2-8-2021

Noise optimizes super-Turing computation in recurrent neural networks

Emmett Redd
Missouri State University

Steven Senger
Missouri State University

Tayo Obafemi-Ajayi
Missouri State University

Noise optimizes super-Turing computation in recurrent neural networks

Emmett Redd,[1,*] Steven Senger,[2] and Tayo Obafemi-Ajayi[3]

[1]*Physics, Astronomy, and Materials Science Department, Missouri State University, Springfield, Missouri 65987 USA*
[2]*Mathematics Department, Missouri State University, Springfield, Missouri 65987 USA*
[3]*Engineering Program, Missouri State University, Springfield, Missouri 65987 USA*

(Received 21 November 2019; accepted 21 January 2021; published 8 February 2021)

This paper explores the benefit of added noise in increasing the computational complexity of digital recurrent neural networks (RNNs). The physically accepted model of the universe imposes rational number, stochastic limits on all calculations. An analog RNN with those limits calculates at the super-Turing complexity level BPP/log*. In this paper, we demonstrate how noise aids digital RNNs in attaining super-Turing operation similar to analog RNNs. We investigate moving limited-precision systems from not being chaotic at small amounts of noise, through consistency with chaos, to overwhelming it at large amounts of noise. A Kolmogorov-complexity-based proof shows that an infinite computational class hierarchy exists between P, the Turing class, and BPP/log*. The hierarchy offers a possibility that the noise-enhanced digital RNNs could operate at a super-Turing level less complex than BPP/log*. As the uniform noise increases, the digital RNNs develop positive Lyapunov exponents intimating that chaos is mimicked. The exponents maximize to the accepted values for the logistic and Hénon maps when the noise equals eight times the least significant bit of the noisy recurrent signals for the logistic digital RNN and four times the Hénon digital RNN.

DOI: 10.1103/PhysRevResearch.3.013120

I. INTRODUCTION

Noise improves deep learning and is recognized as an important contributor to neural networks (NNs) in artificial intelligence and neuroscience [1–3]. Redd *et al.* [4] demonstrated how stochastic resonance (SR) is the underlying principle of analog recurrent neural networks (RNNs) attaining super-Turing machine capability. The noise benefit experienced in deep learning systems is essentially a type of stochastic resonance where a small amount of noise improves the performance of a nonlinear system while too much noise harms the performance [5]. In general, an SR benefiting system is observed when the output signal from a system provides a better representation of the input signal (or of some useful aspect of it) than it would in the complete absence of noise [3]. There is a vast literature on the benefits of noise injection in neural networks [6–10]. The classic theory of stochastic resonance has two components, noise and periodic forcing functions [11]. However, some practical uses of SR rely only on the noise and not the periodic forcing, such as in increasing precision of analog-to-digital converters or subthreshold signal detection. This paper explores the benefit of this SR noise from a different perspective: computational complexity class. The overall hypothesis is that noise aids digital RNNs in moving closer to attaining super-Turing machine complexity.

At various levels of correspondence, artificial neural networks implement operations analogous to biological neural networks [12,13]. Since digital computers are inspired by Turing's computational model [14], many artificial neural network implementations do not seek a more powerful computational model than a universal Turing machine (TM) which computes the complexity class P. This paper will describe *noise-enhanced* digital RNNs seeking to approach a more powerful computational complexity model. Previous effort [4,15,16] has indicated computation results consistent with super-Turing complexity. Super-Turing computational models describe classes of machines that are stronger than digital/deterministic TMs. Siegelmann discovered that a rational stochastic RNN computes the BPP/log* complexity class [17–19]. Younger *et al.* [15,16] demonstrated the implementation of optical analog RNN hardware for computing beyond the Turing limit. In [4], the focus was on concepts of analog signals which differ significantly from the digital techniques currently used to develop NNs. The main thrust was that the SR contained in artificial intelligence and neuroscience is inherent in the proof of super-Turing BPP/log* networks and likely contributed to the hardware mimicking chaos. This hints that the use of artificial neural networks following the super-Turing BPP/log* computation model is a path forward to better alignment with biological neural network operation.

This paper demonstrates how noise aids digital RNNs in attaining super-Turing operation in the realm of the BPP/log* computation class, similar to analog RNNs. We investigate moving limited-precision systems (digital RNNs) from not being chaotic at small amounts of noise, through consistency with chaos, to overwhelming it at large amounts of noise. We also extend a hierarchy theorem to quantify the computational

[*]Corresponding author: emmettredd@missouristate.edu

power of stochastic digital RNNs, which we hypothesize might be super-Turing and lies between P and BPP/log*. The noise source and its quality may impact the super-Turing properties of noise-enhanced digital RNNs. The empirical analyses investigate both pseudonoise and true-random noise in the digital RNN calculations to determine if there are any resulting differences.

The significant contributions of this paper are as follows.

(1) We demonstrate that the BPP/log* computation complexity proof [17–19] is not only a mathematical idealization but also fits into accepted physical SR theory [11,20].

(2) We extend and apply theorems from Balcázar *et al.* concerning an infinite hierarchy between P and P/poly [21] to show an infinite hierarchy exists between P and BPP/log*.

(3) We investigate how different noise magnitudes optimize chaos mimicking calculations in digital RNNs. Mimicking chaos infers noise-enhanced digital RNNs achieve super-Turing operation.

The outline of the paper is as follows. We discuss the details of SR and super-Turing complexity in Sec. II. In Sec. III, we describe an infinite hierarchy of computational complexity classes, to provide a theoretical context for the experimental results, given in Sec. IV, and conclude in Sec. V.

II. STOCHASTIC RESONANCE AND SUPER-TURING COMPLEXITY

This section reviews the preliminaries of SR and its connection with super-Turing concepts to provide a context for the analysis of the computational complexity of RNNs. The complexity classes considered in this paper are described as follows. P denotes the class of functions that compute in time comparable to a polynomial of the input size. Turing machines calculate at this complexity level. P/poly is the class of functions that compute in polynomial time augmented by some additional reference data (called an “advice string” or an “oracle”) of size polynomially related to the input length. Strictly between these two is the class at the heart of our paper, BPP/log*, which computes in bounded probabilistic polynomial time with an advice string of length on the order of the logarithm of the size of the input.

Siegelmann [19] describes multiple variants of analog RNNs that yield different mathematical complexity classes. This paper specifically discusses the analog RNN variant which computes at the BPP/log* complexity class. It has neurons constrained to rational inputs and outputs that are both stochastic. It is super-Turing and has a finite number of physical computations. We postulate that the super-Turing computational complexity class (BPP/log*) matches the physical reality of analog RNNs. This section describes the concept of stochasticity and demonstrates how it is integrated in the super-Turing complexity proof, and subsequently how it matches with the physical model of SR.

A. Stochastic resonance background

Stochastic resonance is a physical phenomenon first described to explain the transitions between ice ages and interglacials [22]. It has been applied to or used in many systems including analog-to-digital converters (ADCs), neuron operation, and visual perception [20]. SR is a well-understood phenomenon that increases precision and resolution anywhere there is a threshold. The SR theory shows that the magnitude of input noise can result in optimizing the signal-to-noise ratio of an SR benefiting system [20]. SR consists of a bistable system with two inputs: a coherent signal and random noise. Using only random noise, for example, consider this simple SR-benefiting system: a model designed to increase the resolution of ADCs. This model assumes adjacent ADC levels which are the two stable states, $\{0,1\}$, with a signal, $a \in [0, 1]$. Given that there exists a transition level at $t = \frac{1}{2}$, if $a \geqslant t$ (or, otherwise $a < t$) then multiple measurements of a will read 1 (0). However, if the signal has uniform noise such that its average value is still a, then multiple measurements of it will read 1 for about (a) portion of the measurements and zero for about $(1 - a)$ portion of them. Averaging those measurements will give a higher-precision approximation to the actual value of a. The precision will increase as the $\log_2$ of the number of measurements. Although each system has an optimum noise level, there is no limit to the logarithmic precision or resolution increase. Note that this SR model, with only the noise component, illustrates the power of stochasticity.

McNamara and Wisenfeld [11] developed a generalized model to explain the theoretical and experimental behavior of SR, including both the noise and periodic forcing components. They introduce a double-well system that is bistable and has a quartic potential of

$$U(x,t) = -\frac{a}{2}x^2 + \frac{b}{4}x^4 - \epsilon x \cos(\omega_s t) \qquad (1)$$

where ϵ and ω_s are the amplitude and frequency of a modulating signal that aids the system transition between the two wells [our Eq. (1) is Eq. (5.1) in [11]]. It models the output's signal-to-noise ratio as a function of the rate at which noise induces hopping between the two states. Harmer *et al.* [20] illustrate this potential over one cycle of the modulating signal in Fig. 1.

This generalized SR model uses a rate equation approach. A first-order differential equation [Eq. (2)] relates the change in probability of a particle being in one of the two states to the rate at which particles flow in and out of the state. The SR dynamical equation from [20] is

$$\frac{dx}{dt} = -\frac{dU(x)}{dx} + A\sin(\omega_s t) + \sqrt{D}\xi(t) \qquad (2)$$

where A is the $-\epsilon$ of Eq. (1), $\xi(t)$ is the noise, and D is a noise magnitude parameter. The phase change in the periodic signal is of no consequence.

This physical SR theory is important as it is the bridge in understanding the connection between super-Turing complexity classes and RNNs. The next section briefly describes the mathematical super-Turing theory so we can subsequently show how it invokes a process congruent with SR.

B. Super-Turing complexity proof

It has been shown that rational stochastic networks exhibit a super-Turing complexity, as modeled by probabilistic TMs that use binary coins with real probabilities [19]. These TMs compute the class BPP/log*, bounded-error probabilistic

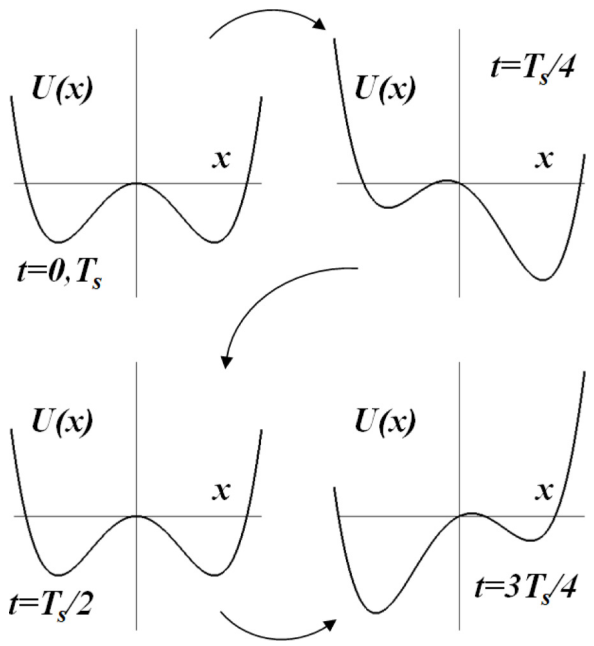

FIG. 1. The double-well potential tilting back and forth over one cycle of the modulating signal where $T_s = 2\pi/\omega_s$. This is adapted from Fig. 4 in [20].

polynomial-time, augmented with logarithmic prefix advice. Here, prefix logarithmic advice means an additional string of length $\log n$ given to the TM to help it process any input of length n or less. To provide some context for this paper, the following summarizes the theoretical proof of the complexity of rational stochastic networks. The underlying concept is that analog RNNs with rational neurons and stochasticity are able to implement a computational model that is stronger than one without stochasticity. However, stochasticity does not increase the computational power of analog RNNs with integer or real neurons. The proofs use a Markovian model, which allows for real-valued, evolving, and neighbor-dependent malfunction probabilities.

In a probabilistic TM, real probabilities and rational probabilities with log prefix advice are equivalent. Let $\mathrm{BP}_R(T)$ denote the class of languages recognized by a probabilistic bounded error TM that uses real probabilities and computes in time T, i.e., a real numbered probabilistic TM. $\mathrm{BP}_Q(T)/\log *$ denotes the class for TMs based on rational probabilities along with a logarithmic prefix advice. Lemma 1 [17–19] states that these two classes are subsets of each other to a polynomial degree.

Lemma 1. The classes $\mathrm{BP}_R(T)$ and $\mathrm{BP}_Q(T)/\log *$ are polynomially related.

This lemma demonstrates that for stochastic networks "logarithmic precision suffices" for the real probabilities; i.e., for up to the qth step of the computation, only the first $O(\log q)$ bits in the probabilities of the neurons influence the result. We sketch the proof of Lemma 1 below, but the interested reader can find details in [17–19]. The claim will follow by demonstrating these two inclusions: (i) $\mathrm{BP}_R(T) \subseteq \mathrm{BP}_Q[O(T\log T)]/\log *$ and (ii) $\mathrm{BP}_Q(T)/\log * \subseteq \mathrm{BP}_R[O(T^2)]$.

To establish (i), the idea is to show that given any probability of error $\epsilon > 0$, one can model the behavior of any machine in $\mathrm{BP}_R(T)$ by a machine in $\mathrm{BP}_Q[O(T\log T)]$, with error no more than ϵ. In this setting, the latter machine has rational neurons instead of real neurons of the first machine, but more time to make its decision, as well as an advice string that is as long as a constant multiple of the logarithm of the input.

This shows that a machine using rational probability (here a fair coin with probability of $\frac{1}{2}$) and a logarithmically long advice string can simulate a machine having real probabilities. Note that SR is not involved in this simulation, since the simulating machine's probability is rational and could be found in a finite time. The power of SR ultimately shines in proving the second inclusion.

To prove (ii), observe that given any error probability $\epsilon > 0$, a machine $\mathcal{M} \in \mathrm{BP}_Q(T)/\log *$ can be simulated by a machine $\mathcal{M}' \in \mathrm{BP}_R[O(T^2)]$. This is essentially the converse of (i), but with some different runtime bounds for the machines in question. The main idea here is that, by flipping an unfair coin, $\mathcal{M}'$ can generate the logarithmic advice string used by $\mathcal{M}$. The trick is to encode the advice string in a probability represented by binary number $p \in [\frac{1}{4}, \frac{1}{2}]$. The range of probability is a technical artifact of the proof. From the $z = O(T^2)$ coin flips, the estimation of the probability is $\tilde{p}$, which is "01" followed by the advice string. This estimation is a logarithmically long string and can be found to an arbitrarily high precision by

$$\tilde{p} = \frac{\#1}{z} = \frac{\sum_{i=1}^{z} f_i}{z}, \tag{3}$$

where #1 is the number of ones found in the z flips and f_i is the value of the ith flip. This is the typical way that measurement systems take advantage of SR, i.e., taking the average of z analog-to-digital conversions. At the end of this process, by flipping the unfair coin no more than $z = O(T^2)$ times, $\tilde{p}$ is an approximate value of p, up to some acceptable error. The initial "01" is removed from the binary representation of $\tilde{p}$, and the remaining binary string is used as advice. There is more to the proof and to ensuring that the error bounds line up correctly. The features highlighted here demonstrate the use of SR to relate the models with real and rational neurons.

This proof that rational stochastic networks compute the super-Turing BPP/log* complexity class depends on flipping an unfair coin. The number of flips are finite and produce an estimate of the real probability which is a logarithmically long advice string. Note that the estimated advice can only equal the uncomputable real probability in the infinite limit. Nonetheless, the finite estimated advice is physically possible and the rational, stochastic neuron analog RNN can compute at a super-Turing complexity of BPP/log*. The next section will show that this unfair coin can be physically modeled as a SR process by mapping it into a double-well system. Hence, the BPP/log* complexity class can serve as a mathematical basis for understanding the relevance of SR to physical super-Turing computation.

C. Stochastic resonance and unfair coin flips

Lemma 1 estimates the logarithmic advice through a process based on flipping an unfair coin. The idealized unfair coin

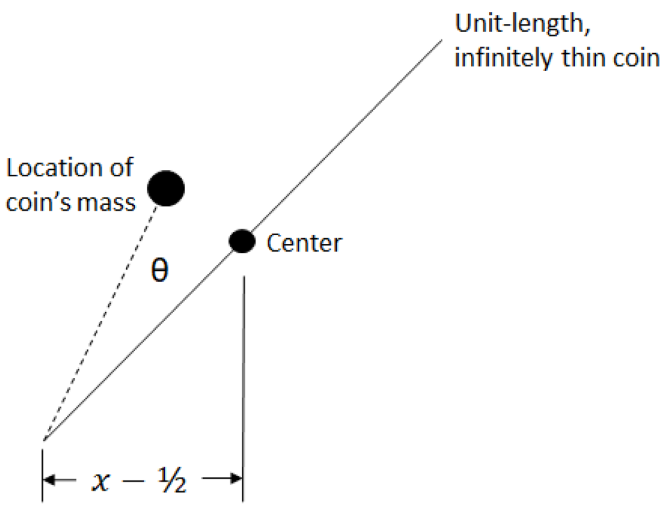

FIG. 2. Illustration of the two-dimensional unfair coin model used to develop the potential-energy function $U(x)$. The coin's mass is separated from the center of the coin by $\theta \in (-\pi/2, \pi/2)$. The corresponding unfair probability ranges from 1 to 0. The x coordinate is chosen so when the coin falls to the right (left) we have $x = 1$ ($x = 0$).

can be modeled as a mass that has unity weight at an angle θ away from the coin's axis (see Fig. 2). The model results in a double-well potential $U(x)$ for use in Eq. (2) (see Sec. II A) to demonstrate the SR phenomenon. The potential $U(x)$ is a periodic function with each cycle of length 2. It is described over one cycle as follows:

$$U(x) = \begin{cases} \sin(\arccos(2x+1) - \theta)/2, & -\frac{1}{2} \leqslant x < 0 \\ \sin(\arccos(2x-1) + \theta)/2, & 0 \leqslant x < 1 \\ \sin(\arccos(3-2x) + \theta)/2, & 1 \leqslant x < \frac{3}{2} \end{cases}. \quad (4)$$

The probability of the unfair coin is $p = [1 - \sin(\theta)]/2$. The second condition denotes when the left end of the coin in Fig. 2 is the fixed rotation point at $x = 1/2$. The first and third conditions are when the upper end of the coin becomes a fixed rotation point at $x = -1/2$ and 1.5, respectively (i.e., the coin flipping up after the coin falls to horizontal on the left or right).

To physically model the flipping of the unfair coin as an SR process, we derive a numerical solution of the SR dynamical equation Eq. (2) using $U(x)$ [Eq. (4)] for two coin probability values ($p = 1/4$ and $1/2$). Recall that the super-Turing complexity proof of the prior section (see Sec. II B) assumes coin probabilities $1/4 \leqslant p \leqslant 1/2$. Hence, we illustrate the coin flip potentials here for the probability limits, though the model presented extends to all discrete values within the range. As can be observed from the optimized solution (Fig. 3), it does reflect a bistable potential that yields the classic SR picture of bouncing between the two levels.

The numerical simulation of Eq. (2) is obtained by utilizing the following parameter values: $dt = 10^{-3}$, $\omega_s = 2\pi/10$, and $\xi(t)$ is a vector of 1×10^6 elements chosen from $\{-1, 1\}$, given by MATLAB's `rand` function. Note that in [20], A was set so the particle remains in one well and D adjusted while measuring the signal-to-noise level. Here, the values of the tuning parameters A and D are optimized specifically for each given coin probability value (Fig. 4). The large slopes near the minima of the potential cause stability issues with the numerical simulation. Therefore, the absolute value of the slope is limited to a maximum of 2.

Figure 4 illustrates the solutions obtained for coin probabilities of 1/4 and 1/2 and verifies that unfair coin models exhibit stochastic resonance, i.e., two stable states with quick, random transitions between them. However, these solutions are more akin to putting a coin in a box that is continuously tilted and shaken and are not how coin flips occur. Rather, we stop tilting and shaking the box and then observe the results of a coin flip. This is simulated by randomly selecting a sampling point in the solution and recording in which potential well the point resides, i.e., where the coin will settle in the double wells when the tilting and shaking stop. The SR calculations shown in Fig. 4 were sampled 262 144 (2^{18}) times. The left (right) SR calculation, is optimized (using MATLAB `fminsearch`) to have a probability of 1/4 (1/2), and has a *sampled* probability, $\tilde{p}$, of 0.2503 (0.4990). A different random selection could give a slightly different result as one would expect from the statistical process of flipping a coin. Stripping off the "01" of $\tilde{p}$ gives the logarithmic advice from this sampled probability as hexadecimal `0x0055` (`0xfeed`). These sampled probabilities are calculated using Eq. (3), which is the source of the prefix logarithmic advice in Lemma 1. Hence, this demonstrates that

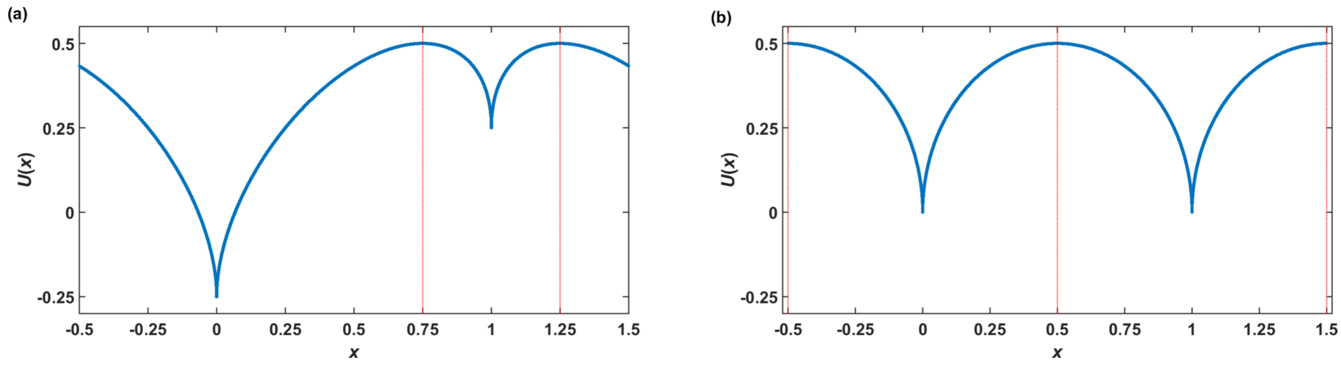

FIG. 3. Potentials for two coins. Shown are the points of the potentials $U(x)$ in Eq. (4) accessed during the stochastic resonance calculations shown in Fig. 4. The left is for a coin with $p = 1/4$, and the right is for a coin with $p = 1/2$. The different well depths for the left potential result from whether the angular offset mass is below or above the coin's center as it goes to horizontal.

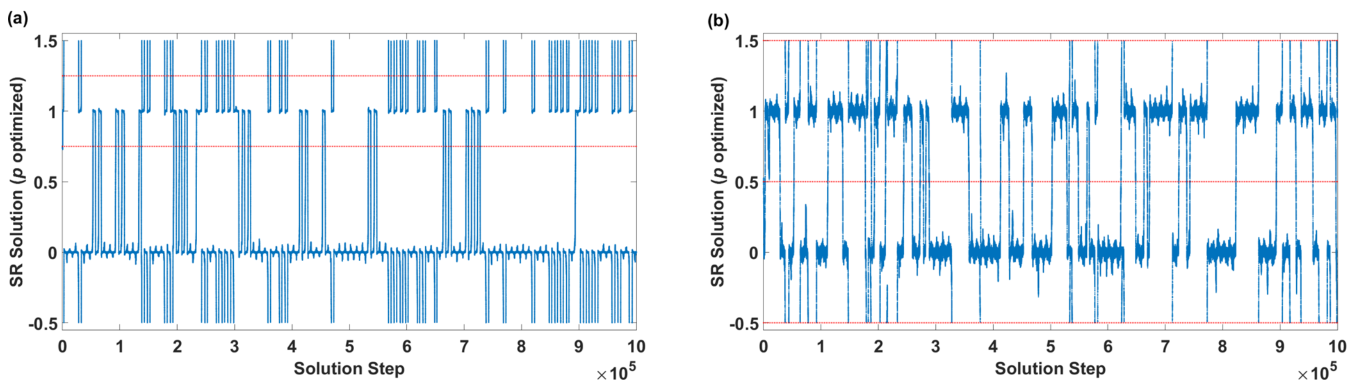

FIG. 4. Stochastic resonance for two coin potentials, showing the horizontal position, x, of the coin center from solving Eq. (2) [20] for 10^6 steps. Each step has the x dot plotted along the vertical dimension. The values of A and D were optimized with the MATLAB `fminsearch` function so the proportion of points between the red dotted lines on either side of 1 equaled the probability of the coin. The red dotted lines are at the maxima of the corresponding potentials in Fig. 3. Solutions optimized for $p = 1/4$ are on the left and those for $p = 1/2$ are on the right.

SR indeed provides the prefix logarithmic advice essential for the super-Turing complexity class.

III. DIGITAL RNNs AND SUPER-TURING COMPLEXITY CLASSES

Digital computers are modeled on the Turing machine and share this characteristic with the super-Turing model, as discussed above; they compute using rational numbers. However, they eschew noise by their basic design and even have error-correcting hardware and software to make sure that noise-induced bit flips are corrected. In this section, guided by the super-Turing model (Sec. II), we lay out the theoretical foundation for adding stochasticity to rational, deterministic, digital RNNs operating at the Turing complexity level so they may exceed the Turing computation limit.

A. Deterministic vs stochastic computations in recurrent neural networks

Table I summarizes the computational power of analog RNNs that have various restrictions applied to them [19]. It also points out the physically possible computational classes. The stochastic-signal, rational-RNN computation class, BPP/log*, is the only super-Turing class in the table that has computations allowed by the physical universe in a finite time.

TABLE I. Complexity classes of recurrent neural networks. This table was adapted from Table 9.1 in [19]. See first paragraph of Sec. II for complexity class descriptions.

Weights/signals	Deterministic	Stochastic	Notes
Integer ($\mathbb{Z}$)	Regular	Regular	Physical
Rational ($\mathbb{Q}$)	P	BPP/log*	Physical
Real ($\mathbb{R}$)	P/poly	P/poly	Nonphysical[a]

[a]Quantum field theory quantizes the free electromagnetic field [23] and that does not allow weights or signals to have real numbered values.

Biological and computer-calculated artificial neural networks physically compute under the same restrictions as those leading to the BPP/log* super-Turing computation class.

Consider a RNN $\mathcal{R}$ composed of a matrix of input weights W_x, a parametric state transition model $\mathcal{S}$, a set of output weights W_y, and an output bias b_y. The network $\mathcal{R}$ operates on an input sequence $\mathbf{x}$ to produce an output sequence $\mathbf{y}$. Deterministic computation (second column of Table I) implies no noise and complete predictability for $\mathbf{x}$. Thus, at each time step, $\mathbf{x}_t$ maps deterministically to each $\mathbf{y}_t$. Stochastic computation (third column of Table I) implies that the $\mathcal{R}$, with a certain probability, unreliably computes $\mathbf{y}_t$ from a given $\mathbf{x}_t$. This means $\mathbf{y}$ is indeterminate in that it involves a level of randomness.

The rows in Table I depict the varying levels of restrictions imposed on the analog RNNs by limiting the synaptic weights W. At the most restrictive level of W, when the weights are only integers, we observe that stochasticity does not increase its computational power over deterministic computation. Though these networks have calculations consistent with our physical understanding of the known universe, they are less powerful than TMs. They are simply automatons. The case where W can also contain rational numbers results in deterministic computations having TM computation power (P) while stochastic computation computes at a super-Turing computation power, BPP/log*. This super-Turing complexity class has physically feasible calculations. In the final and least restrictive case, when W includes irrational numbers (so that any real number is possible), deterministic and stochastic computation have the same computational power. In addition, since real numbered neuron weights are not possible due to charges and fields being quantized [23], this highest complexity class, i.e., P/poly (a super-Turing class), is not accessible.

The digital RNNs that will be investigated in our experiments (Sec. IV) have only one injected-noise neuron whereas the analog RNN may have several stochastic neurons. Both RNNs have deterministic neurons. The difference between the P and BPP/log* complexity levels in Table I is movement from deterministic to stochastic signals, as simulated in the experiments by adding various magnitudes of uniform random numbers to the recurrent signals. The postulate is that digital

RNNs seem to attain, with the aid of SR noise, a super-Turing level of operation on a complexity hierarchy similar to that between P and P/poly in [21]. Somewhere in such a hierarchy, a transition from Turing to super-Turing complexity must occur. Recognizing that the noise added in our digital RNNs is different from the stochasticity of [19], the hierarchy offers a possibility that the noise-enhanced digital RNNs could operate at a super-Turing level less complex than BPP/log*.

B. Infinite hierarchy of computational complexity classes between P and BPP/log*

This section proves that an infinite hierarchy of computational classes exists between P and BPP/log* (a super-Turing class). One of the classes in the hierarchy could quantify the computational power of noise-enabled digital RNNs. The existence of these computational classes is demonstrated with Kolmogorov complexity (KC), which measures the algorithmic information content of an object [24]. It has been shown that the computational power [21] of RNNs depends on the complexity of the neurons in the network. In contrast, this proof shows that rational stochastic networks are more computationally complex than deterministic ones.

Let x be a finite binary string. The Kolmogorov complexity of x is the length of the shortest computer program (in binary) that generates x. This definition assumes unlimited computational resources such as time and space. Note that KC does not completely characterize computational complexity but it can be used to discriminate between computational complexity classes. KC has been shown to be a much more reliable and robust basis for constructing the complexity measure for compound objects such as NNs [25]. In the context of RNNs, a resource-bounded version of KC is utilized as it constrains the amount of information and the time used by the algorithm.

Balcázar *et al.* [21] demonstrated the existence of an infinite hierarchy of computational classes between P (the complexity class of TMs) and P/poly (the complexity class of networks with real number neurons). We follow their construction to show there exists an infinite hierarchy between P and BPP/log* (complexity class of rational stochastic networks). The heart of their argument, which we state below, is a result of partial orderings of function classes. These partial orderings emerge since the KC of the neurons of the NN relates to a structural notion: the amount of advice for nonuniform classes.

In order to state their result (Theorem 7.1 in [21]), we record some definitions. Let $\mathcal{F}$ and $\mathcal{G}$ be nonempty function classes. We say that $\mathcal{F} \prec \mathcal{G}$ if there is some nondecreasing $s(n) \in \mathcal{G}$, computable in time polynomial in n, so that the following hold:

(i) $s(n) = o(n)$.

(ii) For any $r \in \mathcal{F}$, and any polynomial p, we have that $r[p(n)] = o[s(n)]$.

Note that by definition, any such function class $\mathcal{F}$ will be sublinear.

We adopt the following notation for resource-bounded KC: $K[\mathcal{F}, \mathcal{P}]$, where $\mathcal{F}$ denotes the length of advice information needed, and $\mathcal{P}$, the running time. If there is no bound on the running time, it is simply written $K[\mathcal{F}]$.

Given some ordered list of programs, a tally set $\mathcal{T}$ can be thought of as a subset of these programs. We denote the characteristic sequence $\chi_{\mathcal{T}}$ as a binary string, the ith bit of which is a 1 if the ith program can be accepted, and a zero if not. Finally, $\mathcal{P}(\mathcal{T}_{\mathcal{F}})$ denotes the family of all tally sets $\mathcal{T}$ with the property that $\chi_{\mathcal{T}} \in K[\mathcal{F}, \text{poly}]$, i.e., the class of all sets decidable in polynomial time with a polynomially long advice string. This class coincides with the P/poly computational class of deterministic and stochastic RNNs with real neurons (Table I). With these definitions in tow, we state the partial ordering theorem of Balcázar *et al.* [21].

Theorem 1 (Theorem 7.1 in [21]). Let $\mathcal{F}$ and $\mathcal{G}$ be nonempty function classes, such that $\mathcal{F} \prec \mathcal{G}$. Then, $\mathcal{P}(\mathcal{T}_{\mathcal{F}})$ is properly included in $\mathcal{P}(\mathcal{T}_{\mathcal{G}})$.

Note that in Theorem 1, there is nothing special about P classes. According to [21], "similar separation results can be proved for other well-behaved families of run-time functions." Given any natural number m, we define $\log^{(m)}$ as log composed with itself m times. We can easily see that for any natural number m, we have that $\log^{(m+1)} \prec \log^{(m)}$. Following their proof of Theorem 1 but applying the argument to classes of the form $\text{BPP}/\log^{(m)}*$ yields the following variant of the theorem.

Theorem 2. Given a natural number m, we have that $\text{BPP}/\log^{(m+1)}* \subsetneq \text{BPP}/\log^{(m)}*$.

The following lemma allows us to relate classes of the form $\text{BPP}/\log^{(m)}*$ to the computational complexity class of TMs.

Lemma 2. For any natural number m, we have that $P \subsetneq \text{BPP}/\log^{(m)}*$.

To see this, we again adapt the argument in the proof of Theorem 1 with $\mathcal{F}$ being the class of constant functions, denoted $O(1)$ and $\mathcal{G} = \log^{(m)}$. This gives us that $\text{BPP}/O(1)*$ is properly included in $\text{BPP}/\log^{(m)}*$. But $\text{BPP}/O(1)*$ is the class of BPP with constant length prefix advice, but any constant length advice (with or without a prefix constraint) can be coded into any program. Therefore, we have that $\text{BPP}/O(1)*$ coincides with the class BPP. It is known that P$\subseteq$ BPP (although it is not known if this inclusion is proper, what is known will suffice for this argument). Combining these facts, we get the following chain of set inclusions, which completes the proof of the lemma:

$$\text{P} \subseteq \text{BPP} = \text{BPP}/O(1)* \subsetneq \text{BPP}/\log^{(m)}*.$$

The combination of Theorem 2 and Lemma 2 proves the following claim.

Theorem 3. There is an infinite hierarchy of computational complexity classes strictly between P and $\text{BPP}/\log^{(m)}*$.

Given that there are such computational complexity classes exhibiting super-Turing computational power, but not attaining the full scope of a stochastic neural net with rational neurons and real probabilities (BPP/log*), we propose that noise-enabled digital RNNs have power falling somewhere in between these two extremes. Thus, noise-enabled digital RNNs may have super-Turing computational power, while not yet reaching the full power of BPP/log*.

IV. NOISE CONTRIBUTIONS TO SUPER-TURING COMPUTATION

This section empirically explores the relative magnitude of noise, which is SR without the periodic forcing component, that aids the movement of digital RNNs from Turing to super-Turing complexity. SR was critical to proving that rational, stochastic RNNs were super-Turing (Sec. II). As shown in [26], various magnitudes of noise can optimize the signal-to-noise ratio provided by SR. Since digital RNNs already operate using rational numbers, determining whether they can be made super-Turing can be empirically evaluated simply by adding noise. The experiments are motivated by previous work on analog RNNs [16] in which an OpticARNN was demonstrated to be consistent with mimicking chaos and super-Turing operation. However, it may have fallen short of being a fully BPP/log* analog RNN since its noise components had an unknown relationship to the sparsely discrete stochasticity of the computational complexity proof [19]. This could place it somewhere in the hierarchy given by Theorem 3, strictly between P and BPP/log*. These inconsistencies with the BPP/log* assumptions and realization of the importance of noise prompted this paper's investigation of whether pseudorandom or true-random number generators could be used in digital RNNs to cause them to mimic chaos.

The only known method of indicating super-Turing operation is mimicking chaos. The main argument against Turing machines (or anything with that computational power) being able to mimic chaos results from the *limited precision of representing chaotic formula parameters* [19,27]. However, the floating-point calculations on many (TM-inspired) digital computers make it very time consuming to observe their failure to mimic chaos. On the other hand, *limiting the precision of the floating-point calculation results that are recurred back* to the input(s) of RNNs implemented on digital computers allows timely observation of their failure to mimic chaos. The experiments here use neural networks developed and trained in MATLAB's Neural Network package. To make the chaotically trained digital RNNs have reasonably short repeat cycles, the experiments limit the precision of the floating-point NN results that are recurred back to the inputs of the digital RNNs. These experiments can then elucidate the effect of adding noise in developing super-Turing operation.

Noise applied to chaotic systems has had diverse results. Noise in some systems induced chaos [28], while in others it induced order [29]. Minimization of the largest Lyapunov exponent was observed in coupled, heterogeneous (disordered-length) pendulums for three types of disorder in [30]. Positive Lyapunov exponents are used in [31] to indicate that a system is acting chaotically. Lyapunov exponent values [32] quantify the sensitive dependence on initial conditions which is an inherent property of a chaotic system. Chaotic systems also lack repeat cycles. Repeat cycles can be observed in the time series or by autocorrelation peaks in their transformed time series. A time series must pass both Lyapunov exponent and no-repeat-cycle (or no-autocorrelation-peaks) tests to provide sufficient evidence for consistency with chaos. In general, both methods often agree in indication of chaos as the more positive the Lyapunov exponent, the smaller the autocorrelation peaks. Since Lyapunov exponents are quantitative while observations of the repeat cycles and autocorrelation peaks are qualitative, the simulation results are reported using the former. For varying magnitudes of noise, the Lyapunov exponents are computed to determine consistency with chaos.

A. Empirical evaluation of noise-enhanced digital RNNs

To investigate whether noise added to digital RNNs would simulate super-Turing operation, we utilize pseudorandom and true-random number sequences as the simulated noise. Both were chosen so the experiments had the possibility of determining any difference between the different kinds of random-number noise sequences. To generate experimental statistics, each kind of noise had ten independent sequences. The experimental time series of the logistic and Hénon networks are generated using Algorithm 1 for both sequences. The feed-forward portions of the RNNs are trained with the Levenberg-Marquart back propagation option of MATLAB's Neural Network package. The output neuron for each network uses a linear activation function while the hidden-layer neuron activation functions for the logistic and Hénon networks are log-sigmoid and hyperbolic tangent sigmoid, respectively. These feed-forward neural networks (FFNNs) are used in Algorithm 1 to build the RNNs that have limited-precision, noisy recurred signals. The ten independent sequences for each noise type are uniform in the range $[-1, 1]$. To generate the data for Figs. 5 and 6, these sequences are multiplied by powers of 2 (see x axes of the figures). Algorithm 1 adds this scaled noise to the output of the FFNN and limits the precision of that sum before it is recurred back to the input,

Algorithm 1 Limited-Precision Time Series Generator for Noise-Enhanced Digital RNN.

```
Given $p$ = precision; $\xi = k \times n$ noise matrix; $s$ = seed;
$k$: # of independent time series; $n$: length of time
  series;
$m$: # of inputs for RNN;
($m = 1, p = 18 \rightarrow$ logistic; $m = 2, p = 11 \rightarrow$ Hénon)

Initialize
    $t = 1$
    $x \leftarrow$ LimitPrecision $(s, p)$
        $\triangleright$ all values of $x$ initially same
    $l_P = 2^{p+1}$
    $x(:, t) \leftarrow$LimitPrecision $((x(:, t) + \xi(:, t)/l_P), p)$
End

for $t = 2 : n$ do
    $x(:, t) \leftarrow$LimitPrecision $((x(:, t) + \xi(:, t)/l_P), p)$
    for $j = 1 : k$ do
        $x(j, t + 1) =$ FFNN $(x(j, (t - m + 1) : t))$
            $\triangleright$ FFNN (feed-forward neural network)
    end
end
Return $x$

LimitPrecision$(a, p)$
    $a_P = (round(a \cdot 2^p))/2^p$
Return $a_P$
```

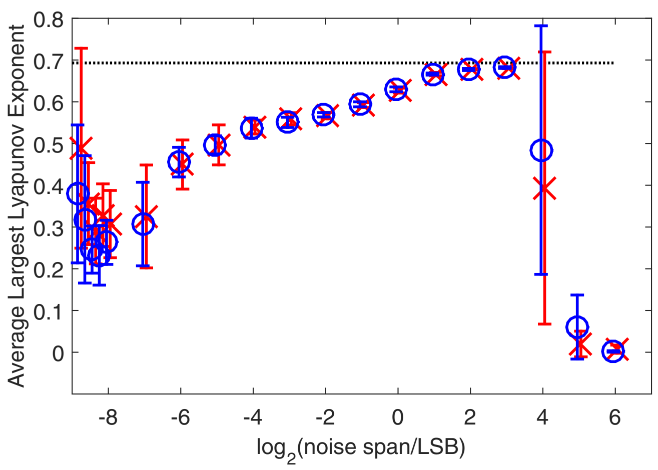

FIG. 5. Largest Lyapunov exponent of a digital recurrent neural network trained with a logistic map series for various noise levels. The neural network had one input, seven hidden nodes, and one output node. Each output of the calculated series was limited to 18 bits of precision before it was recurred back to the input. The x coordinates are actually integers, but the plotted points and error bars are offset by ± 0.05 for the true random noise [33] (red $\times$) and for the pseudorandom noise from MATLAB's `rand()` function (blue o), respectively. The dotted line (black) represents the accepted value for the logistic map [32].

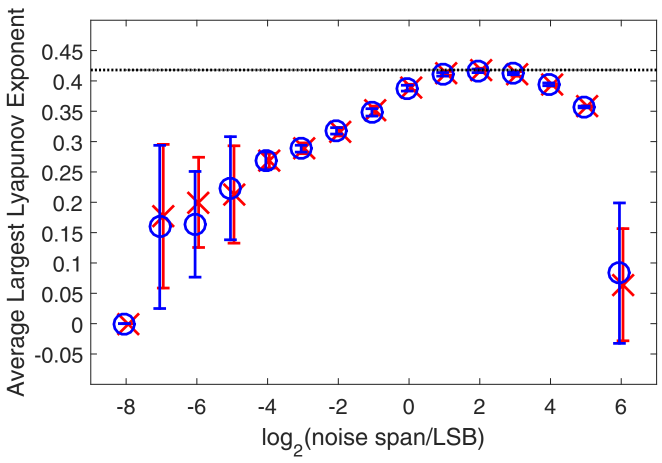

FIG. 6. Largest Lyapunov exponent of a digital recurrent neural network trained with a Hénon map series for various noise levels. The neural network had two inputs, 15 hidden nodes, and one output node. Each output of the calculated series was limited to 11 bits of precision before it was recurred back to one input while the previous precision-limited output plus noise was saved for the other input. The x coordinates are actually integers, but the plotted points and error bars are offset by ± 0.05 for the true random noise [33] (red $\times$) and for the pseudorandom noise from MATLAB's `rand()` function (blue o), respectively. The dotted line (black) represents the accepted value [32].

thereby completing the RNN. When the power of 2 is zero, the uniform noise is distributed over [−LSB/2, LSB/2] of the precision-limited recurred sum, where LSB denotes least-significant bit.

Most of the results obtained for both noise-enhanced digital RNN experiments demonstrated consistency with chaos at certain magnitudes of noise. Figures 5 and 6 show the results of the computed values of Lyapunov exponents (mean and standard deviations) for various noise levels for the logistic and Hénon RNNs, respectively. As noted earlier in Sec. IV, noise has diverse effects in different experiments. For the logistic experiments, there were two noise sequences in the true-random case (one in the pseudorandom case) that returned positive Lyapunov exponents near the accepted value for very low noise magnitudes. Inspecting those time series revealed that they had a short run (about 7%) of a chaotic-looking area embedded in a repeating time series. Although the Lyapunov calculation found a positive exponent in the series, the time-series diagrams had large portions with repeat cycles. Since a prospective chaotic time series has to pass both tests showing consistency with chaos, those three noise sequences were removed from the results shown in Fig. 5.

From the results illustrated in Figs. 5 and 6, we can observe that up to a point increasing the noise added to the recurrent signals causes limited-precision digital simulations to better mimic chaos. This is indicated by the calculated Lyapunov exponents approaching the expected values for the logistic and Hénon maps [32]. The added noise maximizes the Lyapunov exponent and the full restoration of chaos when the noise is eight times the size of the LSB of the logistic RNN, and four times LSB for the Hénon RNN. These are in a range near that of a theoretical optimum for removing quantization noise using a single LSB dither signal [34]. The figures also indicate that there is no significant difference between pseudorandom and true-random noise. This is not unexpected as no other tests for randomness would show a difference between these types of noise over the sequence lengths used in these experiments.

These precision-limited digital RNNs empirically approach some of the physical properties of the analog RNNs that compute at the BPP/log* complexity level: stochastic signals and rational neurons. The experiments had deterministic signals within the feed-forward portion of the network while noise was injected into the recurrent signals via random number sequences. They mimicked chaos in calculating multiple, but limited-length, time series with positive Lyapunov exponents. The complexity class BPP/log* is an appropriate super-Turing class to explore physical experiments, as its assumption of rational synaptic neurons in the RNN matches the quantized nature of the universe. The construction of the logarithmic advice (Sec. II) is identical to the physical SR. Noise, inherent in SR and stochastic signals in the RNN, gives rise to the super-Turing complexity class, BPP/log*. The RNN experiments demonstrate operation consistent with mimicking chaos and, thereby, consistency with operating at a super-Turing computation level. The BPP/log* proof flips a coin n^2 times and only indirectly depends on real probabilities and is, therefore, not dependent on continuous signals. Noise-enhanced digital simulations can be influenced by an uncountably infinite space like BPP/log* can, even if they do not completely reach that complexity level. The infinite hierarchy between P and BPP/log* established in Sec. III B provides a place for noise-enhanced digital RNNs to be super-Turing.

V. CONCLUSION

This paper demonstrates that a super-Turing mathematical computational complexity class (BPP/log*) matches the physical reality of recurrent neural networks. Generally accepted physical models of the universe supply rational neurons through quantized charge and stochastic signals via nonzero thermal fluctuations and interference from other signals. The random unfair coin flips in the super-Turing proof provide an oracle to a probabilistic Turing machine operation that leads to super-Turing computation. The oracle results from a modeled unfair coin and exhibits physical stochastic resonance. So, the rational, stochastic, analog RNN matches physical reality and is a good model to follow in attempting to achieve super-Turing computation.

It has been shown, both theoretically and in practice, that the addition of stochastic resonance (usually in the form of random noise) to various computation systems can be beneficial, if done in a principled manner. We have simulated an array of calculations done by machines with limited precision for computation yet augmented by random processes. The simulations show measurably greater consistency with chaotic computations than had they not been augmented by stochastic resonance. The experiments of this paper may be more limited than the rational, stochastic analog RNN since the noise has different character than the stochastic neuron disturbances. This paper shows that an infinite series of strictly increasing computational complexity classes exists between Turing machine complexity and bounded error probabilistic Turing machine complexity having prefix logarithmic advice. This provides theoretical support and justification that the experiments could attain super-Turing operation.

The empirical evaluations show that optimum noise levels cause limited-precision digital RNNs to match the Lyapunov exponents of their respective chaotic systems, provided there are no repeat cycles in their series. These noise-added digital RNNs mimicking chaos indicate super-Turing operation. The optima occurred at a magnitude of uniform noise about eight times LSB for the logistic time-series trained digital RNN, and about four times LSB for Hénon. This is comparable to the theoretical optimum noise level to mitigate quantization noise in analog-to-digital conversion. No difference was detected between noise provided by pseudorandom and true-random number sequences, but the shortness of the pseudorandom number sequences made them indistinguishable from true-random ones. Future simulations will explore detectable pseudorandom numbers (as opposed to true random) added to the digital RNN recurrent signals.

[1] K. Audhkhasi, O. Osoba, and B. Kosko, Noise-enhanced convolutional neural networks, Neural Networks **78**, 15 (2016).

[2] A. Smart, Is noise the key to artificial general intelligence? converging evidence indicates noise plays a fundamental role in the brain, https://www.psychologytoday.com/intl/blog/machine-psychology/201606/is-noise-the-key-artificial-general-intelligence (2016).

[3] M. D. McDonnell and D. Abbott, What is stochastic resonance? definitions, misconceptions, debates, and its relevance to biology, PLoS Comput. Biol. **5**, e1000348 (2009).

[4] E. Redd, A. S. Younger, and T. Obafemi-Ajayi, Stochastic resonance enables BPP/log* complexity and universal approximation in analog recurrent neural networks, in *2019 International Joint Conference on Neural Networks* (IEEE, New York, 2019).

[5] O. Adigun and B. Kosko, Using noise to speed up video classification with recurrent backpropagation, in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, New York, 2017), pp. 108–115.

[6] M. Sabri and T. Kurita, Effect of additive noise for multi-layered perceptron with autoencoders, IEICE Transactions on Information and Systems **100**, 1494 (2017).

[7] N. Nagabushan, N. Satish, and S. Raghuram, Effect of injected noise in deep neural networks, in *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (IEEE, New York, 2016), pp. 1–5.

[8] B. Kosko, *Noise* (Penguin, Toronto, 2006).

[9] F. Moss, L. M. Ward, and W. G. Sannita, Stochastic resonance and sensory information processing: a tutorial and review of application, Clin. Neurophysiol. **115**, 267 (2004).

[10] C. M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University, New York, 1995).

[11] B. McNamara and K. Wiesenfeld, Theory of stochastic resonance, Phys. Rev. A **39**, 4854 (1989).

[12] O. Eluyode and D. T. Akomolafe, Comparative study of biological and artificial neural networks, Eur. J. Appl. Eng. Sci. Res. **2**, 36 (2013).

[13] J. Schmidhuber, http://people.idsia.ch/~juergen/ (accessed May 27, 2020).

[14] A. M. Turing, On computable numbers, with an application to the entscheidungsproblem, Proceedings of the London Mathematical Society **2**, 230 (1937).

[15] A. S. Younger, E. Redd, and H. Siegelmann, Development of physical super-turing analog hardware, in *International Conference on Unconventional Computation and Natural Computation* (Springer, New York, 2014), pp. 379–391.

[16] A. S. Younger, E. Redd, H. Siegelmann, and C. Bell, A physical machine based on a super-turing computational model, https://asg.uark.edu/wp-content/uploads/sites/176/2017/04/PC-2017-Physical-Machine.pdf (2017).

[17] H. T. Siegelmann, Neural dynamics with stochasticity, in *Adaptive Processing of Sequences and Data Structures* (Springer, New York, 1998), pp. 346–369.

[18] H. T. Siegelmann, Stochastic analog networks and computational complexity, Journal of Complexity **15**, 451 (1999).

[19] H. T. Siegelmann, *Neural Networks and Analog Computation: Beyond the Turing Limit* (Birkhauser, Boston, 1999).

[20] G. P. Harmer, B. R. Davis, and D. Abbott, A review of stochastic resonance: Circuits and measurement, IEEE Trans. Instrum. Meas. **51**, 299 (2002).

[21] J. L. Balcázar, R. Gavalda, and H. T. Siegelmann, Computational power of neural networks: A characterization in terms of Kolmogorov complexity, IEEE Trans. Inf. Theory **43**, 1175 (1997).

[22] R. Benzi, A. Sutera, and A. Vulpiani, The mechanism of stochastic resonance, J. Phys. A: Math. Gen. **14**, L453 (1981).

[23] M. O. Scully and M. S. Zubairy, *Quantum Optics* (Cambridge University, Cambridge, England, 1997).

[24] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications* (Springer, New York, 2013).

[25] M. Morzy, T. Kajdanowicz, and P. Kazienko, On measuring the complexity of networks: Kolmogorov complexity versus entropy, Complexity **2017**, 3250301 (2017).

[26] B. McNamara, K. Wiesenfeld, and R. Roy, Observation of Stochastic Resonance in a Ring Laser, Phys. Rev. Lett. **60**, 2626 (1988).

[27] H. T. Siegelmann, Computation beyond the Turing limit, Science **268**, 545 (1995).

[28] I. I. Fedchenia, R. Mannella, P. V. E. McClintock, N. D. Stein, and N. G. Stocks, Influence of noise on periodic attractors in the lorenz model: Zero-frequency spectral peaks and chaos, Phys. Rev. A **46**, 1769 (1992).

[29] K. Matsumoto and I. Tsuda, Noise-induced order, J. Stat. Phys. **31**, 87 (1983).

[30] J. F. Lindner, B. S. Prusha, and K. E. Clay, Optimal disorders for taming spatiotemporal chaos, Phys. Lett. A **231**, 164 (1997).

[31] H. Yamazaki, T. Yamada, and S. Kai, Can Stochastic Resonance Lead to Order in Chaos? Phys. Rev. Lett. **81**, 4112 (1998).

[32] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, A practical method for calculating largest Lyapunov exponents from small data sets, Physica D **65**, 117 (1993).

[33] https://www.random.org/ (2019).

[34] L. Schuchman, Dither signals and their effect on quantization noise, IEEE Trans. Commun. Technol. **12**, 162 (1964).