of N, L, and $\bar{K}$ to test the effect of reservoir size, degree of network perturbation by the signal, and dynamical regime, respectively. We use N= 10, 20, …, 50, 100, 200, ..., 500; L= 0.1, 0.2, ... ,1; and $\bar{K}$= 1,2,3. For example, with the 3-bit functions, we trained 3(values of $\bar{K}$) * 10(values of N) * 10(values of L) * 100(RCs instances) * 256(3-bit functions) = 7,680,000 RBN RCs.

Each RC is trained and tested for approximating a set of functions, reporting an accuracy for each function. We chose three sets of functions for testing the RCs. First, 3-bit functions, for which we test all possible $2^{2^3} = 256$ functions. Second, 5-bit functions, for which there are $2^{2^5} = 4{,}294{,}967{,}296$ functions, a much larger function space than for 3 bits. Working with the complete function set is impractical, so we made a random sample over the function space, choosing to test 1000 functions. Some additional hand-selected key functions, such as median and parity, were also used.

The last set of functions were recursive 3-bit Boolean functions. This set of functions is defined as taking a set of arguments that includes the last produced output, $y^t = g(u^{t-\tau}, u^{t-\tau-1}, y^{t-1})$. This is a more difficult task because the BN RC must approximate the function with a hidden variable, y^{t-1}.

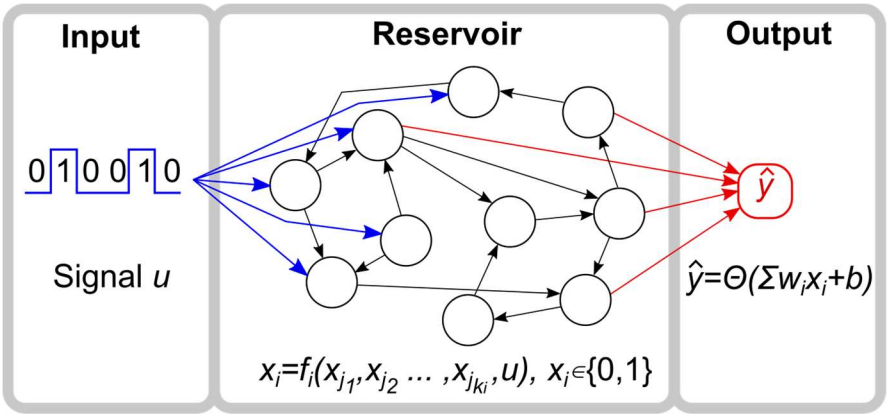


Figure 1. Reservoir computer layout. The RC is composed of a binary input node, a Boolean network reservoir, and a binary output node.

Results

Benchmark Functions: Median and Parity

We start our analysis with the approximation of the two functions tested in previous BN RC works [38,50], the temporal median and parity functions. The median function calculates whether there are more 1s than 0s in the window and is defined as

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

$$d_t = 2\sum_{i=0}^{M-1} u(t-\tau-i) > M$$

The parity function calculates whether there is an odd number of 1s in a bit string. It is defined as

$$p_t = \bigoplus_{i=0}^{M-1} u(t-\tau-i)$$

where $\oplus$ is addition modulo-2. These two functions are frequently used as benchmarks for reservoir performance. For each function g_j, we trained 100 BN RCs for each set of the parameters $\bar{K}$, N, L, measuring the mean accuracy with respect to the 100 BN RCs, $\bar{a}_j = \frac{1}{100}\sum_{i=1}^{100} a_{ij}$.

Median

On average, the 3-bit median function is approximated with high accuracy (Fig. S1). All BN RCs perform better than random, with the lowest mean accuracy of ~0.7 for N= 10, L= 0.1. Increasing N and L increases accuracy logarithmically up to ~0.98 for N= 500, L= 1 (Fig. 2A). However, reservoirs of size N > 200 already have near perfect performance (>0.9) and increasing L has a minimal effect in these reservoirs. The general trend in the accuracy in relation to N and L remains the same for the 5-bit median (Fig. S2). However, the overall accuracy is lower relative to the 3-bit median (Fig. 3A).

The average in-degree,$\bar{K}$, of the reservoirs also affects reservoir performance. For the 3-bit case, the performance is clearly reduced for $\bar{K}$= 3, with those reservoirs having the lowest accuracy. $\bar{K}$ = 1 and 2 have similarly high levels of accuracy (Fig. 2A). However, increasing the size of the function window from 3 to 5 creates a more obvious separation between these $\bar{K}$ values (Fig. 3A). Interestingly, for most values of N and L, $\bar{K}$= 2 reservoirs perform the best; however, for higher values of N and L, accuracy for $\bar{K}$= 1 reservoirs is highest. Overall, these results qualitatively agree with previous work by Snyder *et al* [38,50].

Unexpectedly, when the reservoir is tasked with approximating a 3-bit recursive median, the performance is nearly as high as for the non-recursive 3-bit median and higher than for the 5-bit (non-recursive) median (Fig. 4A). Two notable differences for the recursive median are that (1) increasing L does not grant as sharp of an increase in accuracy and (2) the performance of $\bar{K}$ = 3 is more reduced as compared to lower $\bar{K}$ (Fig. S3).

Parity

The accuracy curves for approximating the 3-bit parity function have a strikingly different appearance than those for the median (Fig. 2B). Accuracy is highly dependent on the value of L, with no reservoir performing better than random (~0.5) for L = 0.1. Very small reservoirs (N < 40) are incapable of high accuracy (Fig. S1). As reservoirs become larger, maximum accuracy increases. When the reservoirs are used to approximate the 5-bit parity function, performance drops significantly (Fig. 3B). No reservoir is capable of performing better than

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

random at L < 0.5 and accuracy > 0.8 is only achievable with larger networks (N > 300) and L near 1 (Fig. S2). These observations are consistent with previous work [38,50].

There is not much separation between approximation accuracy curves by values of $\bar{K}$ for the 3-bit parity, though $\bar{K}$ = 3 shows slightly lower performance on the task (Fig. 2B). For the 5-bit parity, reservoirs with different $\bar{K}$ show a clear difference in accuracy (Fig 3B), with $\bar{K}$ = 2 reservoirs being best. As with approximating the median function, accuracy of $\bar{K}$ = 1 reservoirs improves rapidly with L, changing from lowest to highest accuracy. The relative relationship between the accuracies of different $\bar{K}$-valued reservoirs that we observed is consistent with the above-mentioned previous work, but there are some discrepancies in the actual accuracies observed. We found a much better performance of $\bar{K}$ = 1 and 3 for the 3-bit parity and $\bar{K}$ = 1 and 2 for the 5-bit parity. These differences are likely due to possible differences in how the average in-degree of the reservoir is computed.

The recursive 3-bit parity appears to be extremely difficult to approximate. All reservoirs have an accuracy ~0.5, regardless of N and L, though there is an upward trend in accuracy associated with L. Interestingly, there is no improvement with increasing N and it is unclear whether reservoirs larger than 500 nodes would perform any better. Given the nature of the recursive parity function, it seems that a reservoir would need to "remember" the very first bit of the input signal, which may be too difficult for any reservoir to accomplish.

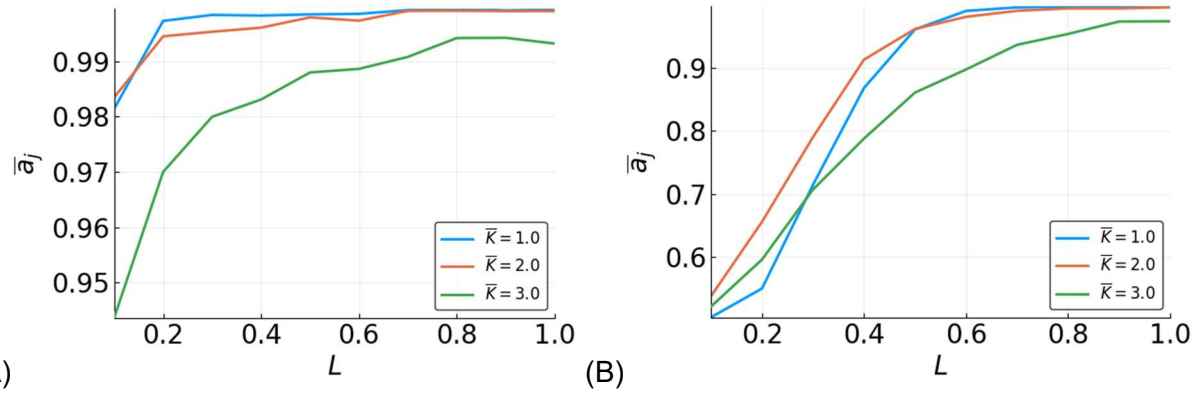
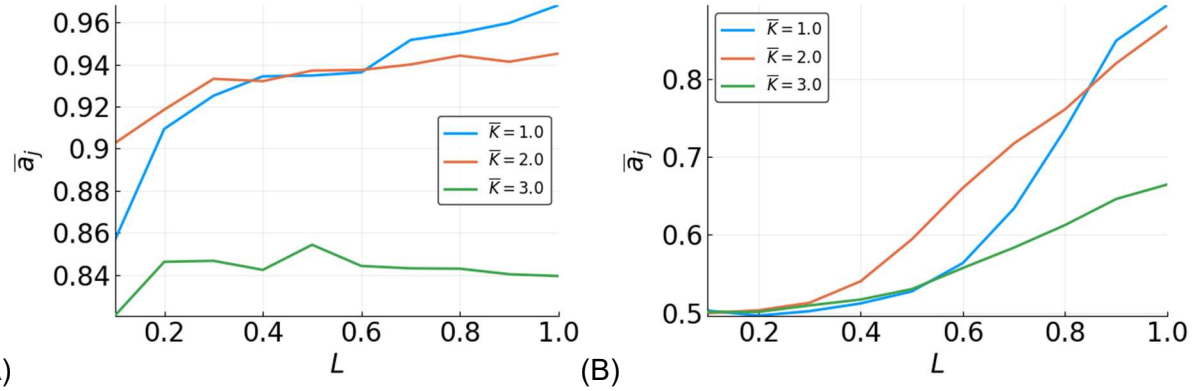


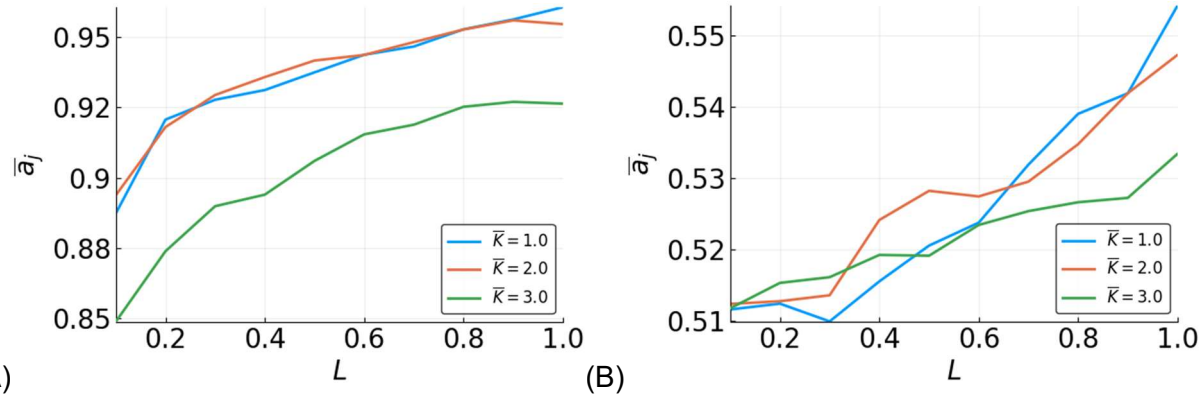
Figure 2. Mean accuracy, $\bar{a}_j$, vs. L for the 3-bit median(A) and parity(B) functions for different $\bar{K}$-valued reservoirs with N = 500.



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at Entropy 2018, 20, 954; doi:10.3390/e20120954

Figure 3. Mean accuracy, $\bar{a}_j$, vs. L for the 5-bit median(A) and parity(B) functions for different $\bar{K}$-valued reservoirs with N = 500.



(A) (B)

Figure 4. Mean accuracy, $\bar{a}_j$, vs. L for the recursive 3-bit median(A) and parity(B) functions for different $\bar{K}$-valued reservoirs with N = 500.

Estimating a Range of Functions

To assess the flexibility of reservoir topologies, all possible 3-bit functions were tested. For each N, L pair, we measured the mean accuracy with respect to all functions and all reservoirs - e.g. $\bar{a} = \frac{1}{100}\frac{1}{256}\sum_{i=1}^{100}\sum_{j=1}^{256} a_{ij}$ for 3-bit functions. Similar to what we observed with median and parity functions, $\bar{K}$ = 3 reservoirs show lower performance on the task (Fig. 5).

The observations made in the 3-bit space were clarified in the 5-bit function space. A sample of 5-bit functions showed similar trends as seen in the 3-bit function set, but with greater separation between dynamical regimes (Fig. 6). However, the ability to approximate functions is noticeably lower compared to 3-bit functions. At N= 100 and L= 0.5, all prediction accuracies are less than 0.75, regardless of $\bar{K}$. Compared to the performance in the 3-bit function space where accuracy was above 0.95 using similar parameters, in the 5-bit function space, it is not until the parameters are maximized that an accuracy of 0.95 is achieved.

Looking at the effect of $\bar{K}$ in the 5-bit function space, accuracy is increased with N and L for all $\bar{K}$, but $\bar{K}$ = 2 is most often the most accurate. The accuracy for $\bar{K}$ = 1 is higher than $\bar{K}$ = 2 after certain values of L. The value of L at which $\bar{K}$ = 1 reservoirs outperform $\bar{K}$ = 2 is decreases as N increases. This effect is also partially seen in the 3-bit functions (Fig. 5) and 3-bit recursive functions (Fig. 7), though the relationship is less clear, except when looking at specific functions, such as the median and parity (Fig. 3).

In terms of recursive functions, accuracy is lower than both non-recursive 3-bit and 5-bit functions. For N = 500, L = 1, and $\bar{K}$ = 2 or less, the accuracy is above 90%. It does not appear that with the current system, accuracy above ~95% is possible. This is likely due to certain functions that are essentially impossible to approximate. This is discussed below, where the temporal order of the arguments to Boolean functions has an effect.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at Entropy 2018, 20, 954; doi:10.3390/e20120954

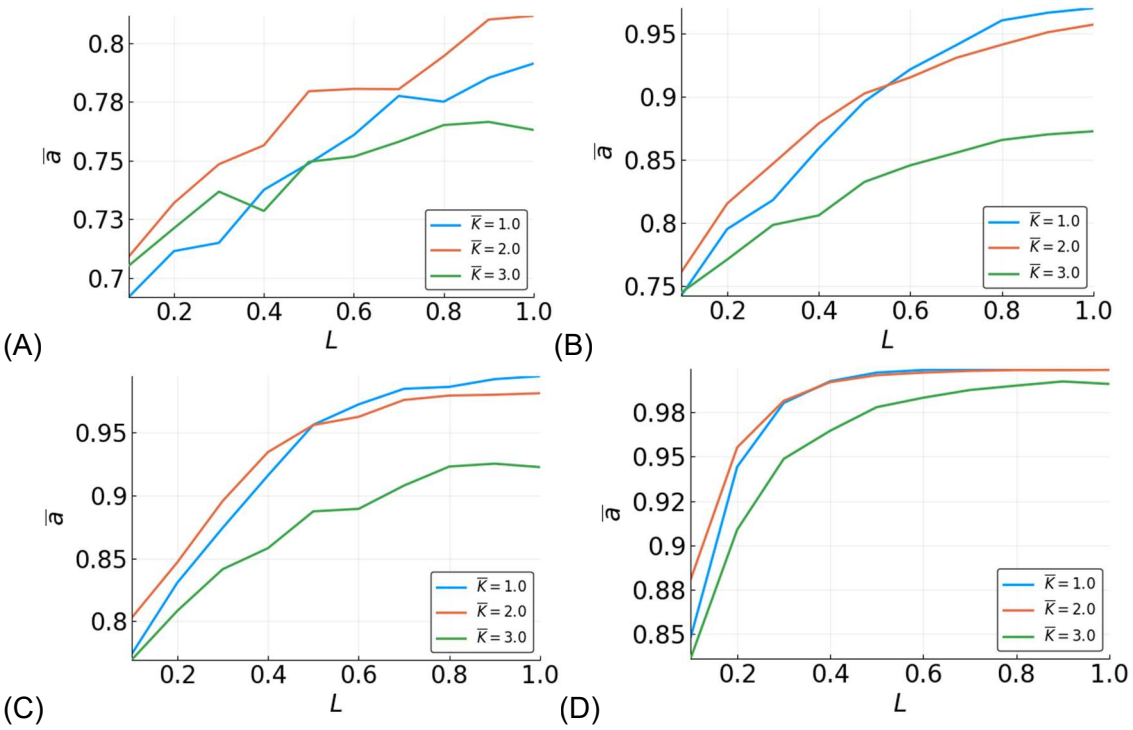


Figure 5. Mean accuracy, $\bar{a}$, vs. L for all 3-bit functions for different $\bar{K}$-valued reservoirs. Different sizes of reservoirs are shown: N = 10 (A), N = 50 (B), N = 100 (C), and N = 500 (D).

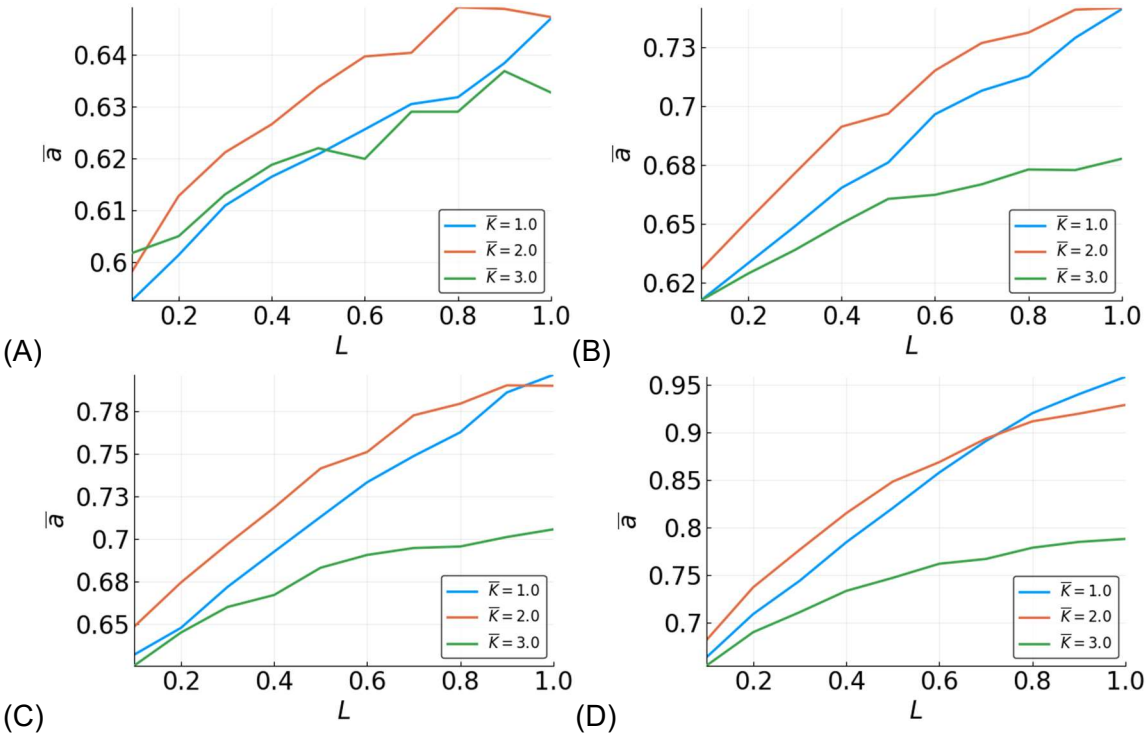


Figure 6. Mean accuracy, $\bar{a}$, vs. L for 5-bit functions for different $\bar{K}$-valued reservoirs. Different sizes of reservoirs are shown: N=10 (A), N = 50 (B), N = 100 (C), and N = 500 (D).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

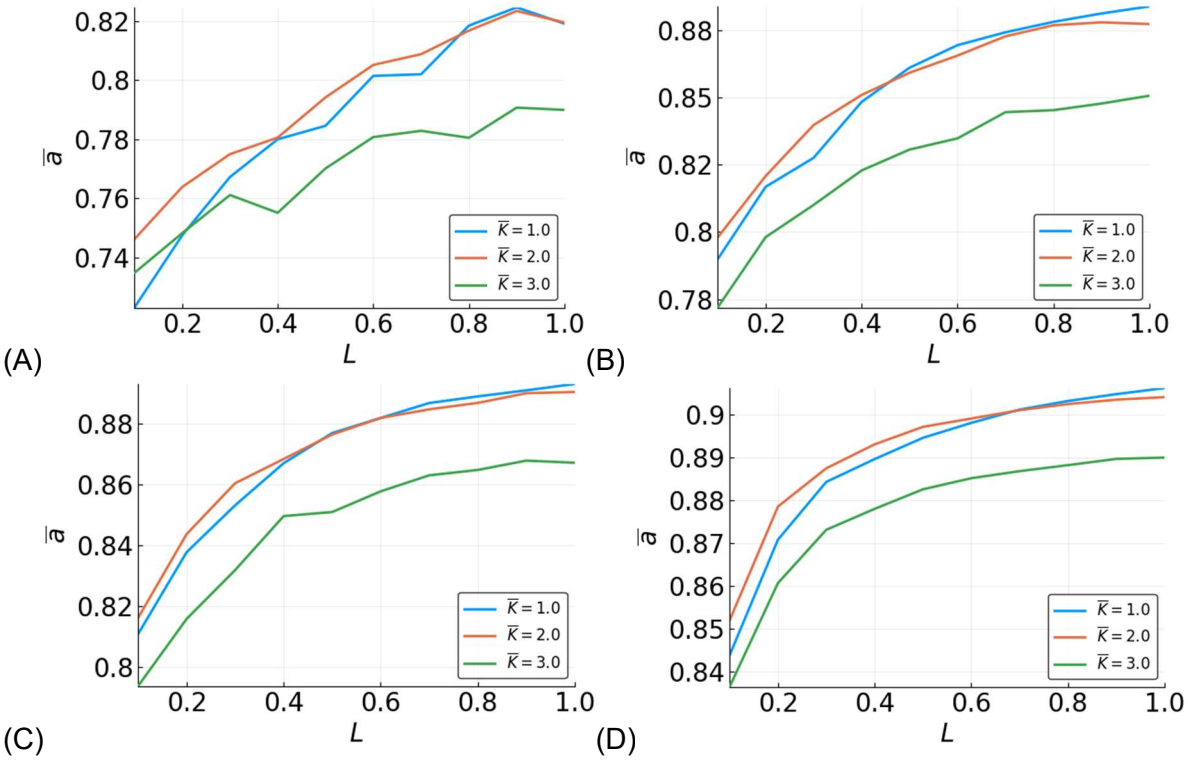


Figure 7. Mean accuracy, $\bar{a}$, vs. L for all recursive 3-bit functions for different $\bar{K}$-valued reservoirs. Different sizes of reservoirs are shown: N=10 (A), N = 50 (B), N = 100 (C), and N= 500 (D).

Reservoir Flexibility

One goal of this work was to assess the flexibility of reservoirs. A highly flexible reservoir can approximate a high number of functions without any changes to the network topology or size. To assess reservoir flexibility, we compute a flexibility metric Φ_i for each reservoir i, defined as,

$$\Phi_i = median(a_{ij} - 0.5)/mad(a_{ij}),$$

where the median and mad (median absolute deviation from the median) are taken over all functions g_j. A reservoir with a low Φ_i value cannot approximate many (or any) functions well. As the number of functions a reservoir can approximate increases, so does Φ_i.

For each combination of N and L, we have a distribution of Φ_i values, one for each of 100 BN RCs. These can be visualized as density curves (Fig. 8). With increasing values of N and L, the distributions become increasingly right skewed and sharply peaked - low N, L values being characterized mostly by reservoirs with low Φ and high N, Lvalues with high Φ. However, for many intermediary values of N and L, there is a rather flat distribution of Φ values, indicating that there is heterogeneity of flexibility in performance for reservoirs with these parameters (i.e., some are highly flexible while some are not). The gradation of distribution shape with increasing

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at *Entropy* **2018**, *20*, 954; doi:10.3390/e20120954

N and Ldepends on the value of $\bar{K}$ (Fig. S4A,B), though the overall trend remains. For higher values of $\bar{K}$, the transition point towards higher accuracies shifts towards higher values of L, N.

The flexibility of reservoirs approximating 5-bit functions is markedly reduced compared to 3-bit functions (Fig. S5A,C,E). For all but the highest values of $N$and L, most reservoirs have only moderate flexibility (Φ = 0.25. Unlike for 3-bit functions, the shape of the distributions remain the same across N and L. Consistent with the effect of $\bar{K}$ on mean accuracy, the Φ distributions for $\bar{K}$ = 1 and $\bar{K}$ = 3 are shifted to lower values as compared to $\bar{K}$ = 2.

Surprisingly, the distributions of Φ values for reservoirs approximating recursive 3-bit functions behave differently than those of non-recursive 3-bit functions. This was unexpected, since the mean accuracy curves (Fig. 7) for the recursive functions resemble those for the non-recursive functions (Fig. 5). While Φ increases with $N$and L as it does for non-recursive 3-bit functions, the change in the distributions does not follow the same pattern (Fig. S5B,D,F). For low N, L the distribution of Φ values is wide and becomes increasingly narrow and peaked with increasing N and L, as compared to the narrow peaky edges and flat wide middle of the non-recursive 3-bit Φ distributions.

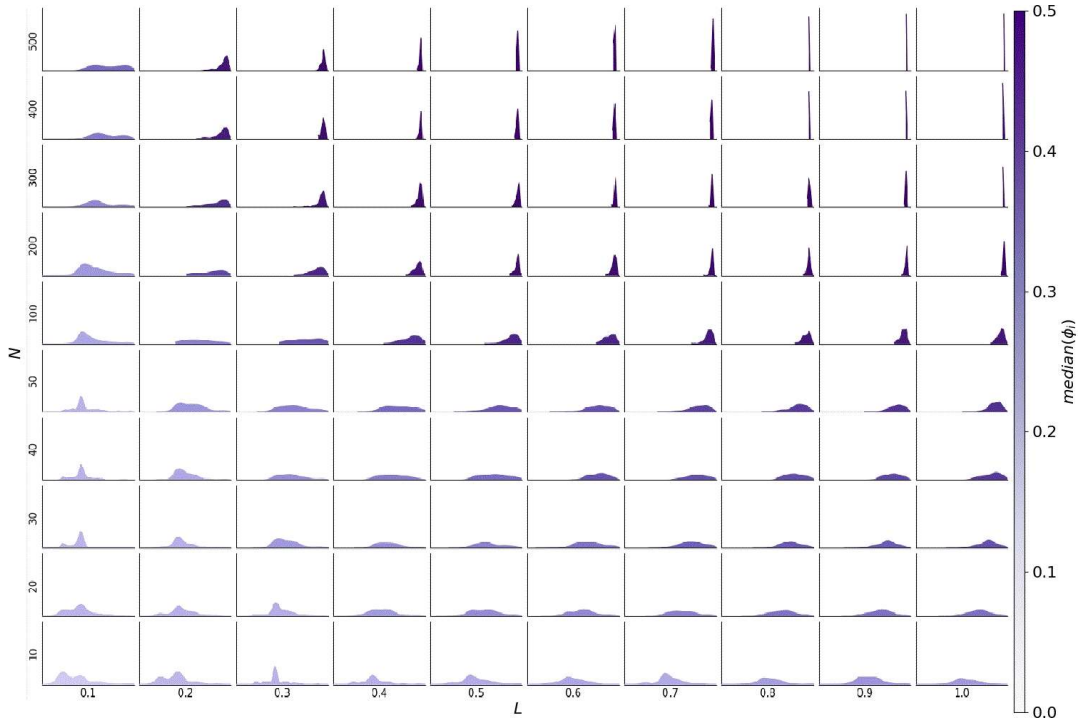


Figure 8. Histogram of Φ_i across all 100 reservoirs for each N, L with $\bar{K}$ = 2 for 3-bit functions. Each subplot represents the density for all the reservoirs with one N and L, with the x-axis being Φ and the y-axis being number of reservoirs[0,256].

Determinants of Difficulty

To better understand why reservoirs do not perform uniformly well for all Boolean functions of a given size, we investigated possible factors related to the functions to be

approximated that could contribute to lower estimation accuracy. One possible way to compare Boolean functions with the same number of input variables is via the average sensitivity [57,58] of the function, $\bar{s}_g$. As the name suggests, this metric evaluates how sensitive the output of a function is to any change in the inputs. The average sensitivity of a function g is given by

$$\bar{s}_g = E[s_g([u^1, u^2, \ldots, u^M])]$$

where $$s_g([u^1, u^2, \ldots, u^M]) = \sum_{i=1}^{M} \chi[g([u^1, u^2, \ldots, u^M]) \oplus e_i \neq g([u^1, u^2, \ldots, u^M])]$$

and e_i is the unit vector with a 1 in the ith position and 0s elsewhere and $\chi[A]$ is an indicator function that gives a 1 if A is true and 0 if A is false. The expectation is typically taken with respect to a uniform distribution over the M-dimensional hypercube.

A function that is insensitive has a low average sensitivity. For example, the constant function, $g([u^1, u^2, u^3]) = 1$, is not dependent on any of its inputs and thus, $\bar{s}_g$ = 0. On the other hand, the 3-bit parity, $g([u^1, u^2, u^3]) = u^1 \oplus u^2 \oplus u^3$, has a high average sensitivity as it will take a different value if any of its input variables are toggled, making $\bar{s}_g = M$ = 3, the maximum possible value. The sensitivity can be interpreted as the smoothness of a function. We hypothesized that a function with high sensitivity would be more difficult to estimate because it requires more information about the inputs and is harder to generalize.

To investigate whether there is a relationship between average sensitivity and accuracy, we computed the average accuracy over all reservoirs approximating all the functions with a given average sensitivity, $\bar{a}_s$ (Fig. 9 and 10). Here, we only discuss results for $\bar{K}$ = 2 since the results for $\bar{K}$ = 1,3 are comparable (Fig. S6). For 3-bit and 5-bit functions there is a clear inverse linear relationship between $\bar{s}_g$ and accuracy. As the average accuracy increases with N and L, the slope of the line becomes less steep, but the relationship remains. Additionally, approximation of functions with greater $\bar{s}_g$ does not improve as much with larger reservoirs. For low N, the relationship between sensitivity and accuracy for recursive functions closely matches that for the non-recursive functions. However, as N increases, reservoir performance for recursive functions with high sensitivity remains relatively unimproved.

While the relationship between the average sensitivity and accuracy appears mostly linear, there are clear points of deviation from linearity, most notably at $\bar{s}_g$ = 1, 2, 3, 4. Functions with the same average sensitivities can have different degrees of dependence on each variable. The effect of each variable, u^i, on the output can be measured by the activity, α_g^i, of that variable in function g [57,59]. The activity of a variable is the probability (hence, α_g^i is between 0 and 1) that toggling the variable's value changes the output of the function. If a uniform distribution of input states is assumed, the sum of the activities is equal to the average sensitivity.

We examined the distribution of activities for 3-bit functions and found that the points of deviation correspond to groups of functions with the same sensitivity, but different possible activities, e.g., $\bar{s}_g = 1$, $A_g = [0, 0.5, 0.5]$ or $A_g = [1, 0, 0]$, where $A_g = [\alpha_g^1, \alpha_g^2, \alpha_g^3]$ (Fig. 11A). In these cases, reservoir accuracy generally decrease when variables observed farthest in the past have higher activities. Thus, the performance of the reservoir is not only dependent on the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at Entropy 2018, 20, 954; doi:10.3390/e20120954

sensitivity of the function, but also the temporal distribution of activities (Fig. 11B). For 5-bit functions, the relationship between accuracy and activities is not as well defined, though still identifiable (data not shown).

Recursive functions behave similarly to non-recursive functions with some exceptions (Fig. S7A,B). Since the recursive variable is the most temporally distant, having high activity on this variable causes an even greater reduction in accuracy. In fact, reservoirs cannot approximate any recursive functions with $\alpha_g^3 = 1$. There are also some cases in which reservoirs approximating functions with high activity on the recursive variable have higher accuracy compared to others with the same sensitivity. It is possible that this is related to the recursive nature of the function.

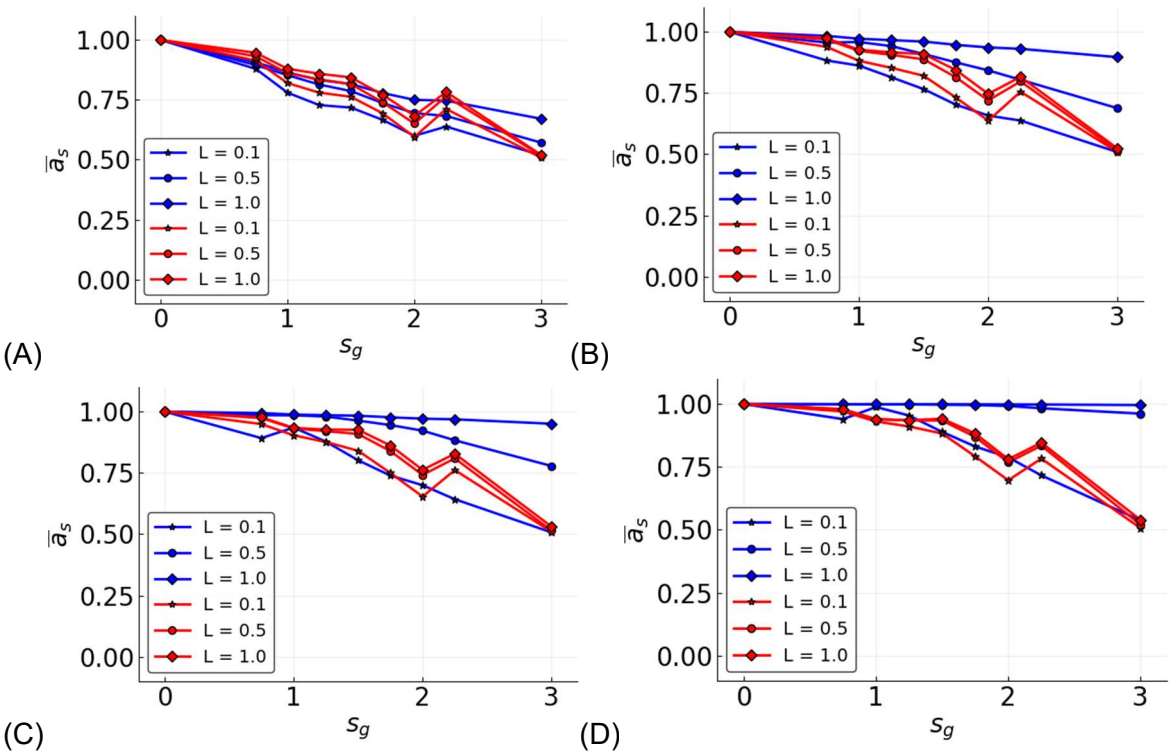


Figure 9. Mean accuracy, $\bar{a}_s$, vs. function average sensitivity, $\bar{s}_g$. 3-bit functions shown in blue and recursive 3-bit functions shown in red with L = 10 (stars), 50(circles), 100(diamonds). Four different values for N are shown - (A) N = 10. (B) N = 50. (C) N = 100, and (D) N = 500. Only reservoirs with $\bar{K}$ = 2 are shown. See supplemental materials for $\bar{K}$ = 1 and $\bar{K}$ = 3 (Fig. S6).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 November 2018 doi:10.20944/preprints201811.0203.v1

Peer-reviewed version available at Entropy 2018, 20, 954; doi:10.3390/e20120954

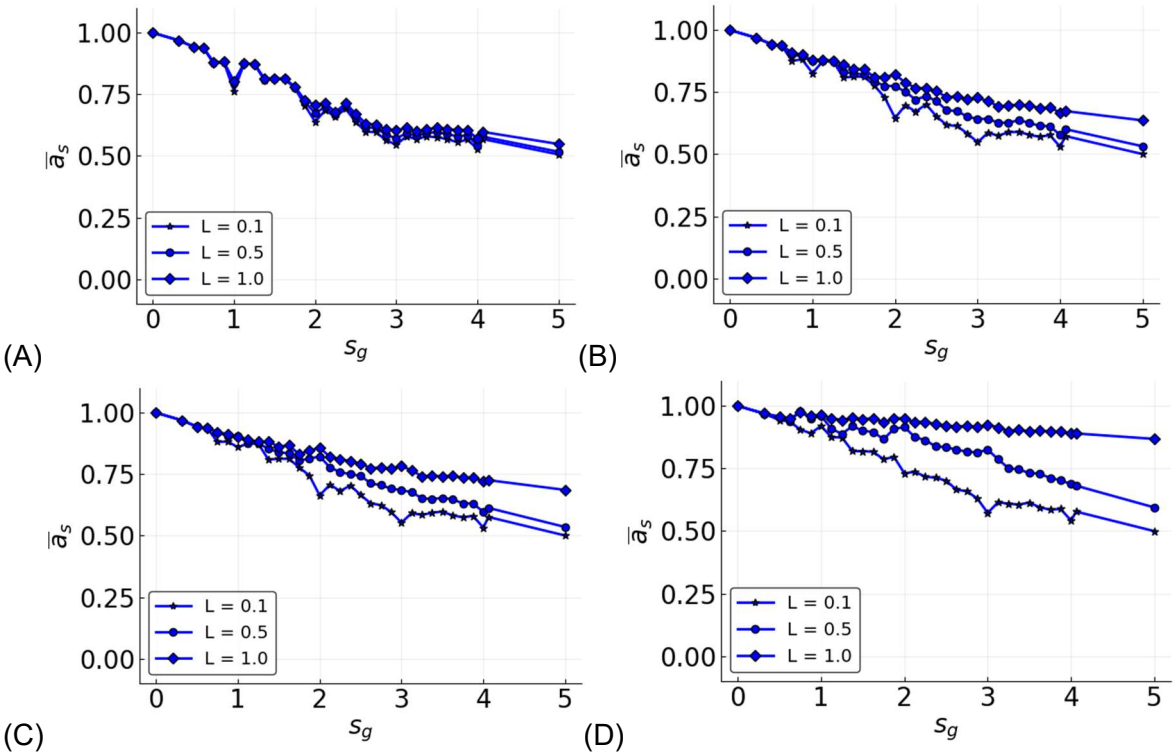


Figure 10. Mean accuracy, $\bar{a}_s$, vs. function average sensitivity, $\bar{s}_g$ for 5-bit functions. L = 0.1 (stars), .0.5(circles), and 1(diamonds) are given in each plot. Four different values for N are shown - (A) N = 10. (B) N = 50. (C) N = 100, and (D) N = 500. Only reservoirs with $\bar{K}$= 2 are shown.

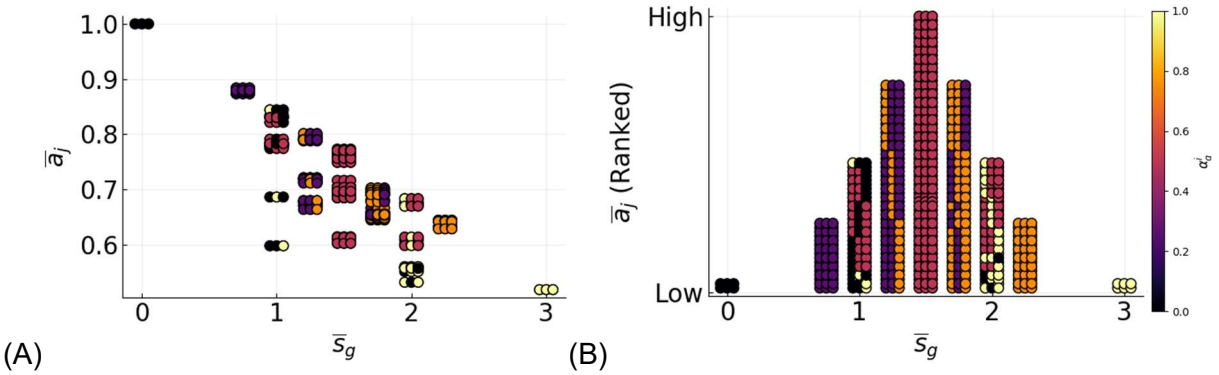


Figure 11. Example of mean function accuracy vs. average sensitivity with activities of each variable displayed. Data shown for 3-bit functions - N = 10, L= 0.1, and $\bar{K}$ = 2. (A) Each of 256 functions is visualized as a horizontal triplet of circles, where each circle corresponds to a variable (left to right,$u^{t-\tau}$,$u^{t-\tau-1}$,$u^{t-\tau-2}$), colored by its activity. For example, the parity function can be seen at $\bar{s}_g$= 3 and A_g= [1,1,1] (B)In order to more clearly see the relationship between distribution of activity and accuracy, functions are plotted by ranked accuracy rather than absolute accuracy.

Discussion

In this work, we found that the flexibility of a particular BN RC instantiation is a function of the topology and size of the reservoir, encoded by the parameter set ($N, L, \bar{K}$). Generally, higher values of N and L result in more flexible reservoirs; however, there is heterogeneity when N or L are low and flexible reservoirs can be found at these values. As noted in research concerning BN RCs, the optimal parameter set includes tuning the dynamics towards criticality, which leads to more flexible systems. For signal processing, a flexible reservoir will be more accurate and more efficient, requiring less searching and training when different filters are being applied to the same data. For biological systems, where survival is dependent on signal response, N, L, and $\bar{K}$ can be tuned to balance the restraints of the system with the demands of the environment.

In terms of the challenge we provided to the RCs, we found 3-bit functions to be relatively easy to approximate, while 5-bit functions were more difficult, essentially requiring more resources. However 3-bit recursive functions were the most difficult where we noted some recursive functions that are essentially impossible to approximate with this system, likely due to a long memory requirement.

Approximating the 5-bit functions depend on a large reservoir and input size, and more strongly depend on $\bar{K}$. The 5-bit function space is massive, so while we were only able to sample a tiny fraction of it, we clearly saw the effect of dynamics close to criticality on approximation accuracy. As $\bar{K}$ increased towards chaotic dynamics, the approximation became very poor, leading to large separation between dynamical regimes. Looking forward, it is likely that 7-bit or 9-bit function approximations would require increasingly more resources in terms of reservoir size with more of a dependence on dynamics. This seems to be the case for the median and parity functions, at least [50]. We found that the relative accuracy of reservoirs with $\bar{K}$ = 1, 2, or 3 could change, depending on the value of L- the input connectivity into the reservoir. This is most evident with the 5-bit functions in the $\bar{K}$ =2 accuracy plot, which shows a tapering off curve (convex) where improvement with reservoir size is not linear (Fig. 6). In the curves, as L increases, we see the $\bar{K}$ = 1 and $\bar{K}$ = 2 accuracy curves crossing each other. One possible explanation for this is that the external perturbations to the reservoir's nodes shift the dynamics of the reservoir to a less ordered regime - a higher fraction of perturbed nodes resulting in a greater shift. So the dynamics in $\bar{K}$ = 1 approach criticality with higher L, while the dynamics in the $\bar{K}$ = 2 case moving past criticality, towards chaos, thus dropping in approximation accuracy. However, the effect of time-varying external perturbation has not been well-studied for Boolean network dynamics. In [50] , it is shown that the mutual information between the reservoir and the signal increase monotonically as L increases for $\bar{K}$ = 1, so there may be other effects of increasing L at play. Although our results indicate a strong effect of mean in-degree ($\bar{K}$) in accuracy, it is possible that accuracy is also affected by other properties of reservoir topology; additional in-degree distributions should be investigated.

We found that Boolean function sensitivity also plays a role in the accuracy, where more sensitive functions are more difficult to approximate. The sensitivity can be broken down further into the activities of the function variables. Functions appear to be more difficult to approximate

if the activities of temporally distant variables are higher. For non-recursive functions, increasing N and L are sufficient to enable BN RCs to approximate sensitive functions, however, for non-recursive functions, large reservoirs with high values of L are not as affected. This points to a large portion of the reservoir needing perturbation to avoid problems related to function argument sensitivity.

However, for recursive functions, even large reservoirs are affected by Boolean function sensitivity. This is likely due to the non-smooth output, which is difficult to approximate. Further, the recursive sliding-window filtering formulation essentially creates infinite memory, albeit fading. Many questions remain, such as what is be needed for good approximations of recursive functions, and whether there is a change to the model that might help. That said, for the most part, many recursive functions are approximated very well, and that may be enough.

Overall, having an idea of the sensitivity of the processing task will inform the minimum parameter values necessary for a successful reservoir. This is intriguing from a biological standpoint and worth exploring how signal responses that are more sensitive, especially to more historic signal values, are handled. The inability of approximating certain functions may not impinge on the use for modeling biology. When viewed through the lens of 'complex adaptive systems', recursion is a key characteristic, and one that BN RCs would need to address in order to model biological systems.

Conclusion

Reservoir computers with adequately sized Boolean network reservoirs and input sizes show that even a fixed topology is flexible enough to approximate most Boolean functions, applied to signals in a sliding-window fashion, with high accuracy. On one hand, BN RCs have structural similarity to biological systems like gene regulatory networks, while on the other hand, they are useful objects for operating on signals, such as through the application of recursive filters to biological data. We observe that Boolean networks that are closer to the critical dynamics regime lead to more flexible reservoirs. Recursive functions are found to be the most difficult to approximate, owing to the necessity for approximating a function with hidden variables. Although most recursive functions can be approximated, we find that some can never be approximated. Boolean functions with more arguments are more strongly dependent on the dynamics of the system for accuracy, leading to a greater separation in accuracy curves corresponding to different dynamical regimes. We find a connection between Boolean function sensitivity, an estimate of the smoothness of a function, and the ability to approximate a given function.

Supplementary Materials

Additional figures can be found in the Supplement. Software for constructing and running BN RCs can be found at *https://github.com/IlyaLab/BooleanNetworkReservoirComputers.*

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No.DGE-1762114. We also would like to acknowledge the Institute for Systems Biology for providing institutional support.

Author Contributions

Conceptualization, M.E. and I.S.; Methodology, D.G., M.N., M.E., and I.S.; Software, M.N., B.A., and D.G.; Validation, M.N., and B.A.; Resources, I.S.; Writing – Original Draft Preparation, M.E. and D.G.; Writing – Review & Editing, M.E., D.G., B.A., I.S.; Visualization, M.E.; Supervision, I.S.; Project Administration, D.G.; Funding Acquisition, I.S.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Dasgupta S, Stevens CF, Navlakha S. A neural algorithm for a fundamental computing problem. Science. science.sciencemag.org; 2017;358: 793–796.

2. Jaeger H. Adaptive Nonlinear System Identification with Echo State Networks. In: Becker S, Thrun S, Obermayer K, editors. Advances in Neural Information Processing Systems 15. MIT Press; 2003. pp. 609–616.

3. Kitano H. Systems biology: a brief overview. Science. science.sciencemag.org; 2002;295: 1662–1664.

4. Shivdasani RA. Limited gut cell repertoire for multiple hormones. Nat Cell Biol. nature.com; 2018;20: 865–867.

5. Maass W, Natschläger T, Markram H. Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. 2002;14: 2531–2560.

6. McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys. 1943;5: 115–133.

7. Lu Z, Pathak J, Hunt B, Girvan M, Brockett R, Ott E. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. Chaos. 2017;27. doi:10.1063/1.4979665

8. Pathak J, Lu Z, Hunt RB, Girvan M, Ott E. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. Chaos. 2017;27. doi:10.1063/1.5010300

9. Fonollosa J, Sheik S, Huerta R, Marco S. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. Sens Actuators B Chem. 2015;215. doi:10.1016/j.snb.2015.03.028

10. Caluwaerts K, D'Haene M, Verstraeten D, Schrauwen B. Locomotion without a brain: physical reservoir computing in tensegrity structures. Artif Life. MIT Press; 2013;19: 35–66.

11. Aaser P, Knudsen M, Ramstad HO, van de Wijdeven R, Nichele S, Sandvig I, et al. Towards making a cyborg: A closed-loop reservoir-neuro system. MIT Press; 2016. pp. 430–437.

12. Antonelo AE, Schrauwen B, Van Campenhout J. Generative Modeling of Autonomous Robots and their Environments using Reservoir Computing. Neural Process Letters. 2007;26. doi:10.1007/s11063-007-9054-9

13. Bianchi FM, Santis ED, Rizzi A, Sadeghian A. Short-Term Electric Load Forecasting Using Echo State Networks and PCA Decomposition. IEEE Access. ieeexplore.ieee.org; 2015;3: 1931–1943.

14. Gallicchio C, 2014 EAM-FIW, 2014. A preliminary application of echo state networks to emotion recognition. 2011;

15. Gallicchio C, Ia AAM-AA, 2016. A Reservoir Computing Approach for Human Gesture Recognition from Kinect Data. 2013;

16. Waibel A. Modular Construction of Time-Delay Neural Networks for Speech Recognition. Neural Comput. MIT Press; 1989;1: 39–46.

17. Triefenbach F, Jalalvand A, Schrauwen B, Martens J-P. Phoneme Recognition with Large Hierarchical Reservoirs. In: Lafferty JD, Williams CKI, Shawe-Taylor J, Zemel RS, Culotta A, editors. Advances in Neural Information Processing Systems 23. Curran Associates, Inc.; 2010. pp. 2307–2315.

18. Palumbo F, Gallicchio C, Pucci R, Micheli A. Human activity recognition using multisensor data fusion based on Reservoir Computing. J Ambient Intell Smart Environ. 8. doi:10.3233/AIS-160372

19. Luz EJ da S, Schwartz WR, Cámara-Chávez G, Menotti D. ECG-based heartbeat classification for arrhythmia detection: A survey. Comput Methods Programs Biomed. Elsevier; 2016;127: 144–164.

20. Merkel C, Saleh Q, Donahue C, Kudithipudi D. Memristive Reservoir Computing Architecture for Epileptic Seizure Detection. Procedia Comput Sci. Elsevier; 2014;41: 249–254.

21. Buteneers P, Verstraeten D, van Mierlo P, Wyckhuys T, Stroobandt D, Raedt R, et al. Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing. Artif Intell Med. Elsevier; 2011;53: 215–223.

22. Ayyagari S. Reservoir computing approaches to EEG-based detection of microsleeps. University of Canterbury; 2017;

23. Kainz P, Burgsteiner H, Asslaber M, Ahammer H. Robust Bone Marrow Cell Discrimination by Rotation-Invariant Training of Multi-class Echo State Networks. Engineering Applications of Neural Networks. Springer International Publishing; 2015. pp. 390–400.

24. Reid D, Barrett-Baxendale M. Glial Reservoir Computing. Second UKSIM European Symposium on Computer Modeling and Simulation. IEEE Computer Society; 2008. pp. 81–86.

25. Enel P, Procyk E, Quilodran R, Dominey PF. Reservoir Computing Properties of Neural Dynamics in Prefrontal Cortex. PLoS Comput Biol. journals.plos.org; 2016;12: e1004967.

26. Yamazaki T, Tanaka S. The cerebellum as a liquid state machine. Neural Netw. 2007;20: 290–297.

27. Dai X. Genetic Regulatory Systems Modeled by Recurrent Neural Network. Lecture Notes in Computer Science. 2004. pp. 519–524.

28. Jones B, Stekel D, Rowe J, Fernando C. Is there a Liquid State Machine in the Bacterium Escherichia Coli? 2007 IEEE Symposium on Artificial Life. 2007. doi:10.1109/alife.2007.367795

29. Tibshirani R. Regression Shrinkage and Selection via the Lasso. J R Stat Soc Series B Stat Methodol. [Royal Statistical Society, Wiley]; 1996;58: 267–288.

30. Lynn PA. Recursive digital filters for biological signals. Med Biol Eng. 1971;9: 37–43.

31. Burian A, Kuosmanen P. Tuning the smoothness of the recursive median filter. IEEE Trans Signal Process. ieeexplore.ieee.org; 2002;50: 1631–1639.

32. Shmulevich I, Yli-Harja O, Egiazarian K, Astola J. Output distributions of recursive stack filters. IEEE Signal Process Lett. 1999;6: 175–178.

33. Dambre J, Verstraeten D, Schrauwen B, Massar S. Information processing capacity of dynamical systems. Sci Rep. nature.com; 2012;2: 514.

34. Fernando C, Sojakka S. Pattern Recognition in a Bucket. Advances in Artificial Life. Springer Berlin Heidelberg; 2003. pp. 588–597.

35. Kulkarni SM, Teuscher C. Memristor-based reservoir computing. ACM Press; 2009. pp. 226–232.

36. Dale M, Miller JF, Stepney S, Trefzer MA. Evolving Carbon Nanotube Reservoir Computers. Unconventional Computation and Natural Computation. Springer International Publishing; 2016. pp. 49–61.

37. Kauffman SA. Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biol. 1969;22: 437–467.

38. Snyder D, Goudarzi A, Life ACT-, 2012. Finding optimal random boolean networks for reservoir computing. 2009;

39. Derrida B, Pomeau Y. Random networks of automata: a simple annealed approximation. EPL. IOP Publishing; 1986;1: 45.

40. Luque B, Solé RV. Lyapunov exponents in random Boolean networks. Physica A: Statistical Mechanics and its Applications. Elsevier; 2000;284: 33–45.

41. Van der Sande G, Brunner D, Soriano MC. Advances in photonic reservoir computing. Nanophotonics. degruyter.com; 2017;6: 8672.

42. Shmulevich I, Dougherty ER, Zhang W. From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. Proc IEEE. ieeexplore.ieee.org; 2002;90: 1778–1792.

43. de Jong H. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. J Comput Biol. Mary Ann Liebert, Inc., publishers; 2002;9: 67–103.

44. Davidich MI, Bornholdt S. Boolean network model predicts cell cycle sequence of fission yeast. PLoS One. 2008;3: e1672.

45. Fumiã HF, Martins ML. Boolean network model for cancer pathways: predicting carcinogenesis and targeted therapy outcomes. PLoS One. 2013;8: e69008.

46. Serra R, Villani M, Barbieri A, Kauffman SA, Colacci A. On the dynamics of random Boolean networks subject to noise: attractors, ergodic sets and cell types. J Theor Biol. 2010;265: 185–193.

47. Helikar T, Konvalina J, Heidel J, Rogers JA. Emergent decision-making in biological signal transduction networks. Proc Natl Acad Sci U S A. 2008;105: 1913–1918.

48. Thakar J, Pilione M, Kirimanjeswara G, Harvill ET, Albert R. Modeling systems-level regulation of host immune responses. PLoS Comput Biol. 2007;3: e109.

49. Damiani C, Kauffman SA, Villani M, Colacci A, Serra R. Cell–cell interaction and diversity of emergent behaviours. IET Syst Biol. 2011;5: 137–144.

50. Snyder D, Goudarzi A, Teuscher C. Computational capabilities of random automata networks for reservoir computing. Phys Rev E Stat Nonlin Soft Matter Phys. APS; 2013;87: 042808.

51. Bertschinger N, Natschläger T. Real-time computation at the edge of chaos in recurrent neural networks. Neural Comput. 2004;16: 1413–1436.

52. Balleza E, Alvarez-Buylla ER, Chaos A, Kauffman S, Shmulevich I, Aldana M. Critical dynamics in genetic regulatory networks: examples from four kingdoms. PLoS One. 2008;3: e2456.

53. Goudarzi A, Teuscher C, Gulbahce N, Rohlf T. Emergent criticality through adaptive information processing in boolean networks. Phys Rev Lett. 2012;108: 128702.

54. Torres-Sosa C, Huang S, Aldana M. Criticality is an emergent property of genetic networks that exhibit evolvability. PLoS Comput Biol. 2012;8: e1002669.

55. Muñoz MA. Colloquium : Criticality and dynamical scaling in living systems. Rev Mod Phys. 2018;90. doi:10.1103/revmodphys.90.031001

56. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res. 2011;12: 2825–2830.

57. Shmulevich I, Kauffman SA. Activities and sensitivities in boolean network models. Phys Rev Lett. APS; 2004;93: 048701.

58. Cook S, Dwork C, Reischuk R. Upper and Lower Time Bounds for Parallel Random Access Machines without Simultaneous Writes. SIAM J Comput. 1986;15: 87–97.

59. Kahn J, Kalai G, Linial N. The Influence of Variables on Boolean Functions. Proceedings of

the 29th Annual Symposium on Foundations of Computer Science. Washington, DC, USA: IEEE Computer Society; 1988. pp. 68–80.