Turing on Super-Turing and Adaptivity

Hava T. Siegelmann
BINDS lab; Dept. of Computer Science; Prog. of Neuroscience and Behavior
UMass Amherst, Amherst MA 01002
hava@cs.umass.edu

Abstract

Biological processes are often compared to computation and modeled on the Universal Turing Machine. While many systems or aspects of systems can be well described in this manner, Turing computation can only compute what it has been programmed for. It has no ability to learn or adapt to new situations. Yet, adaptation, choice and learning are all hallmarks of living organisms. This suggests that there must be a different form of computation capable of this sort of calculation. It also suggests that there are current computational models of biological systems that may be fundamentally incorrect. We argue that the Super-Turing model is both capable of modeling adaptive computation, and furthermore, a possible answer to the computational model searched for by Turing himself.

I. Introduction

Like computers, organisms and their systems can be thought of as following "programs.". Because all extant computation is based on the Turing computation paradigm, researchers model natural systems in this way by default. Much progress has been made in explicating natural processes through the Turing model in everything from cellular function to memory and intelligence. But, there is reason to believe that biological computation is not limited to the Turing type and, in fact, cannot function as purely Turing computation. The problem, in essence, is that while many systems or aspects of systems can be well described in this manner, Turing computation though precise, can only compute what it has been programmed for; it has no ability to learn or adapt to new situations.

Adaptivity is the critical element in natural selection and therefore an underpinning of evolution; organisms unable to adapt sufficiently to their environment do not survive to reproduce. Adaptivity is equally critical in behavior and in the operations of virtually all systems and at all scales. It has long been considered an indicator of health, from both psychological and physiological points of view [Carver&Scheier 1982]. It is apparent in processes as varied as the immediate adaptation of sensory afferent neurons for tuning to levels of stimulation and for maintaining postural and movement control during locomotion [DiCaprio et al. 2002] to robust adaptation demonstrated in signal transduction networks and chemotaxis [Barkai & Leibler 1997]. And significantly, life experience has been shown to cause adaptive gene expression via epigenetics [McGowan & Szyf 2010].

The field of Artificial Intelligence (AI) looks to biology for answers on how to accomplish intelligent behavior; likewise, biology looks to computer models for insight into biological processes. Lacking the ability to express true adaptation, current computational models of biological systems may be fundamentally incorrect, which holds significant implications for both AI and biology.

We propose Super-Turing as an alternative computational form, capable of modeling adaptive and natural systems. This paper is organized as follows: Section II reviews Turing Computation. Section III describes Turing's search for superior computational models more closely aligned with human intelligence. We provide a possible answer to Turing's call in Section IV with the introduction of the Super-Turing Theory of Computation. We introduce the ARNN model in Section V as a demonstration of a Super-Turing machine model; we also introduce Super-Turing as a method of acting in an infinite environment while subject to resource constraints. In Section VI we provide an alternative analysis to compare Turing and Super-Turing systems in a more biological setting; we provide a different view on the Turing Test. In Section VII we discuss questions raised about computational models that go beyond the Turing limit; we reply briefly to these; some replies were written in 1950 by Turing himself. We end in Section VIII with a discussion and directions for future work.

II. Turing Computation

The Artificial Intelligence community was captivated in 2011, when IBM's Watson computer won an impressive series of Jeopardy matches against human champions - seemingly a feather in the cap of machine intelligence. Watson pushes the current computational paradigm toward its limits using massive computer clusters, increasingly large amounts of memory, enormous databases and clever programming with myriad "if-then-else" statements. But, Watson can only operate in very orchestrated, specific environments, for which its program was designed in advance. Alter the task beyond programmed parameters, and Watson, as powerful as it is, will not be able to perform the computation. Watson, like other AI programs, robots and, in fact, virtually all computers, is based on the Turing computational paradigm, invented by one of the greatest minds of the last century, Alan Turing.

II.1 Turing's 1936 Model

English mathematician and computer scientist, Alan Turing (1912-1954), is considered the father of computer science. As an undergrad at King's college, Cambridge, Turing developed a mathematical model for a machine that would simulate the "human calculators" of his time - clerks hired for rote number manipulation (e.g., sums, square roots, percentages, etc.).

Prior to Turing's model, only finite controllers existed - where a machine's structure determined its operations. It was Turing's innovation to separate the machine's structure from the operation it performed; instead his model makes use of a "Universal" (general) Machine, which executes a separate set of instructions (program) spelling out how to make particular calculations. Other Turing innovations are external storage used by the Universal Machine as "scrap paper" (memory) during calculations and the sub-routine – a program that can be called by the main program. All computers that have the structure of machine-memory-program – in practice all current computers – follow Turing's theoretical exercise to simulate the human-calculators of his time. And all Turing machines are limited in function to exactly what is written in the program fed to them.

II.2 Computational Biology Based on the 1936 Turing Model

In the field of System Biology, cells with their genetic regulatory networks have been described in computational terms since the 1960s: By analogy, cellular genetic material is the "program" that guides protein production; protein levels determine the evolution of the network, thus serving as "memory" [Kauffman 1969]. Bray used a computational analogy to describe the process of gene expression [Bray 1995], suggesting that protein based circuits are the devices by which unicellular organisms react to their environment instead of a nervous system. Classical Kinetics [Magnasco 1997], Dynamics of Ciliates [Landweber & Kari 2002], and Genetic Networks [Ben-Hur & Siegelmann 2004] are but a few model systems that were demonstrated to behave like the Turing model, attesting to the model's centrality. Yet, biology is replete with examples of systems changing to meet the requirements of changing environment: Mice given dietary supplements develop changes in fur color, weight and propensity to cancer - due to epigenetic changes [Morgan et al. 1999]; genetically identical, human monozygotic twins have been shown to diverge in their epigenetic modifications with age and lifestyle differences (Fraga et al. 2005). The classical computational Turing paradigm assumes a fixed program; however, long-term changes in phenotypic behavior based on environment weaken the analogy to Turing computation.

Turing computation has likewise been adapted to explain neural computation. The early mathematical model of a neuron by McCulloch and Pitts [McCulloch & Pitts 1943] considers it a binary device, leading to binary interactions in a network with other neurons. McCulloch stated:

> "…it was not until I saw Turing's paper that I began to get going the right way around, and with Pitts' help formulated the required logical calculus. What we thought we were doing (and I think we succeeded fairly well) was treating the brain as a Turing machine." [von Neumann 1963, Copeland 2004].

Rosenblatt's seminal "Perceptron's" learning algorithm (Rosenblatt 1958) introducing the first method to update neuronal connections also uses binary units. In fact, current models of biologically inspired spiking neurons (e.g., integrate-and-fire, leaky-integrate-and-fire) are direct descendants of the fundamental McCulloch-Pitts model, e.g., [Pillow et al. 2004]. In the neuroscience of memory, consolidation was considered the underlying process that created fixed, stable memory traces, well suited to modeling via Turing computation. Yet, the relatively recently recognized process of reconsolidation, has been shown to actively modify memory traces [Sara 2000]. Reconsolidation occurs shortly after memory retrieval, when memories become labile and capable of being updated by new experience. During this reconsolidation step, memories may fade, strengthen or change [Tronson & Taylor 2012]. Reconsolidation is at the heart of the adaptation that facilitates learning and thus at the root of our intelligence. With reconsolidation, memories can no longer be considered stable and fixed as assumed in the classical Turing paradigm.

With all its centrality, the Turing model is based on the concept of a predesigned, loadable program, lacking any sort of adaptivity at its core. Recognizing adaptivity's place as a key component of intelligence, other computational models need to be considered in AI, modeling natural systems and in future technologies.

III. Turing, Beyond His Universal Machine

The first to note that the Universal machine was insufficient to describe nature and imitate intelligence, was Turing himself:

> "Electronic computers are intended to carry out any definite rule of thumb process which could have been done by a human operator working in a disciplined but unintelligent manner." [Turing 1950a]

This year, the centennial of Turing's birth (1912), many biographical articles were published about him and his work. It is telling however, that given such intense scrutiny, his early work is highly emphasized, while Turing's true passion - to find a more appropriate machine to describe nature and intelligence - has been almost universally overlooked. Yet, it is this rarely mentioned area of Turing's research that history may very well identify as his most visionary.

III.1. Turing's Passion: The New Type of Machine

In a letter to cyberneticist, W. Ross Ashby during the design of the ACE (1946), one of the earliest Turing computers, Turing wrote:

> "I am more interested in the possibility of producing models of the action of the brain than in the practical applications to computing." [Copeland 2004]

In “Intelligent Machinery, A heretical Theory,” (on “The ’51 Society,” a BBC radio program) Turing states:

> “My contention is that machines can be constructed which will simulate the behaviour of the human mind very closely. They will make mistakes at times, and at times they may make new and very interesting statements.” [Turing 1951]

Turing’s mind ran far ahead of those around him. While his thoughts turned to a new type of computation, his peers were still trying to understand and implement his 1936 machine. Turing’s boss at the National Physical Lab, Sir Charles Darwin, was displeased with Turing:

> “hitherto the machine [ACE] has been planned for work equivalent to that of the lower parts of the brain, and he [Turing] wants to see how much a machine can do for the higher ones; for example, could a machine be made that could learn by experience? This will be theoretical work, and better done away from here.”[Darwin 1947]

Turing replied by requesting a year off, and using it to write what is considered today, the first manuscript of AI, “Intelligent Machinery” [Turing 1948]. Darwin was still displeased, describing the manuscript as a ‘schoolboy’s essay’ and ‘not suitable for publication’ [Copeland 2004].

III.2 The Principles of Turing’s New Paradigm of Computation

In various works and interviews, Turing cumulatively described four different principles governing the creation of a better, stronger computational machine. I refer to them together as, “the four principles of Turing’s paradigm of superior computation.”

1. Series of Machines: Turing introduced Ordinal Logic (PhD thesis, 1938), showing how a logic can be associated with any constructive ordinal number, and that it is possible to climb repetitively from a given logic to one that is incrementally more complete. In a letter to Newmann (1940) Turing explains:

> “One imagines different machines allowing different sets of proofs, and by choosing a suitable machine one can approximate ‘truth’ by ‘provability’ better than with a less suitable machine, and can in a sense approximate it as well as you please.”

Turing suggested the use of a series of machines as a way to get a stronger logic. The process of attaining the next machine in the series is complex; Turing related it to the faculties of intuition and ingenuity [Turing 1939].

2. Learning & Adaptivity: On February 1947, Turing lectured to the London Mathematical Society about his work in designing and building ACE at NPL. While ACE was supposed to be a mere implementation of the universal machine, Turing explains:

> "What we want is a machine that can learn from experience." [Turing 1947]

He suggests allowing the machine to make mistakes in order to be creative:

> "There are indications however that it is possible to make the machine display intelligence at the risk of its making occasional serious mistakes." [Turing 1947]

Turing clarifies the extent of learning he would like to see in a BBC program from January 1952:

> "If the machine is built to be treated only as a domestic pet, and is spoon-fed with particular problems, it will not be able to learn in the varying way in which human beings learn." [Turing 1952]

3. Randomness:
If we search only what we know in a deterministic process, we are limited to those answers that exist within our knowledgebase. To enable a computational system to gain efficiency and access to other choices, we can incorporate a random element within the deterministic program. Total randomness would be worse than no randomness at all; what is desired is a balance between information contained in the system and a level of randomness for inspiration, unpredictability and free will.

Turing considered adding a random element to the ACE [Turing 1947]. In his 1950 paper, "Computing Machinery and Intelligence," Turing introduces his notion of "partially random" computation:

> "An interesting variant on the idea of a digital computer is a 'digital computer with a random element'. These have instructions involving the throwing of a die or some equivalent electronic process;" [Turing 1950]

In a BBC talk "Can Digital Computers Think?" Turing clarified further:

> "a machine which is to imitate a brain must appear to behave as if it had free will, … make its behaviour depend on something like a roulette wheel or a supply of radium." [Turing 1951a]

4. Rich information:
Richness of information reflects quantitatively the space of possible behavioral

choices available to the system. Turing suggested a connection between probability and information reminiscent of contemporaneous advances in the field of Information Theory. Interestingly, in 1943, Turing, the presumptive father of computation, was posted to Washington, D.C. to assist in code breaking efforts; while there he met Claude Shannon, considered the father of information theory. Later, Turing suggested that in a practical sense, a deterministic machine might seem unpredictable to an observer, as if it had a random coin - given that the machine has access to rich information unknown to the observer:

> "A determined machine may appear partially random. This would occur if for instance the digits of the number π were used to determine the choices of a partially random machine, where previously a dice thrower or electronic equivalent had been used." [Turing 1948]

He further clarified:

> "It is not difficult to design machines whose behavior appears quite random to anyone who does not know the details of their construction." [Turing 1951a]

IV. The Super-Turing Model of Computation

Super-Turing is not merely a more powerful form of computation; it is entirely different than Turing computation; there exists a hierarchy of computational powers directly related to the ability to adapt to the surrounding environment or equivalently to the richness of information available for operations - bounded with Turing computation at the bottom (least powerful and unable to adapt) and Super-Turing computation at its summit (most powerful and greatest ability to adapt). A natural system's computational power may fall anywhere in the hierarchy based on its particular level of richness and adaptability. The hallmark qualities of Super-Turing computation include efficiency using resources (e.g., precision and information, only to the extent needed) and flexibility in changing its program based on environmental pressures. Super-Turing computation encompasses principles found in biology.

Consider the possibility of early forms of life with only the Turing ability to act as programmed and no ability to adapt; the ability to adapt confers such great advantage in terms of survival that in the face of organisms with this ability, those lacking it would be quickly overtaken and soon extinct. An organism capable only of Turing type computation would need to store infinite programs to deal with whatever it might encounter including unanticipated and unlikely events. A solely Turing based organism is clearly untenable; the alternative and the only functional approach, is a system that adapts and learns based on its environment. Adaptation

can lead to an appropriate behavior ultimately comparable with a machine inefficiently programmed for every eventuality.

Suggesting that living organisms utilize Super-Turing computation at many system levels from cellular to processes involved in memory and intelligence does not mean that Turing computation is not present. On the contrary, repetitive functions like a plant's pollen tube extending to the ovules at the base of the pistil, are likely carried out through Turing calculations [Cheung 2010]. The process only becomes Super-Turing if something unexpected is encountered that necessitates adaptation. Having adapted and arriving at a more advantageous situation via Super-Turing computation, the new instructions can be coded as Turing sub-routines for repeated use increasing the system's efficiency by lessening computational overhead, as in the case of epigenetic change. The Super-Turing program can further modify learned Turing sub-routines as needed to suit future conditions.

There are countless examples of Super-Turing computation in biology, e.g., Bacteria collectively "glean latent information from the environment and from other organisms, process the information, develop common knowledge, and thus learn from past experience" [Ben-Jacov 2008]. Using this information, bacteria can change the morphology of their colony to defend against a threat. Bacteria that have most successfully altered genetically to be antibiotic resistant are used as bulwarks to protect those more vulnerable. In a behavioral context, an infant tries various forms of interaction with its environment, i.e. crying, cooing, smiling, etc., adapting its behavior to gain reward, i.e. attention, food, etc. The infant can both transfer and alter behavior toward other caretakers to maximize her benefits.

Super-Turing computation is not precise in the sense of Turing, it makes mistakes and tries new avenues. This does not mean that Super-Turing computation is inaccurate, rather, it constantly tries things and makes small corrections to achieve an overall accuracy supported by experience and learning. A similar concept supports the "ecological rationality" theory, suggesting that humans are not necessarily rational with respect to some logical formula but rather they follow heuristics to adapt that are fast, simple to compute and require little information [Gigerenzer et al. 1999].

IV.1 Super-Turing Computation – Brief History

In the early 1990's, as a grad student at Rutgers University, I was curious about the computational capabilities in a mathematical model of artificial neural networks, where the network is made up of neurons and information flows among them without any external tape for bits. While the Universal Turing machine has separate areas for a fixed, reliable memory and the program it runs, in artificial neural networks, memory and instructions are interleaved and the program is updated as the memory is. In the Turing model, memory is updatable, but the program cannot

be spontaneously altered. Intelligence is better modeled by the adaptive neural network since transformative experiences change our programs.

I wondered if the neural network model's information flow could be regarded as computation and made it into my grad dissertation under the guidance of Eduardo Sontag (1991-3). At the time I was unaware of Turing's four principles, yet I ended up independently arriving at these same four principles from a different direction.

A network of infinitely many binary neurons, e.g., where each neuron computes the NAND function, has long been known to be able to simulate a universal Turing machine [Turing 1948]. The model that interested me – which I named, "Analog Recurrent Neural Network" (ARNN), has neurons (the number is bounded to avoid unbounded computational resources), which communicate numbers in the analog [0,1] range instead of discrete binary values {0,1};. This allows the ARNN to model incremental decision-making. Consider a pilot landing on a runway. If choices were strongly discrete, this or that, the chances of successfully landing would be minute. In reality, small incremental control steps are made, each correcting or building on the previous in an unbroken chain, giving the pilot a better chance to bring the plane correctly onto the runway. A similar control principle was found for example in the Escherichia coli, which adapts to the environment by applying two movements: "run" towards an attractant and "tumble" in a random new direction. [Gomez-Ramirez & Sanz, 2012].

In an effort to understand the computation possible in a neural network, I tried to learn its limits. Classical computer science theory regards Turing machines as the most powerful form of computation. I approached the investigation with this understanding, yet, for each recursive (Turing) function I thought of, I managed to construct an ARNN that computed this function. I resorted to an upper bound proof, trying to prove that the general ARNN model cannot compute more than the Turing machine. This did not work either, so I took a different tack. I tried to prove the reverse - that ARNN *can* compute more than the Turing Machine. The rationale was that I would find through this effort where the ARNN's weaknesses lay, and then use this to bound the ARNN by Turing machines.

It caught me completely by surprise when I proved the ARNN was mathematically stronger than the Turing machine - what I ultimately called, Super-Turing computation (1993 thesis, Siegelmann & Sontag 1994), and that it would compute efficiently many more binary functions than the Turing machine is capable of (though not all existing binary functions). In a follow-up *Science* paper [Siegelmann 1995] I showed that Turing computation was insufficient to model chaotic systems of the sort one can expect to encounter in nature. Of particular interest is the fact that the simulation of chaotic physical/dynamical systems required the full Super-Turing strength of the ARNN. I showed that not only could the ARNN simulate chaotic systems, but a particular realizable chaotic system could simulate the Super-Turing ARNN; this was an important argument establishing Super-Turing as a practical complexity class in describing physical systems.

IV.2 Principles of Super-Turing Computation

Studying ARNN's I independently reached the following conclusions concerning Super-Turing computation, which correspond with Turing's principles of superior computation. I mention this not to credit myself, but to point out that two researchers following different paths both came to these same conclusions, which I believe lends credence to the idea that this is a naturally occurring computational form common to living organisms.

1. Series of Machines:
The Super-Turing theory of computation was described [Siegelmann 1998] mainly as an analog model of computation, but also as one that comprises a series of Turing machines, each calculating in its turn, each addressing a particular set of conditions, and while the execution of each machine is clearly Turing, the ability to create/choose another more appropriate Turing machine constitutes Super-Turing computation.

> "the machine can start with rational parameters, and increase its precision over time. The combined process of learning and computation creates the Super-Turing computation." [Siegelmann 1998, pg. 154]

2. Rich information:
A central result we proved in [Balcazar, et al. 1997], re-stated in [Siegelmann 1998], demonstrates a computational continuum spanning from Turing at the bottom of the hierarchy to Super-Turing at its height. The computational power of a particular system is based on the amount of available information that can be utilized during the bounded time computation.

> "Previous work in the field of information theory… has defined the complexity of a sequence as a 'measure of the extent to which a given sequence resembles a random one… we focus on one of the variants of the notion, called resource-bounded Kolmogorov complexity. This complexity is obtained by constraining not only the amount of information but also the time used by the universal algorithm to construct the number, thus making the notion of Kolmogorov complexity efficient… We prove that the predictability of processes such as those described by neural networks depends essentially on the resource-bounded Kolmogorov complexity of the … numbers defining the process…. We find that the equivalence between neural networks and either Turing machine … or classes such as P/poly [efficient Super-Turing] … are but two special extreme cases on the rich and detailed picture … spanned by the polynomial time neural networks." [Siegelmann 1998, pg 77-8]

3. Randomness:

Randomness is the antithesis of pre-programmed Turing computation. While many researchers proposed that mere randomness may make computation stronger, I proved that indeed adding a random element can make Turing computation more powerful in varying degrees based on the richness of the process or bias governing the random element. Turing recognized the necessity of adding a random element [1947], though he did not consider the randomness hierarchy from a mathematical standpoint.

> "The element of stochasticity, when joined with exact known parameters, has the potential to increase the computational power of the underlying deterministic process… we consider binary coins having real probabilities [or finite prefixes of the binary expansion of these reals]… It is perhaps surprising that the real probabilities strengthen the Turing machine, because the machine still reads only the binary values of the coin flips. However, a long sequence of coin flips allow indirect access to the …probability." [Siegelmann 1998; pg 121-2]

4. Learning & Adaptivity:

Any system that allows adaptation of its program is beyond Turing. Adaptivity is a key quality in learning and therefore intelligence; the proficiency of adaptivity in nature is based on the necessity of speed, accuracy and efficiency - and specifies the computational class beyond the Turing machine level on the Turing – Super-Turing computational continuum.

> "It is reasonable to suggest that the speed and accuracy of the learning process are the two parameters that determine the network's exact location on the computational continuum, ranging from the Turing machine (no learning) all the way up to Super-Turing (highest level learning)." [Siegelmann 1998; pg 154]

IV.3 A Note On The Terms, "Hypercomputation" And "Super-Turing"

The term 'Hypercomputation' [Copeland and Proudfoot 1999] was coined a number of years after the introduction of Super-Turing theory [Siegelmann 1993]. Since that time, Hypercomputation has become a catchall encompassing diverse concepts that are either Turing as in the case of Quantum Computation, but differ on a mechanical level from classical systems; or are not Turing, and may be more thought experiment and thus not subject to physical limits, e.g., a machine that computes the

next step twice as fast as the previous one, with no constraint to remain under the speed of light. Super-Turing is a very particular, feasible theory of computation, modeled after nature's adaptivity and along the same principles that guided Turing in his search for a new model, and thus the terms are not interchangeable.

V. The Analog Recurrent Neural Network (ARNN) Model

The ARNN constitutes a parametric model that can demonstrate how to attain Super-Turing power from Turing computation. While proposed as the standard model in the Super-Turing theory, other equivalent models exist that are Super-Turing even without following the neural network paradigm, e.g., the analog shift map [Siegelmann 1995] and Turing machines with a real coin [Siegelmann 1999].

The Analog Recurrent Neural Network (ARNN) prototype consists of an interconnection of N (finite number of) neurons, where each updates by the following equation:

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right), \quad (1)$$
$$i = 1, \ldots, N$$

All a_{ij}, b_{ij}, and c_i are numbers describing the weighted synaptic connections and weighted bias of the network, u are inputs arriving on M input lines in a stream, and P, (P<N) of the neurons will serve as output neurons, whose values will propagate on P output streams. The activation function σ applied to each neuron is the saturated-linear function defined by:

$$\sigma(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } 0 \le x \le 1, \\ 1 & \text{if } x > 1. \end{cases}$$

The main difference between this neuron and the Perceptron is that the activation function here is analog and it enables the communication of numbers in the range [0,1] between the neurons rather than only the {0,1} bits available in the Perceptron.

V.1 The Computational Hierarchy

The ARNN can be thought of as a parametric model that introduces a linear infinite hierarchy of classes of increasing computational power. At its lowest level, it is equivalent to the Turing model, when the ARNN is deterministic and its weights are small rational numbers. At the summit of the hierarchy is the full Super-Turing model [Balcazar et al. 1997; Siegelmann 1998].

1. Turing equivalence theorem: If the ARNN's weights are all loaded in advance (no adaptation) and are all simple short rational numbers (with no information content beyond their constant short length) then the resulting ARNN is exactly equivalent to a Turing machine. Some physiologists like this model due to the discrete property of the weights.

Proof description of Turing equivalence: The number of neurons is finite – as in the finite controller of the Turing machine. The input that comes to the Turing machine, here in the ARNN, comes as a sequence of bits from the outside or is the initial state of a designated input neuron. The output of the Turing machine is like the output of the ARNN and can be released in bits form or remain as a value of a designated output neuron. There is one major difficulty in proving the Turing equivalence: The Turing machine has an external unbounded tape where bits can be stored and then brought up by need; the neural network has no external tape, the only place where it has temporary information storage is in the values flowing among the neurons. Furthermore, whereas the Turing machine can run if-then-else statements and control the next steps toward different areas of the program, the neural network is a continuous dynamical system and has no ability to fork control. These two issues are solved by one trick: the space information processing and flow, including weights and neural values, will not take any number in the continuous range [0,1], rather it will take values of a fractal only, e.g., a Cantor set. Information flow on a fractal, we proved, still with simple ARNN, is exactly equivalent to a computation with a universal Turing machine [Siegelmann & Sontag 1995; Siegelmann 1998].

2. ARNN as in 1, but with additional binary random coin: Considering the same ARNN as in item 1 above, or equivalently a Turing machine, the incorporated coin has the output of 0/1 and it is included in the computation. In the Turing machine set-up it is part of the table affecting the next step. If the random coin is fair (has the probability ½ to fall on 0 or 1) then in the Turing machine paradigm, the resulting complexity class is BPP (bounded-error probabilistic polynomial time)- BPP is considered the largest practical class of problems, solved quickly on modern machines. Analogously, we can use a binary coin for the ARNN by incorporating it into neural integration processing as if it were a neural value. If the coin is fair, this model is equivalent to BPP as well. BPP is a Turing class.

I found that if the binary coin is not fair but rather relies on a probability value with rich information content, then the computational complexity of the ARNN with such a coin (and equivalent to a TM with such a coin) grows beyond the Turing power. The rich information probability can be expressed as a real number; yet, the number does not need to be considered in its full details beyond the first polynomially many bits in the binary expansion [Siegelmann 1998]. The particular class of the stochastic machine is on the hierarchy mentioned above, correlated with the descriptive complexity of the coin and how it is characterized by a Kolmogorov bounded resource measure. This extra computational power is an interesting phenomenon, since the measures taken from the coin are binary only, and it is only

the hidden probability that contain the rich information.

Asynchronous computation with ARNN, where different neurons update at different rates, will have the same exact computational power.

3. Turing Machines with non-uniform polynomial advice: Turing machines with non-uniform advice were suggested in [Karp & Lipton 1980] and were found to be computationally equivalent to the full Super-Turing model [Siegelmann1998]. These are Turing machines that work in polynomial time, and receive polynomially long advice: Such a machine uses the same advice for all inputs of the same length, but for inputs of different length, the advice can be different. An equivalent paradigm is a non-uniform family of Boolean circuits of polynomial size: where all inputs of same length use the same circuits but inputs of different length use a different one. This complexity class can be thought of as describing machines that can process input efficiently, but only after (information rich) pre-processing – the preprocessing is the same for inputs of same length.

The advice (and the growing or real number weights ARNN, items 4 and 5 below) serves as a theoretical bound since when advice is given, learning is no longer necessary. Similarly, Turing never considered an oracle as part of his search for the new intelligent machines; he used oracles [Turing 1939] purely for number theoretical proofs of relative computability and not as a practical form of computation.

4. ARNN with growing amount of bit precision in the weights: As stated above, the ARNN with rational weights but linearly growing precision in neurons is Turing equivalent. If we now let the weights' precision be linear in the computation time and have non-redundant information in bits, the model is completely Super-Turing. If the precision of weights is sub-linear or the information there is redundant – the machine is on the Super-Turing hierarchy between Turing machines and complete Super-Turing computation. Having a Super-Turing system defined with linear rather than infinite precision enables it to attain the series of machines by learning the weights.

5. ARNN with real weights: ARNN with real weights are Super-Turing. A lemma "linear precision suffices" [Siegelmann-Sontag 1994; Siegelmann 1998;pg 68] bridges between this model and the one described in item 4. The lemma states that in our Super-Turing model, if we are allowed only T time steps to reach a decision (for any integer T), not only we can reach the exact same decision by following programs having T bit precision as we would reach by following fully analog systems, but that in fact an observer would never know whether we confined ourselves to this limited information and precision or not. This is thus a mathematical support of using growing size heuristics.

6. Evolving ARNN: Evolving ARNN have at each stage rational numbers as weights, but the weights can change in consecutive steps. This configuration is the epitome of

adaptivity; and if the weights change in a non-recursive manner, the resulting model is indeed Super-Turing [Cabessa and Siegelmann, 2011]. This model fits the first Super-Turing principle of accumulating stages of learning and computation. Evolving weights are a step closer to biology than the growing weights model of item 4.

V.2 Super-Turing – A Method to Handle Infinite Environments

Like biological organisms that must deal with the infinite universe while having limited resources, the equivalence of full analog and finite precision networks (via the lemma "linear precision suffices") suggests that Super-Turing is capable of making decisions in an environment having infinite information when the machine (like the organism) must conserve resources, e.g., memory, information and time.

The ability to use only linear precision is related to the cognitive miser approach of cognitive psychology [Fiske & Taylor 1991]. "Cognitive miser" refers to impossibility of assimilating all the information we are bombarded with by the world and hence use only the information necessary for a particular task. A similar minimalist concept underlies the free energy theory, which describes perception as an "inevitable consequence of active exchange with the environment" [Friston 2009]. The "linear precision suffices lemma," in addition to supporting minimalistic processes, can also be interpreted (following the relationship between randomness and free will [Turing 1951a] and relating precision in coin and weights) as suggesting that there is no way to differentiate between a process that involves a free will and one that includes infinite determinism.

The Super-Turing paradigm can also be used for gaining efficiency within the Turing domain: an ARNN may have weights that are not truly real numbers but encode instead (via learning, evolution, etc.) complex functions that require exponential Turing time; the ARNN will calculate these Turing exponential functions in polynomial time, leading to significant gains in efficiency.

Theoretically, the SiSo thesis of computation [Siegelmann& Sontag 1994, Siegelmann 1998] stated analogously to the Church-Turing thesis of digital computation as it applies to adaptive, evolving or analog systems:

> *Any "reasonable analog/evolving computer" will have no more power (up to polynomial speedup) than ARNN.*

A thesis like this cannot be proved, only disproved; it remains to be seen whether the thesis holds up and how useful it is in expanding our understanding of natural systems and encouraging out-of-the-box thinking in creating new AI systems.

A given Super-Turing system will not always reach the top of the hierarchy since the complexity class is related to knowledge acquired; it will thus fall into the appropriate computational class in the infinite Turing - Super-Turing hierarchy.

V.3 Related Models

Two related models are worth mentioning. BSS, the model of computing over real numbers [Blum et. Al 1989] is equivalent to a Turing machine that uses real numbers in its registers instead of bits. The model seems similar to the ARNN, which uses real weights, but there are two chief differences: (1) the BSS model has the capability to compare any two (real) values to exact, infinite precision, and fork based on the result; this is very different than our ARNN model, which is not affected by full precision and gives the property according to, 'linear-precision suffices.' The BSS model is less feasible in the sense that full analog precision is necessary. The second difference is that the BSS model does not bound the range of parameters values; The ARNN's range is finite. The two models are compared in [Gavalda & Siegelmann 1999]. Thus, the Super-Turing model is a better fit for biological and practical modeling.

Another related model is described in a theory of computing in dissipative systems [Siegelmann, et. al 1999]: While the ARNN is analog in its phase space, this theory encompasses models, which can also be analog in time. We proved that in the dissipative model the classes P and NP are not equal (a question which is still open in Turing computation); we also demonstrated the practicality of our dissipative analysis in evaluating continuous algorithms for classical problems including Maximum Network Flow, an algorithm for Max, and the Linear Programming problem [Ben-Hur et al. 2003]. In the special case of dissipative ARNNs, the system can be analyzed according to the above stated theory.

VI. Counting Argument: Turing, Super-Turing, and the Turing Test

In the field of Computational Complexity, the complexity of a machine model is measured by observing and characterizing possible input-output functions.

In classical Computational Complexity, the machine is presented with a finite length input and it returns a finite length output. There is no meaning to the order of inputs the machine receives: it calculates and outputs the result, then erases its temporary memory, readies itself for the next input and to calculate the appropriate output from scratch.

In living organisms and in adaptive machines the order of inputs presented matters greatly, since previous experiences are noted in the memory and affect the next action of the machine.

A natural form of analysis, I refer to as the Mnemosyne standard, considers the full, lifetime input-output function of the machine: The input is an infinite sequence of

bits and the output is either finite (if the machine dies at some point in time) or infinite (for a machine that continues to work forever) [Cabessa & Siegelmann 2012]:

$$\varphi_S : \{0,1\}^\omega \longrightarrow \{0,1\}^{\leq\omega}$$

With this analysis, the order of finite sequences given to the machine over its life time matters, and in fact, since the machine never stops there is no explicit removal of memories.

(**Remark:** We consider the characterization of lifetime functions generated by different machine models to be a different form of computational analysis rather than a different computational model.)

We studied the computational power of both Turing machines and ARNNs under this Mnemosyne standard [Cabessa & Siegelmann 2012 and references therein]. We first proved that in the Mnemosyne standard, rational weight ARNNs are equivalent to Turing machines and that real weight ARNNs compute much more and thus are Super-Turing. Most interestingly, our analysis allowed us to count the number of different lifetime series available to the different machine models:

Aleph-null is the smallest infinite cardinal number corresponding to the cardinality of the natural numbers. It is denoted by:

$$\aleph_0$$

The total number of different functions defined from infinite binary sequences into infinite or finite binary output sequences is:

$$2^{2^{\aleph_0}}$$

Our results state that under Mnemosyne analysis, the total number of functions computed by a Turing machine is,

$$\aleph_0$$

while the number of functions computed by the real ARNN is:

$$2^{\aleph_0}$$

That is, the Super-Turing model is exponentially richer than the Turing paradigm, but it is still exponentially smaller than all possible functions. Perhaps what this result says is that given that cells, brains, and other biological systems are adaptive, they can present exponentially richer and less predictable behavior over their lifetime.

In 1950, Alan Turing introduced the "Imitation Game," in which a person (referred to as, "judge") must decide whether an entity behind a door is human or a "Thinking Machine," [Turing 1950]. It was less a significant experiment than a playful method of increasing the scientific community's interest in Artificial Intelligence [Copeland

2004]. The Imitation Game question comes down to a judgment of the richness, lack or excess of repetition, and adaptivity of replies delivered by agents. Though no digital machine has yet convinced a judge that it is human, the question the game is based on points to the richness of our intelligence. Perhaps a Super-Turing machine, with its far richer set of behaviors will be the first to succeed in this challenge.

VII. Some Possible Questions

Understanding the nature and practicality of Super-Turing computation has been clouded by a common misinterpretation of the Church-Turing thesis; it is often taken as stating, 'There is no feasible computational paradigm that can exist which surpasses the Turing machine." Turing never suggested that the universal machine is the only appropriate, or most computationally powerful model; on the contrary, much of his subsequent career was applied to the search for a more biological-like model. Yet, it can still be challenging to think of machine models, which are beyond Turing.

Various interesting questions have been raised regarding the feasibility of Super-Turing models:

a. The precision objection: ARNN use infinite precision and hence are not practical.

This is a basic misunderstanding of Super-Turing, which as previously stated, repeatedly constructs/chooses a particular finite Turing machine (typically by learning), using only as much precision as needed, bounded by the total number of computational steps and thus never computes full, infinite information. It is Turing computation that is incapable of dealing with infinite information in that any given Turing machine can address only the finite matter defined by its program. To understand how Super-Turing deals with the infinite information presented by nature, consider Super-Turing the biological program and different Turing machines as subroutines called by the Super-Turing program.

An interesting answer is given by Turing:

> "Most actual digital computers have only a finite store. There is no theoretical difficulty in the idea of a computer with an unlimited store. Of course only a finite part can have been used at any one time. Likewise only a finite amount can have been constructed, but we can imagine more and more being added as required." [Turing 1950]

Practically, the lemma "Linear precision suffices" means that the precision used in ARNN is much like the amount of memory bits used on the Turing Machine tape; For computation in T steps you do not need more than T bits (in either weights or neural precision). Turing explained [Turing 1947] that his use of infinite memory

was needed to achieve particular mathematical results, but in practical use, computers can have large, but finite memories. Similarly, we consider the unbounded precision for the mathematical result, understanding that in reality systems operate under physical limits.

b. The noise objection: ARNN suffers like other analog models from noise, and we cannot use much precision when noise accumulates.

Noise in biological and physical systems is different than in artificial systems and there are different kinds of noise: intrinsic, background, etc. In artificial systems we think of digital as being safer than analog. If we think of the Super-Turing model as interleaving steps of adaptation and computation, the linear precision suffices can be thought of as partial resistance to noise in the computation. If we are worried about analog noise in an artificial system implementation, we can use only neurons with a fixed amount of precision. In that case, we will need to use growing networks. Super-Turing Theory works with evolving digital hardware just as it works with evolving weights. This second implementation is not affected by noise.

But there is another view: It is possible that the particular sensitivity to noise is part of the adaptive power of the Super-Turing process. It is possible that biological systems may filter less important signals and focus their sensitivity on "noise" that imparts information, providing clues for adaptation.

c. The non-computability objection: ARNN requires non-computable weights for Super-Turing computability, but all constants in nature are computable.

According to Turing, computable numbers are those created by "repetitive non-intelligent steps;" it is not clear that nature is bound by this. Furthermore, random or changing environments can be non-computable by this definition. Lastly, we need only the prefix to be non-computable due to the linear precision property.

d. The linear precision objection: It is not possible that Super-Turing can be simulated by finite precision Turing machines and still be beyond Turing.

This interplay between finite and infinite precision is tricky. Unlike a Turing machine that is given its full program's description in advance, the Super-Turing, in its most generalized form, starts with an initial program description and learns future programs through exchange with the environment. When we look at a Super-Turing computer after it finishes a successful computation, we can say that it is equivalent to a particular Turing machine where its description length is related to the computation time. Yet, if the Super-Turing computer continues to compute in a dynamic environment, it will become a different Turing machine. So, the Super-Turing system can be simulated by an infinite series of Turing machines, one machine per computation on particular input in a particular environment and for a particular time bound. It cannot be simulated by a single Turing machine.

e. The conservative objection: Turing was brilliant, and hence anything that is not a Turing machine does not make sense; "I am a soldier in Turing's Army."

Alas, Turing's true passion was to find a more human-like machine; he gave up his job when his boss could not envision the need for Super-Turing machines. In fact, Turing prepared explanations in support for such superior machines: "I propose to outline reasons why we do not need to be influenced by the above described objections." [Turing 1948]. He replied [Turing 1950] to nine hypothetical objections to superior machines; some of the most interesting of them are:

> **1. The "Infallibility" objection:** The argument from Gödel and others that there are questions (theorems) which machines cannot answer (prove) with either Yes or No, rests essentially on the condition that the machine must not make mistakes. Turing points out that while we do not expect people to not make mistakes, we require it from a machine; and in fact, he says it is important to allow intelligent machines to be able to err:
>
> > "There are indications however that it is possible to make the machine display intelligence at the risk of its making occasional serious mistakes." [Turing 1947]
>
> **2. The "Theological" objection:** Humans are the only creatures that can think. Turing replies that this is a matter of faith and beyond scientific discussion.
>
> **3. The "head in the sand" objection:** Building machines that think would be a disaster. Turing replies that this is not a scientific argument against the possible existence of such machines.
>
> **4. Lady Lovelace's objection:** Current machines can only do what they are told. Turing replies - while it is true of current machines, this does not contradict the possibility that in the future superior machines could be built.

f. The simplicity objection: Aren't ARNN too simple to tell us anything about nature and organisms?

Computational models are simplistic structures that teach us something about more complicated systems. Super-Turing teaches us about adaptive systems in nature. A neuron in our system is very simple, yet the network's emergent behavior becomes increasingly more complex; and as the Super-Turing machine advances, it relies on more information and more experience and becomes quite complex in its behavior.

g. The motivation objection: Can the Super-Turing model explain anything that the Turing cannot?

Definitely, yes. While the Turing model cannot explain either reconsolidation or epigenetics or any other adaptive systems, the Super-Turing model does so.

VIII. Discussion and Future Direction

From the computational perspective, adaptivity of a biological system means that there are behaviors, which are not explicitly programmed into the animal. This is to be differentiated from the classical notion of self-updating Turing machines, where all update instructions are pre-programmed. A system is more feasible, efficient and more likely to succeed if it does not have to spell out behaviors for all possible future changes in inputs and conditions. Survival depends on acquiring new behaviors that the organism was not pre-programmed for. A precursor to adaptive behavior is flexibility - the ability to demonstrate multiple behaviors. A lack of flexibility in a system, a behavior or in a species, is often a harbinger of failure or extinction.

In a sense, our current efforts are all being placed in the basket of Turing computation. As demonstrated, life itself points out that another form of computation is necessary to explain biological systems. If we limit our research to Turing computation only, we may fail in our efforts to create true Artificial Intelligence and likewise we may misunderstand the mechanisms upon which life is based.

One of the most important early goals is to understand and replicate the ability of biological Super-Turing computation to make choices and adapt. We will also have to duplicate Super-Turing's ability to create appropriate Turing machines at each step of computation. Turing proposed learning in combination with the faculties of intuition and ingenuity. We need biological, neurological and psychological analysis of living systems to identify examples of Super-Turing computation and extract methods employed in computation.

Turing was a firm believer in the feasibility of a truly intelligent machine. In January, 1952 on a the BBC [Turing 1952], Turing spoke with Max Newman concerning intelligent machines, specifically, the Imitation Game:

> "**Newman:** I should like to be there when your match between a man and a machine takes place, and perhaps to try my hand at making up some of the questions. But that will be a long time from now, if the machine is to stand any chance with no questions barred?
> **Turing:** Oh yes, at least 100 years, I should say."

Turing also explained the importance of not entirely pre-programming an intelligent machine in advance:

> "If the machine is treated only as a domestic pet, and is spoon-fed with particular problems, it will not be able to learn in the varying way in which human beings learn." Adding, "Obviously if one were to predict everything a computer was going to do one might just as well do without it." [Turing 1952]

Taking Turing's belief of such a system into account, practical considerations still exist. If a Super-Turing machine were built, should it be digital or analog? Would we get better adaptive models on analog machines? When building the ACE, Turing chose digital technology, to allow universality and speed given that analog was not practical at the time; still, Turing pointed out:

> "The nervous system is certainly not a discrete-state machine. A small error in the information about the size of a nervous impulse impinging on a neuron, may make a large difference to the size of the outgoing impulse. It may be argued that, this being so, one cannot expect to be able to mimic the behaviour of the nervous system with a discrete state system." [Turing 1950].

Yet, In today's more advanced technology, analog technology may present a viable platform and one that more closely resembles the brain and other biological systems.

After replying to the nine objections to superior computation, Turing ended his Turing-test article [Turing 1950] with the sentence:

> "We can only see a short distance ahead, but we can see plenty there that needs to be done."

Let me join Turing in suggesting that while we have taken his ideas a small step forward, plenty of work toward Adaptive Super-Turing computation remains to be done.

IX. Acknowledgements

Research support from the National Science Foundation under grant #19200000000036 and the Office of Naval Research (ONR) under grant #N00014-09-1-0069 for this study are gratefully acknowledged. I also thank Eric Goldstein for suggestions and editing for this article.

X. References

T. Agren, J. Engman, A. Frick, J. Björkstrand, E-M Larsson,T. Furmark, M. Fredrikson, Disruption of Reconsolidation Erases a Fear Memory Trace in the Human Amygdala, Science 337(6101), September 2012: 1550-1552.

J. L. Balcázar, R. Gavaldà, and H. T. Siegelmann, Computational Power of Neural Networks: A Characterization in Terms of Kolmogorov Complexity, IEEE Transactions on Information Theory 43(4), July 1997: 1175-1183.

N. Barkai and S. Leibler, Robustness in simple biochemical networks, Nature 387(6636), 1997: 913–917.

A. Ben-Hur, J. Feinberg, S. Fishman and H. T. Siegelmann, Probabilistic analysis of a differential equation for linear programming, Journal of Complexity 19(4), 2003: 474-510.

A. Ben-Hur and H. T. Siegelmann, Computation in gene networks, Chaos 14(1), 2004: 145-51.

E. Ben-Jacob, Social behavior of bacteria: from physics to complex organization, Eur. Phys. J. B 65, 2008: 315–322.

L. Blum, M. Shub, and S. Smale, On a theory of computation and complexity over the real numbers: NP completeness, recursive functions, and universal machines, Bull. A.M.S. 21, 1989: 1-46.

D. Bray. Protein molecules as computational elements in living cells. Nature, 376, July 1995: 307–312.

J. Cabessa and H.T. Siegelmann, Evolving Recurrent Neural Networks are Super-Turing, Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5 2012: 3200-3206.

J. Cabessa and H. T. Siegelmann, The Computational Power of Interactive Recurrent Neural Networks, Neural Computation, 2012, 24(4): 996-1019.

C. S. Carver and M.F. Scheier, Control theory: A useful conceptual framework for personality–social, clinical, and health psychology. Psychological Bulletin, 92(1), Jul 1982: 111-135.

A.Y. Cheung, L. Boavida, M. Aggarwal, H-M Wu, J. Feijo, The Pollen Tube Journey in the Pistil and Imaging the In Vivo process by two-Photon Microscopy. J. Exp. Bot. 61, 2010: 1907-1915.

B. J. Copeland, The essential Turing, Oxford university press, 2004.

B. J. Copeland and D. Proudfoot, Alan Turing's forgotten ideas in computer science, Scientific American, Apr 1999, 280(4): 99-103.

C. Darwin, Automatic Computing Engine (ACE), National Physical Laboratory, 17 Apr. 1946 (Public Record Office (document reference DSIR 10/385) <www.AlanTuring.net/darwin_ace>).

R. A. DiCaprio, H. Wolf, and A. Büschges, Activity-Dependent Sensitivity of Proprioceptive Sensory Neurons in the Stick Insect Femoral Chordotonal Organ, J Neurophysiol 88, November 1, 2002: 2387-2398.

S. P. Diggle, A. S. Griffin, G. S. Campbell, S. A. West, Cooperation and conflict in quorum-sensing bacterial populations, Nature 450, 15 November 2007: 411-414.

S.T. Fiske and S. E. Taylor, Social cognition (2nd ed.). New York: McGraw-Hill, 1991.

M. F. Fraga, E. Ballestar, M. F. Paz, S. Ropero, F. Setien, M. L. Ballestar, D. Heine-Suñer, J. C. Cigudosa, M. Urioste, J. Benitez, M. Boix-Chornet, A. Sanchez-Aguilera, C. Ling, E. Carlsson, P. Poulsen, A. Vaag, Z. Stephan, T. D. Spector, Y. Z. Wu, C. Plass, M. Esteller, Epigenetic differences arise during the lifetime of monozygotic twins, Proc. Natl. Acad. Sci. 102(30) July 2005: 10604-10609.

A. S. French, Two components of rapid sensory adaptation in a cockroach mechanoreceptor neuron, Journal of Neurophysiology, 62(3), 1989: 768-777.

K. Friston, The free-energy principle: a rough guide to the brain? Trends in Cognitive Sciences 13(7), 2009: 293-301.

R. Gavaldà and H.T. Siegelmann, Discontinuities in Recurrent Neural Networks, Neural Computation 11(3), 1999: 715-745.

G. Gigerenzer, P. M. Todd and the ABC Research Group, Simple heuristics that make us smart. New York, Oxford University Press, 1999.

L. Glass and M. C. Mackey, Pathological physiological conditions resulting from instabilities in physiological control systems, Ann. NY. Acad. Sci. 316, 1979: 214-235.

J. Gomez-Ramirez and R. Sanz, What the Escherichia Coli Tells Neurons about Learning, Integral Biomathics, 2012: 41-55.

D. Grimaldi, A fossil mantis (Insecta: Mantoidea) in Cretaceous amber of New Jersey, with comments on early history of Dictyoptera. American Museum Novitates 3204, 1997: 1–11.

R. M. Karp and R. J. Lipton, Some connections between nonuniform and uniform

complexity classes, Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing, 1980: 302–309.

L.F.Landweber and L.Kari, Universal molecular computation in ciliates, In Proc. of Evolution as Computation, In Natural Computing Series, Springer Verlag, L.F.Landweber, E.Winfree, Eds., 2002: 257-274.

N. Lo, G. Tokuda, H. Watanabe, H. Rose, M. Slaytor, K. Maekawa, C. Bandi and H. Noda, Evidence from multiple gene sequences indicates that termites evolved from wood-feeding cockroaches, Current Biology,10(13) June 2000: 801-804.

M. O. Magnasco, Chemical Kinetics is Turing Universal, Physical Review Letters, 78(6), 1997: 1190-3.

W. S. McCulloch and W. Pitts, A logical calculus of the ideas imminent in nervous activity," Bull. of Math. Biophysics, 5, l943: 115-133.

P. O. McGowan and M, Szyf, The epigenetics of social adversity in early life: Implications for mental health outcomes, Neurobiology of Disease 39, 2010: 66–72.

N. Mekel-Bobrov, S. L. Gilbert, P. D. Evans, E. J. Vallender, J. R. Anderson, R. R. Hudson, S. A. Tishkoff, B. T. Lahn, Ongoing Adaptive Evolution of ASPM, a Brain Size Determinant in Homo sapiens, Science, 309 (5741), 2005: 1720-1722.

H. D. Morgan, H. G. Sutherland , D. I. Martin, E. Whitelaw, Epigenetic inheritance at the agouti locus in the mouse. Nat Genet., 23(3), Nov. 1999: 314-8.

J. W. Pillow, L. Paninski, E. P. Simoncelli, Maximum Likelihood Estimation of a Stochastic Integrate-and-Fire Neural Model, Neural Computation, 16 (12), 2004: 2533-2561.

H. A. Reimann, Amyloidosis and a Periodic Disease, JAMA 185(1), 1963: 51-52.

F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, 65(6), 1958: 386–408.

S. J. Sara, Retrieval and Reconsolidation: Toward a Neurobiology of Remembering, Learning Memory 7, 2000: 73-84.

H. T. Siegelmann, Ph.D. Thesis: Foundation of Recurrent Neural Networks, Rutgers University, 1993.

H. T. Siegelmann, Computation Beyond the Turing Limit, Science 238(28), April 1995: 632-637.

H. T. Siegelmann, Neural Networks and Analog Computation: Beyond the Turing Limit, Birkhauser, Boston, December 1998.

H.T. Siegelmann, "Stochastic Analog Networks and Computational Complexity," Journal of Complexity 15(4), 1999: 451-475.

H. T. Siegelmann, A. Ben-Hur and S. Fishman, Computational Complexity for Continuous Time Dynamics, Physical Review Letters, 83(7), 1999: 1463-1466.

H. T. Siegelmann and E. D. Sontag, Analog Computation via Neural Networks, Theoretical Computer Science 131, 1994: 331-360.

H. T. Siegelmann and E. D. Sontag, Computational Power of Neural Networks, Journal of Computer System Sciences 50(1), 1995: 132-150.

N. C. Tronson and J. R. Taylor, Molecular mechanisms of memory reconsolidation, Nature Reviews Neuroscience 8, April 2007: 262-275.

[Turing 1936] A.M. Turing, On Computable Numbers, with an Application to the Entscheidungsproblem, Proceedings of the London Mathematical Society, Series 2, 42, 1936-7: 230–65.

[Turing 1939] A.M. Turing, Systems of logic based on ordinals, Proceedings of the London Mathematical Society, ser. 2, 45, 1939: 161-228.

[Turing 1947] A.M. Turing, Lecture to the London Mathematical Society on 20 February 1947, in A. M. Turing's ACE report of 1946 and other papers, B. E. Carpenter and R. W. Doran (eds.), Cambridge, Mass.: MIT Press (1986).

[Turing 1948] A. M. Turing, Intelligent Machinery, report for National Physical Laboratory, in Machine Intelligence 7, B. Meltzer and D. Michie (eds.) 1969.

[Turing 1950] A. M. Turing, Computing Machinery and Intelligence, Mind, 59, 1950: 433-460.

[Turing 1950a] A. M. Turing, Programmers' Handbook for Manchester Electronic Computer, University of Manchester Computing Laboratory, 1950). <www.AlanTuring.net/ programmers handbook>.

[Turing 1951] A. M. Turing, Intelligent machinery, a heretical theory, a lecture given to '51 Society' at Manchester.

[Turing 1951a] A.M. Turing, Can Digital Computers Think? BBC Radio 15 May 1951.

[Turing 1952] A. M. Turing, R. Braithwaite,G. Jefferson, and M. Newman, Can Automatic Calculating Machines Be Said To Think? BBC January 10th 1952.
J. von Neumann, Collected Works, Vol. 5, ed. A. H. Taub, Pergamon Press, 1963: 329-378.