01 Jul 2019

# Stochastic Resonance Enables BPP/log∗ Complexity And Universal Approximation In Analog Recurrent Neural Networks

Emmett Redd

A. Steven Younger

Tayo Obafemi-Ajayi
*Missouri University of Science and Technology*, towd2@mst.edu



## Recommended Citation

E. Redd et al., "Stochastic Resonance Enables BPP/log∗ Complexity And Universal Approximation In Analog Recurrent Neural Networks," *Proceedings of the International Joint Conference on Neural Networks*, article no. 8851775, Institute of Electrical and Electronics Engineers, Jul 2019.
The definitive version is available at https://doi.org/10.1109/IJCNN.2019.8851775

# Stochastic Resonance Enables BPP/log* Complexity and Universal Approximation in Analog Recurrent Neural Networks

Emmett Redd
*Physics, Astronomy, & Materials Science*
*Missouri State University*
Springfield, MO USA
EmmettRedd@missouristate.edu

A. Steven Younger
*Physics, Astronomy, & Materials Science*
*Missouri State University*
Springfield, MO USA
SteveYounger@missouristate.edu

Tayo Obafemi-Ajayi
*Engineering Program*
*Missouri State University*
Springfield, MO USA
TayoObafemiajayi@missouristate.edu

***Abstract*—Stochastic resonance (SR) is a natural process that without limit increases the precision of signal measurements in biological and physical sciences. Most artificial neural networks (NNs) are implemented on digital computers of fixed-precision. A NN accessing universal approximation and a computational complexity class more powerful that of a Turing machine needs analog signals utilizing SR's limitless precision increase. This paper links an analog recurrent (AR) NN theorem, SR, BPP/log* (a physically realizable, super-Turing computation class), and universal approximation so NNs following them can be made computationally more powerful. An optical neural network mimicking chaos indicates super-Turing computation has been achieved. Additional tests are needed which can verify super-Turing computation, show its superiority, and demonstrate its practical benefits. Truly powerful cognitively inspired computation needs to access the combination of ARNNs, SR, super-Turing mathematical complexity, and universal approximation.**

***Index Terms*—neural networks, cognitive theory, stochastic resonance theory, super-Turing theory, universal approximation**

## I. Introduction

According to Smart [1], "the fields of artificial intelligence and neuroscience are developing a symbiotic relationship in which theoretical and empirical breakthroughs in one field can lead directly to breakthroughs in the other. Could the next breakthrough in either field come from stochastic resonance (SR)?" SR describes the counterintuitive result that noise benefits signal detection. As noise increases in an SR benefiting system, the signal-to-noise ratio (SNR) rises sharply to a maximum value, then gradually decreases for higher noise intensities [2]. SR is widely used in increasing without limit the precision of signal measurements in natural and artificial systems. For example, in neural networks (NNs), it is known that when training one, adding noise to the training data set can improve network generalization, i.e. it's ability to fit real data outside of the initial training set [3].

This paper introduces an artificial intelligence theoretical advance that links together SR, super-Turing mathematical complexity, universal approximation, and Analog Recurrent (AR) NN models. A super-Turing computational model is a class of machines that is stronger than a digital/deterministic Turing machine. They are more powerful than Turing machines which are the most capable digital devices. During the period when he thought that the Universal Turing Machine (UTM) represented the strongest computation class known, Newell [4], who elucidated cognitive theories, voiced the concern that they could not "compute an uncountable number of functions...". He thought a system with an uncountably infinite state space would best describe the brain. This paper focuses on a super-Turing computation class that has such a cardinality [5]. (Note: we use "state space" in a physical context, and "cardinality" in a mathematical context.)

Super-Turing mathematical complexity of ARNNs and some details of universal approximation theory have been quiescent for a couple of decades. This is most likely due to the fact that the physical realization depends on analog hardware implementation. While there has been a rapid development and exponential advancement in the field of digital hardware and software, analog hardware development has progressed at a much slower pace. However, as Siegelmann has shown in [6] and as we also demonstrate in this paper, digital hardware cannot mimic chaos which is a super-Turing computation indicator. Digital technology needs to be augmented with analog technology to attain better cognitive computing. Then NNs enhanced with analog hardware and non-algorithmic learning is a path forward which emulates biological learning vastly better than digital techniques alone could possibly achieve.

Super-Turing computational complexity proofs published in 1999 [7] have been quiescent for a while, probably due to a belief that a UTM is the strongest computation class any physical machine can achieve. Adding the concept of "any physical machine" is an inappropriate extension or misapplication of the Church-Turing Thesis [8]. Siegelmann [9] initiated the studies of ARNNs attempting to prove that their computational complexity was the same as the UTM (the P complexity class). However, the proof was not attained and in the process, Siegelmann discovered two super-Turing computational complexity classes (BPP/log* and P/poly). Rational-numbered synaptic weighted NNs with stochastic signals compute the BPP/log* complexity class. Real-number-weighted

978-1-7281-2009-6/$31.00  paper N-19260.pdf

NNs compute the stronger P/poly complexity class regardless of whether the signals are deterministic or stochastic [7]. For brains and other cognitively computing substrates operating in a 300K environment, stochastic signals are unavoidable. For a finite, quantized universe, real-numbered synaptic weights are not possible [10]. However, rationally numbered synaptic weights are possible. Stochastic signals and rationally numbered synaptic weights leave the BPP/log* computational class as a physically realizable, super-Turing option for biological brains and ARNNs to outperform Turing machines. A stochastic resonance process provides the /log* (prefix logarithmic advice) oracle in its super-Turing proof (see section III-A).

Another quiescent topic is the impossibility of digital computers to implement universal approximation [11]. Wray and Green [11] demonstrate that the universal approximation proof relies on real numbers. They prove that multi-layer perceptron and radial basis function NNs using digital computer calculated output functions cannot achieve universal approximation. The output functions computed result in polynomials with a limited number of terms for optimizing the synaptic weights. This limited number of terms grows slowly as the logarithm of the number of hidden nodes while the precision does not increase. Most citing papers ignore the result that digital computers cannot achieve universal approximation. However, in this paper, we demonstrate that SR applied to analog signals can attain universal approximation since precision grows as the logarithm of the number of readings. Given that the readings are much less costly than hidden nodes, seeking unlimited precision for implementing universal approximation is possible in such analog systems, as we discuss further in section III-B.

This paper focuses on the concepts of analog signals which differ significantly from the digital techniques currently used to develop NNs. SR overcomes the inability of digital computers to truly achieve universal approximation. The main thrust is that the SR of artificial intelligence and neuroscience is involved in the proof of super-Turing BPP/log* networks. We present chaos mimicking empirical results that are shown to be consistent with physically realizable super-Turing operations. Multiple efforts are using analog hardware to duplicate cognitive computing in the brain [12] [13] [14].

The outline of the paper is as follows. We discuss the details of SR, super-Turing complexity, and universal approximation in section II. In section III, we integrate these separate topics. In section IV, we present experimental results and in section V, we conclude with a discussion of related open ended problems.

## II. Background

### A. Stochastic Resonance

Stochastic Resonance theory states that increase in input noise can result in an improvement in the output Signal-to-Noise ratio (SNR) of an SR benefiting system [2]. SR was first described to explain the transitions between ice ages and interglacials [15]. Since then, it has been applied to or used in many systems including analog-to-digital (A/D) conversion, neuron operation, visual perception, and opinion formation [2]. SR is a well understood phenomenon that increases precision and resolution anywhere there is a threshold (i.e. A/D conversion, neuron firing, visual perception, opinion formation, etc.). Although each system has an optimum noise level, there is no limit to the precision or resolution increase. One impressive application is a single pixel camera image which results from properly adding many single-pixel readings of a scene filtered through thousands of sequential, random spatial filters [16].

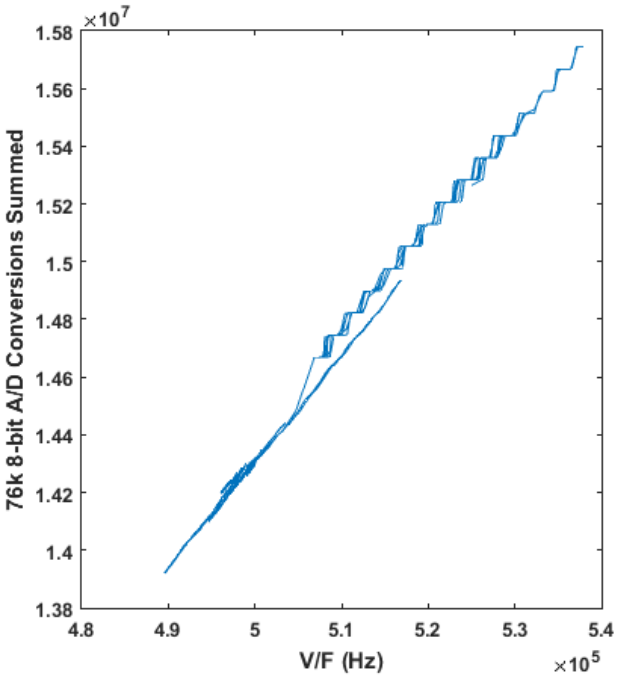


Figure 1. Two methods measuring analog barometric pressure illustrating Stochastic Resonance (SR) and lack thereof. This is from two synchronous, 50-minute time series of 1-second readings. The x-axis is voltage-to-frequency (V/F) conversions of the pressure with about 19 bits of precision. The y-axis is nominally 76,000 summed 8-bit ADC readings taken during the V/F integrating/counting time. No appreciable noise affected the summed ADC readings for upper branch leading to no significant precision increase, i.e. no SR. The analog noise changed for the lower branch (last 2/3 of time series). SR increased the summed ADC precision by about 16 bits.

To understand SR, consider a two–state system, [0,1], with a signal, $a \in [0, 1]$. Also consider that there exists a transition level at $t = \frac{1}{2}$. If $a \geq t$ (or, otherwise $a < t$) then multiple measurements of $a$ will read 1 (0). But, if the signal has uniform noise so that its average value is $a$, then multiple measurements of it will read 1 for about $(a)$ portion of the measurements and 0 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.

Fig. 1 illustrates a transition from no SR to SR for an A/D conversion system. It illustrates the additional precision increase that SR provides. The data came from investigating whether multiple A/D conversions could approach the precision of integrating a barometric pressure voltage-to-frequency (V/F) conversion over a nominal 1-second period. During the period, as many A/D conversions as possible were taken (i.e. -

76,000). The data were taken while the barometer was in a car travelling over a course where the total altitude change was about 300 ft (100 m). The V/F measurement is a linear process while the A/D conversions are non-linear (have thresholds) and can benefit from SR [2]. The upper portion of the data shows SR essentially not being active. Analog voltages between the 8-bit precision steps default to the step above or below. About a third of the way through the data run, the noise in the analog data changed and resulted in the lower portion of the data. However, it was not possible to characterize the noise to determine if it were optimum [2]. Nevertheless, the precision increase was limited only by the number of conversions, $log_2(76,000)$. There is no mathematical bound to the number of conversions. The computer program's 3-byte summing memory accumulated all bits from the 76,000 A/D conversions. In other cases, the SR precision increase is stymied by a digital computer's storage limits or lack of making more-and-more conversions. The SR results here have about 5 bits more precision (-24 vs. -19) than the V/F results.

In addition to the logarithmic precision increase that SR provides, it also provides the prefix logarithmic advice essential for the super-Turing complexity class described next.

### *B. Rational Stochastic Networks and BPP/log**

We summarize the theoretical proof of the complexity of rational stochastic networks, as modeled by probabilistic Turing machines (TM) that use binary coins with real probabilities [7]. The underlying concept is that ARNNs with real weights are able to implement a computational model that is stronger than a real probability TM, based on the Markovian model of stochasicity. According to [7], the time complexity of real probabilities TM is BPP/log*, i.e. Bounded-error Probabilistic Polynomial-time augmented with prefix logarithmic advice. In a probablistic TM, real probabilities and rational probabilities with log prefix advice are equivalent. Let $BP_R(T)$ denote the class of languages recognized by probabilistic bounded error TM that uses real probabilities and compute in time $T$ i.e. real numbered probabilistic TM. $BP_Q(T)/\log*$ denotes the class for TM based on rational probabilities along with a prefix advice i.e. rationally numbered probabilistic TM with prefix logarithmic advice. Lemma II.1 [7], [17], [18] states that these two classes are subsets of each other to a polynomial degree.

**Lemma II.1.** *The classes $BP_R(T)$ and $BP_Q(T)/\log*$ are polynomially related.*

This lemma demonstrates that for stochastic networks, "logarithmic precision suffices" for the real probabilities, i.e. for up to the $q^{th}$ step of the computation, only the first $O(\log q)$ bits in the probabilities of the neurons influence the result. We summarize the proof of the lemma below to provide a context for this paper.

To prove Lemma II.1, it is sufficient to demonstrate that:

1) $BP_R(T) \subseteq BP_Q(O(T\log T))/\log*$
2) $BP_Q(T)/\log* \subseteq BP_R(O(T^2))$

Thus, Lemma II.1 is proven in two parts with the latter part having two phases.

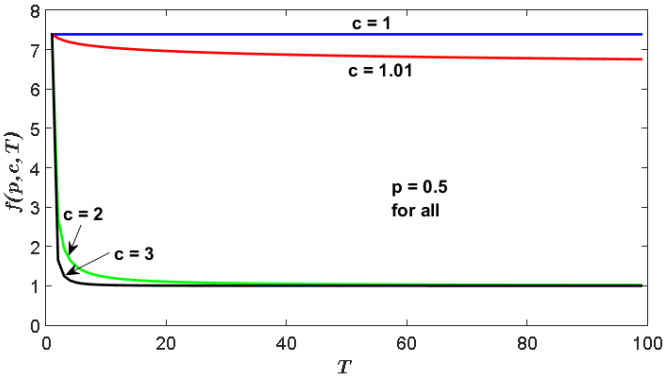


Figure 2. Plot of function $f(p,c,T)$ that relates error probability of $\mathcal{M}$ to $\mathcal{M}'$. $\mathcal{M}$ is a probablistic TM that $\epsilon$-recognizes a language $L$ in time $T(n)$. $\mathcal{M}'$ receives a prefix advice of length $c\log(T(n))$ for a constant $c$ and $\epsilon'$-recognizes $L$ in time $O(T(n)\log(T(n)))$, assuming $p \leq \frac{1}{2}$ and $c > 1$. As $T$ and $c$ increases, $f$ is bounded by 1 for $p = \frac{1}{2}$, which implies $\epsilon' = \epsilon$.

*Proof.* $BP_R(T) \subseteq BP_Q(O(T\log T))/\log*$

Let $\mathcal{M}$ denote a probablistic ($p \in [0,1]$) Turing machine that recognizes a language $L$ in time $T(n)$ with an error $\epsilon$. The error probability of $\mathcal{M}$ indicates its likelihood of generating a sequence of $(T(n))$ binary conjectures that would yield a wrong decision. From $\mathcal{M}$, one can build another probabilistic TM, $\mathcal{M}'$, implementing a model of $\mathcal{M}$ that receives a prefix advice of length $c\log(T(n))$ for a constant $c$ and also recognizes $L$ in time $O(T(n)\log(T(n)))$ with an error $\epsilon'$. In summary, assuming that $p \leq \frac{1}{2}$, the proof (details provided in [7], [17], [18]) demonstrates that the error of $\mathcal{M}'$ is related to error probability of $\mathcal{M}$ and given by:

$$\epsilon' \leq f(p,c,T) * \epsilon, \tag{1}$$

where $f(p,c,T)$ is a decaying exponential function.

$$f(p,c,T) = (2p)^{T(n)}(1 + \frac{1}{pT(n)^c})^{T(n)} \tag{2}$$

By applying the exponential approximation $1 + x \approx e^x$, we can rewrite (2) as :

$$f(p,c,T) = (2p)^{T(n)} e^{\frac{T(n)}{pT(n)^c}} \tag{3}$$

which is a decaying exponential function of T for $c > 1$ and $p \leq \frac{1}{2}$, as shown in Fig. 2. Hence, the error probability of $\mathcal{M}'$, $\epsilon'$, is bounded by $\epsilon$, the error probability of $\mathcal{M}$.

□

This shows that a machine using rational probability (i.e. 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.

*Proof.* $BP_Q(T)/\log* \subseteq BP_R(O(T^2))$

Let $\mathcal{M}$ denote a probablistic Turing machine, having a fair coin and a logarithmically long prefix advice $A$ that recognizes

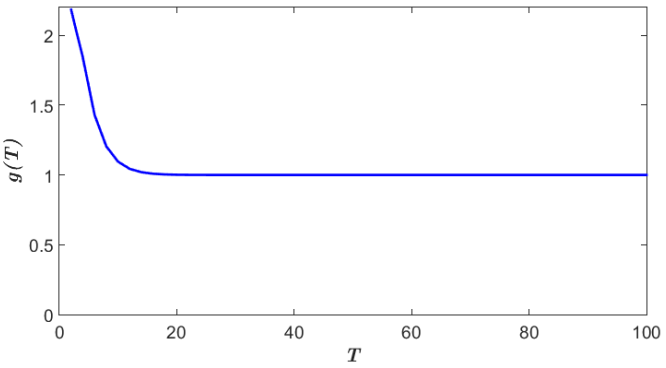


Figure 3. Plot of function $g(T) \approx e^{T(n)\frac{5}{8}^{T(n)}}$. After as few as 10 trials ($T = 20$) $g(T)$ closely approaches 1 and $\epsilon_2$ is nearly equal to $\epsilon$

a language $L$ in time $T(n)$ with an error $\epsilon$. Let $\mathcal{M}'$ denote an equivalent model with an associated real probability $p \in [\frac{1}{4}, \frac{1}{2}]$ that also recognizes $L$ in $O(T^2(n))$ with an error $\epsilon'$. $\mathcal{M}'$ computes in two phases:
Phase I - Preprocessing: $\mathcal{M}'$ estimates the advice sequence $A$ of $\mathcal{M}$ by tossing its unfair coin $z = cT^2(n)$, $c \geq 1$ times.

Let $\tilde{p}$ denote the estimation of the advice, defined as:

$$\tilde{p} = \frac{\#1}{z} \tag{4}$$

where $\#1$ is the number of ones found in the $z$ flips.

Thus the probability of $\mathcal{M}'$ estimation of advice being wrong, $\epsilon_1$, is determined using independent random variables to be:

$$\text{Pr(“wrong advice sequence”)=Pr}\big(|\tilde{p} - p| > \frac{1}{T}\big) \leq \frac{1}{4c} \tag{5}$$

where $c \geq 1$.

Phase II - Simulating the computation of $\mathcal{M}$: $\mathcal{M}'$ simulates each flip of the fair coin of $\mathcal{M}$ by up to $2T$ flips of the unfair coin. The algorithm involves two flips at a time that can end with $< 2T$ flips and must end at $2T$ flips with the details given in [7], [17], [18]. It proves that the error of $\mathcal{M}'$ to guess a bad sequence ($\epsilon_2$) is related to the error probability of $\mathcal{M}$ and given by:

$$\epsilon_2 \leq g(T) * \epsilon, \tag{6}$$

where $g(T)$ is a decaying exponential function for $T \geq 2$ ($T = 1$ is not possible since each simulation trial consists of two coin flips). Given $p'' = (p^2 + (1-p)^2)^{T(n)}$ as the upper bound of the probability of ending a coin flip simulation in $2T$ flips), we can express $g(T)$ as:

$$g(T) = (1 + p'')^{T(n)} \approx e^{T(n)\frac{5}{8}^{T(n)}} \tag{7}$$

where $p'' \leq \frac{5}{8}^{T(n)}$ when $p = \frac{1}{4}$.

As shown in Fig. 3, this means that $\epsilon_2$ is virtually equal to $\epsilon$ after about 10 trials (i.e. $T = 20$) and

$$\text{Pr(“bad guess sequence”)} \leq \epsilon_2 \approx \epsilon \tag{8}$$

Thus, the total error probability, $\epsilon'$, of $\mathcal{M}'$ which consists of Pr(“wrong advice sequence”) from phase I and Pr(“bad guess sequence”) from phase II i.e.

$$\epsilon' = \text{Pr(“wrong advice sequence”) + Pr(“bad guess sequence”)} \tag{9}$$

is bounded by:

$$\epsilon' \leq \epsilon_1 + \epsilon_2 \leq \frac{1}{2}. \tag{10}$$

□

SR is integral to finding the advice sequence $A$ in Phase I. The details of that will be shown in section III-A.

### C. Universal Approximation

Universal approximation capability is an important property of a NN as it guarantees that an arbitrarily close approximation exists for any desired class of continuous functions [19]. However, according to Wray and Green [11], if numerical limitations are taken into account, then the property of universal approximation no longer holds and the property of best approximation is irrelevant when comparing the approximation ability of networks. Numerical limitations refer specifically to the fact that these networks are implemented on computers with finite accuracy bounds, hence the property of universal approximation does not hold in practice. Wray and Green proved this using two types of NNs – multilayer perceptron (MLP) and radial basis function (RBF) network.

Consider a RBF NN with an input vector $u$ of dimension $M$, where $\mathbf{W}$ is the input weight matrix of all $w_{ij}$, $\sigma_1, ..., \sigma_N$ are the output functions of the $N$ hidden nodes, $\mathbf{c}$ the output weight vector and $y$, the output. For an MLP, the output of the network, $y$, can be expressed as the weighted sum of the hidden node outputs passed through an output function $\psi$.

$$y = \psi\Big(\sum_{i=1}^{N} c_i p_i(a_i)\Big) = \sum_{i=1}^{N} c_i p_i(a_i), \quad \text{if } \psi(x) = x. \tag{11}$$

where $a_i$ denotes the activation level of hidden node $i$ given by

$$a_i = \sum_{j=1}^{M} w_{ij} u_j \quad \forall i = 1, ..., N \tag{12}$$

and $p_i$ is the output of the $i$th hidden node in terms of its bias input $b_i$ and activation level, defined below:

$$p_i(a_i) = \sigma_i(b_i + a_i) \tag{13}$$

Note that for majority of the NN architectures, the output function of the nodes of the output layer, $\psi(x)$, can be simplified as $\psi(x) = x$. Hence, the simplified expression of (11).

The main crux of finite precision computation vs. universal approximation argument is based on the notion that the hidden node nonlinearities, $p_i(x)$, of the NN (whether MLP or RBF) are derived by the computers. This implies that the output of the function is calculated using a polynomial approximation. Thus, each hidden node can be considered as equivalent to

a polynomial which implies that the output of the network is a sum of polynomials which is equal to a polynomial $P(\mathbf{u})$. However, according to Leshno et al. [20], multilayer feedforward networks are universal approximators as long as their hidden node output functions are not polynomials.

$$y = P(\mathbf{u}) = \sum_{i=1}^{N} c_i p_i(a_i) \tag{14}$$

Therefore the network can viewed as constructing a polynomial approximation $P(\mathbf{u})$ to the training data. If we consider the case for one input $u$, to further clarify the polynomial nature, we can rewrite (14) as a polynomial series with a coefficient $\rho_k$ for each term.

$$y = P(\mathbf{u}) = \sum_{i=1}^{N} c_i p_i(a_i) = \rho_0 + \rho_1 u + \rho_2 u^2 + ... \tag{15}$$

where

$$\rho_k = \sum_{i=1}^{N} c_i d_{ik} w_i^k \tag{16}$$

where $d_{ik}$ denotes the coefficient of the $k$th power of the polynomial equivalent to $p_i(a_i)$.

We can further analyze how the internal weight changes during training affects the changes in the coefficients, $\rho_k$, of the network equivalent polynomial. This will elucidate the impact on the approximation of the NN.

Let $\Delta\rho_k$ denote the change in $\rho_k$, which is as a result of changes in its components, $c_i$ (the weight of connection from the $i$th hidden node to the output), $d_{ik}$, and $w_i$. As evident from (13), (15), and (16), change in $d_{ik}$ is due to a change in the bias input $b_i$. Thus, $\Delta\rho_k$ can be expressed as an aggregated sum of these components as shown below.

$$\Delta\rho_k = \sum_{i=1}^{N} [k w_i^{k-1} c_i d_{ik} \Delta w_i + d_{ik} w_i^k \Delta c_i + c_i w_i^k \Delta d_{ik} + 0(\Delta^2)] \tag{17}$$

where $0(\Delta^2)$ denotes the terms in $\Delta^2$ or higher.

The equivalent output polynomial converges thus as $k$ increases, the limit of the change in the $\rho_k$ goes to zero.

$$\lim_{k\to\infty} \Delta\rho_k = 0 \tag{18}$$

Thus, independent of the training algorithm, the change in the output polynomial coefficients due to the change in the internal weights goes to zero as the polynomial order $k$ increases. According to Wray and Green [11], this observation is significant. The implication of this is that the coefficients of the power terms in the equivalent output polynomial $P(u)$ higher than a certain limiting order, $k = L$, will not be updated by the training algorithm. This is as a result of the size of the coefficient change, $\Delta\rho_k$ being smaller than the numerical precision of the computer performing the network simulation.

We can split the equivalent output polynomial $P(u)$ into two polynomial segments:

$$P(u) = P_L(u) + P_R(u) = \sum_{j=0}^{L} \rho_j u^j + \sum_{j=L+1}^{\infty} \rho_j u^j \tag{19}$$

The first segment $P_L(u)$ denotes the finite length polynomial model with adjustable parameters while the remainder polynomial $P_R(u)$ denotes the portion of the output polynomial whose terms cannot be changed by the network. This digital computer model cannot make changes to achieve universal approximation.

## III. Linking Stochastic Resonance, BPP/log* Complexity, and Universal Approximation

### A. Stochastic Resonance and BPP/log*

In the second part of the proof of Lemma II.1 (section II-B), the prefix logarithmic advice is given by a Phase I process of flipping an unfair coin having a real probability $z$ times. Then, an estimate of the coin's probability is (see (4))

$$\tilde{p} = \frac{\#1}{z}$$

where $\#1$ is the number of ones found in the $z$ flips. Since the flips not equal to one are zero, (4) could be rendered

$$\tilde{p} = \frac{\sum_{i=1}^{z} f_i}{z} \tag{20}$$

where $f_i$ is the value of the $i$th flip.

Although a discrete, two-state system does not map to the double-well system of SR theory, it nonetheless exhibits SR [21]. In a practical SR context, (20) denotes summing $z$ 1-bit A/D conversions of a stochastic signal. Flipping an unfair coin is also a two-state discrete process with a transition threshold located at the unfair coin's probability. The proof goes on to state that $\tilde{p}$ from (4) is the logarithmically long advice of $BP_Q(T)/\log *$. And $\tilde{p}$ from (20) is exactly the same. Therefore, the process of generating the logarithmically long advice is implicitly an SR process even though that is unstated in the Lemma's proof in [7], [17], [18].

A neuron is often considered a low-precision device. Since neurons have thresholds and SR would be operative [2], a speculation is to consider a neuron as a 1-bit A/D converter. With an estimated sample rate of 100 Hz [4] and operating continuously, through its lifetime (maybe 80 years), its SR precision is about 38 bits. SR turns a low-precision device into one of high precision.

### B. Stochastic Resonance and Universal Approximation

Equation 19 (section II-C) summarizes the limits digital computers have [11]. That reference shows that NNs on digital computers cannot be true universal approximators. While $L$ can grow as the logarithm of the number of hidden neurons, $P(u)$ cannot reach the real-numbered weight adjustments necessary to achieve universal approximation. The exponential growth in neuron number to increase $L$ is very difficult from a resource perspective. However, multiple readings enabling SR are much less resource intensive and automatic since time never stops advancing. SR will keep increasing $L$ as long as the cognitive computing continues, whether hardware or biological. Moreover, [7] has two important mathematical proofs for ARNNs. The first states that "for stochastic networks, 'logarithmic precision suffices' for real probabilities,

Table I
SUMMARY OF NEURAL NETWORK COMPATIBILITY WITH CHAOS.

| *Network* | *Periodic* | *Period* | *Lyapunov* | *Chaos Compatible* |
|---|---|---|---|---|
| DRNN9 | yes | 22 | none + | No |
| DRNN18 | yes | 109 | none + | No |
| OpticARNN | no | N/A | 1.083 | Yes |

A neural network that fails even one chaos consistency test, fails at mimicking chaos. Two tests: the process must be aperiodic and have a positive Lyapunov exponent. Both digital neural networks fail the tests; OpticARNN passes both tests.

that is, for up to the $q^{th}$ step of the computation only the first $O(\log q)$ bits in the probabilities of the neurons influences the result". The second states that "a long sequence of coin flips allows indirect access to the real valued probability, or more accurately, it facilitates its approximation with high probability". The authors are not aware of any such proofs related to similar increases in the number of hidden neurons even if a cognitive computer could increase them.

Therefore, NNs on digital computers with limited $L$ cannot perform universal approximation. However, ARNNs with SR and no limits on $L$ can achieve universal approximation.

### C. SR Fundamentally Enables True Artificial Intelligence

SR is a phenomenon that has wide application from changing climate regimes, to signal measurements, through social-science measurements. Neuron operation is included as well. By linking SR with the BPP/log* complexity class, we have demonstrated how a physically realizable super-Turing complexity theorem incorporates and is dependent on SR. This super-Turing class has an uncountably infinite cardinality (exponentially larger that what Turing machines have) [5]. SR provides for an artificial system to perform true universal approximation. Super-Turing operation, SR, universal approximation, and uncountably infinite cardinality are able to implement any function. This likely explains why human cognition is so varied and powerful. Digital computer neural networks struggle to perform at that level since they operate in a Turing machine's countably infinite cardinality.

## IV. TESTS FOR DETECTING SUPER-TURING COMPUTATION

Examples of physically oriented models that indicates super-Turing operation are chaotic systems, as they cannot be mimicked by the UTM [6]. According to Seigelman [18], due to their sensitivity to minute changes in parameter values, the behavior of chaotic systems can only be approximated by TMs up to a certain point in time, after which the level of precision is too low. Hence, the simulation results in this section focus on mimicking chaos to demonstrate a computation consistent with super-Turing operation.

There are three indications of chaos: determinism, aperiodicity, and sensitive dependence on initial conditions [22]. A system that fails any one is a non-chaotic system. To demonstrate consistency with chaos, all three conditions have to be met. We compare the performance of Optical Analog Recurrent Neural Network (OpticARNN) [23] with two Digital

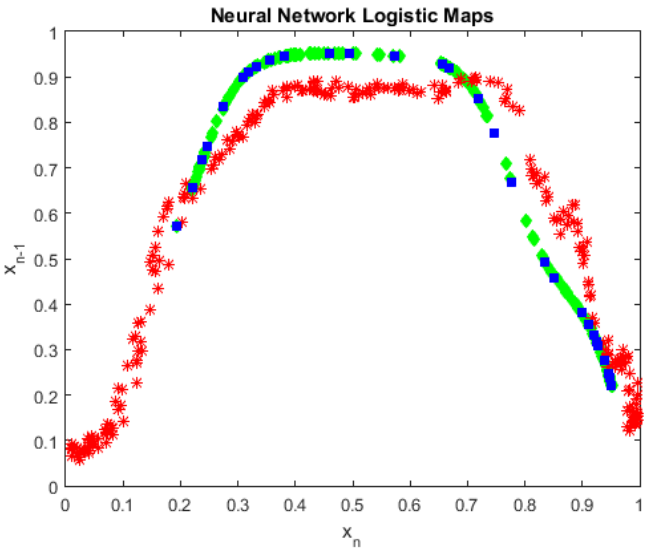


Figure 4. Results of DRNN models compared to OpticARNN model after training on the Logistics Map function. All three NNs had 2 inputs (bias and $x_n$), 5 $logsig$ hidden neurons, and 1 output neuron ($x_{n+1}$). The blue squares denote 9-bit DRNN (DRNN9) results while the the green diamonds: 18-bit DRNN (DRNN18), and the red asterisks: 9-bit OpticARNN. There are 20,000 blue squares repeated in 22 locations; 20,000 green diamonds in 109 locations. There are 255 red asterisks in non-repeating locations. The DRNNs are operating in feedback mode and the OpticARNN is operating in feed-forward mode. All NNs approximate the downward opening parabola shape of the Logistic Map.

Recurrent Neural Networks (DRNNs), one with precision (9bits) similar to OpticARNN and another with twice the precision (18bits) [23]. They all had 2 inputs, 5 hidden neurons, and 1 output neuron. DRNNs are completely deterministic systems while ARNN have a deterministic component, thus all three attain the determinism condition [22].

The OpticARNN, described in [23], [24], is an optical matrix multiplier [25]. It multiples an input vector by a synaptic weight matrix and results in an output vector detected by a line camera. The OpticARNN was not designed with SR in mind, but the photonic noise and multiple thresholds in the line camera's digitization caused SR to be active.

Table I summarizes the performance of all the three NNs tested for mimicking chaos [23]. OpticARNN was designed and trained seeking to implement super-Turing complexity. The OpticARNN results showed compatibility with chaos by having a positive Lyapanov exponent and no repeating time series (i.e. aperiodic). The DRNN model (DRNN9) limited to calculating with 9 bits of precision (OpticARNN's precision) as well as the DRNN limited to 18 bits of precision, DRNN18, (twice OpticARNN's precision) were similarly tested. The DRNNs failed to mimic chaos. Digital computers can calculate a series from a chaotic model but will ultimately fail by repeating the series. Both DRNNs repeated and, thereby, failed the test for aperiodicity. They both also failed to show "sensitive dependence on initial conditions" as indicated by having no positive Lyapunov exponents.

Fig. 4 shows the results for all three networks for logistic

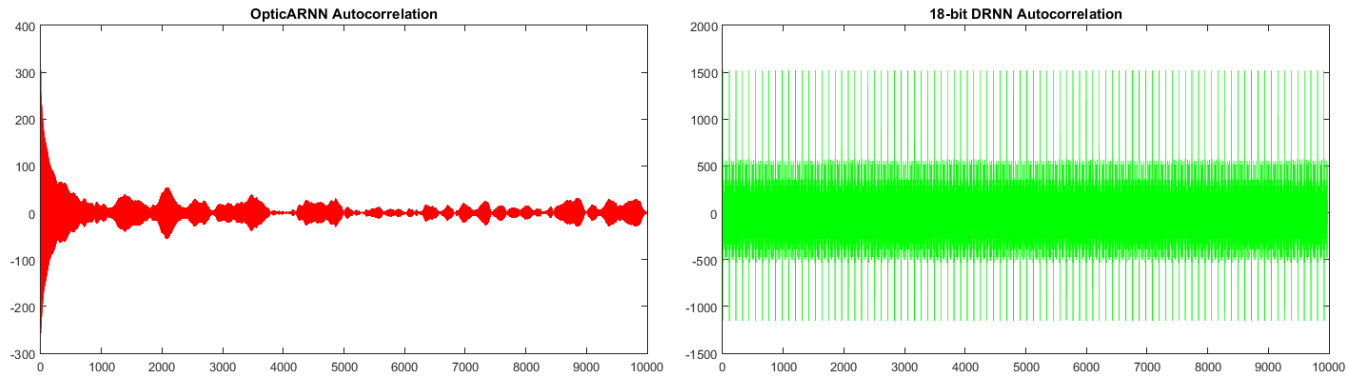


Figure 5. Autocorrelation over 20,000 time series for the OpticARNN and 18-bit DRNN. The OpticARNN results show no repeated sequences. The 18-bit DRNN repeats every 109 points as revealed by the peaks in its autocorrelation results.

map, a well-known chaotic function [22]. The training data was generated from $x_{n+1} = 3.99 * x_n * (1 - x_n)$. All three networks were trained on the same data with the inputs of $x_n$ and a bias. The five hidden nodes had a $logsig$ output function. The output neuron gave $x_{n+1}$. The DRNNs were trained with MATLAB's nntool GDX option. Because of analog inaccuracies in the OpticARNN, it needed further training on these weights via simulated annealing using its hardware-in-the-loop [23], [24], [26]. After training, all networks generated a time series 20,000 points long. The plot shows the DRNN9 repeating with a periodic cycle of 22 points. Fig. 4 shows only the 22 points, because they were plotted exactly on top each other. For DRNN18, the repeat cycle was 109 points. All 20,000 points were plotted for each DRNN. Only 255 points from the OpticARNN time series were plotted to give the outline of its mimic of the logistics function. Plotting more points would have obscured its chaotic behavior.

This shows that the OpticARNN (based on super-Turing BPP/log*) can perform at least one computation that a digital computer (based on Turing Machine) cannot perform. Moreover, it can do this because of the BPP/log* properties of continuous-valued calculations and stochasticity.

Fig. 5 shows autocorrelation diagrams for 20,000 long time series of the OpticARNN and the 18-bit DRNN. Fig. 6 shows autocorrelation results over a 220 shift range for all three NNs. This is done so 9-bit DRNN peaks can be distinctly seen. Fig. 6 also shows three peaks for the 18-bit DRNN. The OpticARNN shows no repeated peaks for the scales of both figures. Hence, no periodicity is detected in the OpticARNN result. For a 20,000-length time series it passes the test by showing no repeats indicating consistency with chaos. The DRNN fails this test by showing peaks. Peaks occur when the time series is shifted multiples of 109 positions, then multiplied with the unshifted series, and the products summed. Notice that the DRNN vertical scale is 5 times as large as the OpticARNN vertical scale. When the series and its shifted version correlate, the summed product values are much higher than when uncorrelated.

These results demonstrate that an OpticARNN is consistent with the possibility of super-Turing computation. Additional tests are needed which can verify super-Turing computation, show its superiority, and demonstrate its practical benefits. While the chaotic mimicking test may be of limited applicability, we believe that super-Turing computing could be much more powerful when used to model biological systems, particularly neural networks; and in developing more powerful Artificial Intelligence systems.

## V. Conclusion

So, we return to the opening question (section I), "Could the next breakthrough in either field [i.e. artificial intelligence or neuroscience] come from stochastic resonance?" [1] The answer is Yes. This paper describes a theoretical and empirical breakthrough centered around stochastic resonance with fundamental implications to both fields.

On a substrate supporting analog signals in a rationally numbered synaptic weight NN (biological or physical):

- SR evaluates signals to unlimited precision achieving indirect access to real numbers.
- SR enables and helps prove rationally weighted, stochastic-signal, physical ARNNs have BPP/log* super-Turing computation complexity and an uncountably infinite cardinality.
- SR provides unlimited precision so NNs truly have universal approximation.

Analog processing will not replace digital, which is very good at deterministic, precise computations. The problems facing the AI/ML community are largely those where the super-Turing power can potentially provide superior results. This breakthrough needs to be extended with the development of methods that demonstrate the increase in power of NNs implementing analog signals, SR, super-Turing computation, and true universal approximation.

## References

[1] 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.

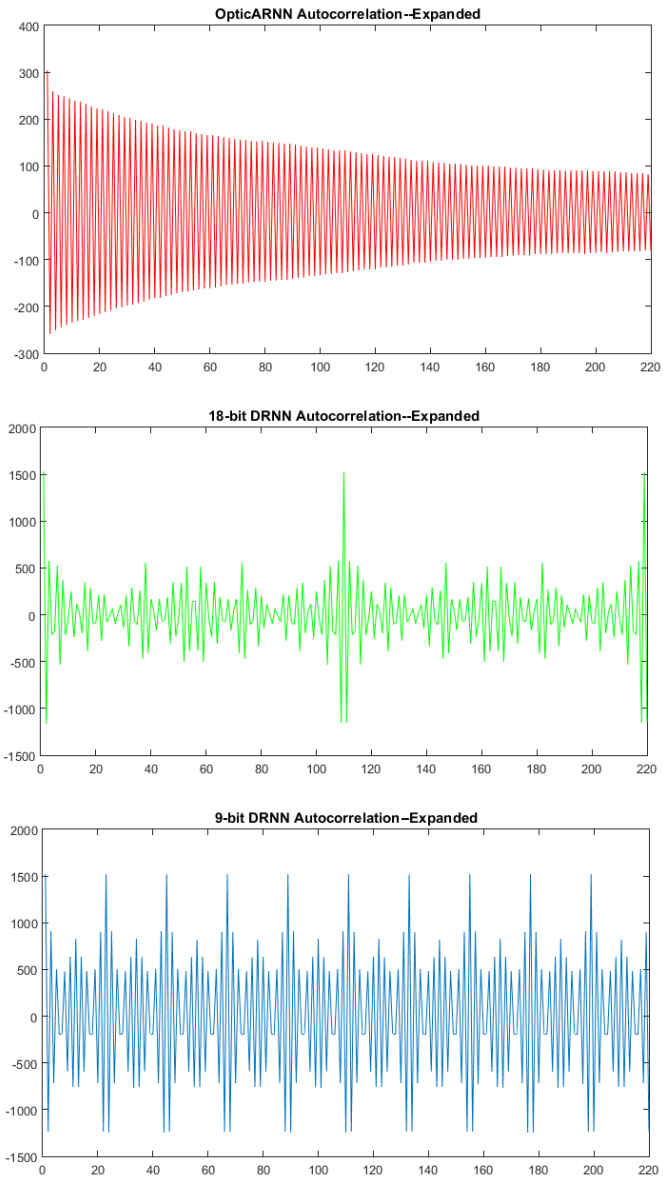


Figure 6. Expanded autocorrelations. The autocorrelation results for three logistic map trained neural networks shown on an expanded scale. The OpticARNN results show no repeated sequences; the 18-bit DRNN repeats after 109 iterations; the 9-bit DRNN repeats after 22 iterations.


[2] G. P. Harmer, B. R. Davis, and D. Abbott, "A review of stochastic resonance: Circuits and measurement," *IEEE Transactions on Instrumentation and Measurement*, vol. 51, no. 2, pp. 299–309, 2002.

[3] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[4] A. Newell, *Unified theories of cognition*. Harvard University Press, 1994.

[5] J. Cabessa, personal communication, 2014.

[6] H. T. Siegelmann, "Computation beyond the turing limit," *Science*, vol. 268, no. 5210, pp. 545–548, 1995.

[7] ——, *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhauser, 1999.

[8] D. Deutsch, "Quantum theory, the church–turing principle and the universal quantum computer," *Proc. R. Soc. Lond. A*, vol. 400, no. 1818, pp. 97–117, 1985.

[9] H. T. Siegelmann, "Turing on super-turing and adaptivity." *Progress in biophysics and molecular biology*, vol. 113, no. 1, pp. 117–126, 2013.

[10] G. Lokhorst, "Why i am not a super-turing machine," in *Hypercomputation Workshop, University College, London*, vol. 24, 2000.

[11] J. Wray and G. G. Green, "Neural networks, approximation theory, and finite precision computation," *Neural networks*, vol. 8, no. 1, pp. 31–37, 1995.

[12] S. Glatz, J. N. Martel, R. Kreiser, N. Qiao, and Y. Sandamirskaya, "Adaptive motor control and learning in a spiking neural network realised on a mixed-signal neuromorphic processor," *arXiv preprint arXiv:1810.10801*, 2018.

[13] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 144–164, 2019.

[14] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[15] R. Benzi, A. Sutera, and A. Vulpiani, "The mechanism of stochastic resonance," *Journal of Physics A: mathematical and general*, vol. 14, no. 11, p. L453, 1981.

[16] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

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

[18] ——, "Stochastic analog networks and computational complexity," *journal of complexity*, vol. 15, no. 4, pp. 451–475, 1999.

[19] M. Nayyeri, H. S. Yazdi, A. Maskooki, and M. Rouhani, "Universal approximation by using the correntropy objective function," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 9, pp. 4515–4521, 2018.

[20] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[21] B. McNamara and K. Wiesenfeld, "Theory of stochastic resonance," *Physical review A*, vol. 39, no. 9, p. 4854, 1989.

[22] D. Kaplan and L. Glass, *Understanding nonlinear dynamics*. Springer Science & Business Media, 2012.

[23] 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.

[24] 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, 2014, pp. 379–391.

[25] J. Goodman, A. Dias, L. Woody, and J. Erickson, *Parallel incoherent optical matrix vector multiplier*. Technical Report L-723-1, Department of Electrical Engineering, Stanford University., 1979.

[26] R. C. Frye, E. A. Rietman, and C. C. Wong, "Back-propagation learning and nonidealities in analog neural network hardware," *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 110–117, 1991.