Recurrent Neural Networks and Super-Turing Interactive Computation

Jérémie Cabessa and Alessandro E.P. Villa

Abstract. We present a complete overview of the computational power of recurrent neural networks involved in an interactive bio-inspired computational paradigm. More precisely, we recall the results stating that interactive rational- and real-weighted neural networks are Turing-equivalent and super-Turing, respectively. We further prove that interactive evolving neural networks are super-Turing, irrespective of whether their synaptic weights are modeled by rational or real numbers. These results show that the computational powers of neural nets involved in a classical or in an interactive computational framework follow similar patterns of characterization. They suggest that some intrinsic computational capabilities of the brain might lie beyond the scope of Turing-equivalent models of computation, hence surpass the potentialities every current standard artificial models of computation.

1 Introduction

Understanding the computational and dynamical capabilities of biological neural networks represents an issue of central importance. In this context, much interest has been focused on comparing the computational powers of diverse theoretical neural models with those of abstract computing devices. Nowadays, the computational capabilities of neural models is known to be tightly related to the nature of the activation function of the neurons, to the nature of their synaptic connections, to the eventual presence of noise in the model, to the possibility for the networks to evolve over time, and to the computational paradigm performed by the networks.

Jérémie Cabessa
Laboratory of Mathematical Economics (LEMMA), University of Paris 2 – Panthéon-Assas, 4 Rue Blaise Desgoffe, 75006 Paris, France
e-mail: jcabessa@nhrg.org

Alessandro E.P. Villa
Neuroheuristic Research Group, Faculty of Business and Economics, University of Lausanne, CH-1015 Lausanne, Switzerland
e-mail: alessandro.villa@unil.ch

P. Koprinkova-Hristova et al. (eds.), *Artificial Neural Networks*,
Springer Series in Bio-/Neuroinformatics 4, DOI: 10.1007/978-3-319-09903-3_1

This comparative approach has been initiated by McCulloch and Pitts who proposed a modeling of the nervous system as a finite interconnection of logical devices [43]. For the first time, neural networks were considered as discrete abstract machines, and the issue of their computational capabilities investigated from the automata-theoretic perspective. In this context, Kleene and Minsky proved that recurrent neural networks with Boolean activation functions are computationally equivalent to classical finite state automata [38, 44]. In Minsky's own words [44]:

> It is evident that each neural network of the kind we have been considering is a finite-state machine. [...] It is interesting and even surprising that there is a converse to this. Every finite-state machine is equivalent to, and can be simulated by, some neural net.

The simulation of finite state machine by various kinds of recurrent neural networks has further been studied for instance in [2, 27, 3, 39, 31, 48].

These foundational works opened up the way to the theoretical approach to neural computation. But the purely discrete and mechanical approach adopted by McCulloch and Pitts quickly appeared too restrictive, far from the biological reality. The neurons that they considered were too similar to classical logic gates, and the structure of the networks was too rigid to allow a biologically plausible implementation of learning.

In 1948, Turing made a step forward by showing the possibility of surpassing the capabilities of finite state machines and reaching Turing universality via neural networks called *B-type unorganized machines* [65]. The networks consisted of a specific interconnection of NAND neurons, and the consideration of infinitely many such cells could simulate the behavior of a Turing machine. The Turing universality of neural networks involving infinitely many binary neurons has further been investigated in many directions, see for instance [49, 28, 20, 19, 52].

Moreover, in the late 50's, von Neumann proposed a particularly relevant approach to the issue of information processing in the brain from the hybrid perspective of *digital* and *analog* computation [47]. He considered that the non-linear character of the operations of the brain emerges from a combination of discrete and continuous mechanisms, and therefore envisioned neural computation as something strictly more powerful than abstract machines, in line with Turing's positions.

Almost at the same time, Rosenblatt proposed the perceptron as a more general computational neural model than the McCulloch-Pitts units [51]. The essential innovation was the introduction of numerical synaptic weights and a special pattern of interconnection. This neural model gave rise to an algorithmic framework of learning achieved by adjusting the synaptic weights of the neuronal connections according to some specific task to be completed. The computational capabilities of the perceptron were further analyzed by Minsky and Papert [45]. These results represent the main achievements concerning the computational power of neural systems until the mid 90's.

In 1995, Siegelmann and Sontag proved the possibility of reaching Turing universality with finite neural networks. By considering rational synaptic weights and by extending the activation functions of the cells from Boolean to linear-sigmoid, the corresponding neural networks have their computational power drastically increased

from finite state automata up to Turing machines [60, 32, 46]. Kilian and Siegelmann then generalized the Turing universality of neural networks to a broader class of sigmoidal activation functions [37]. The computational equivalence between so-called *rational recurrent neural networks* and Turing machines has now become standard result in the field.

Moreover, following von Neumann considerations, Siegelmann and Sontag assumed that the variables appearing in the underlying chemical and physical phenomena could be modeled by continuous rather than discrete (rational) numbers, and therefore proposed a precise study of the computational power of recurrent neural networks from the perspective of *analog computation* [56]. They introduced the concept of an *analog recurrent neural network* as a classical linear-sigmoid neural net equipped with real- instead of rational-weighted synaptic connections, and proved that such analog recurrent neural networks are computationally equivalent to Turing machines with advices, hence capable of super-Turing computational power from polynomial time of computation already [59, 58]. This analog information processing model turns out to be capable of capturing non-linear dynamical properties that are most relevant to brain dynamics, such as rich chaotic behaviors [35, 62, 63], as well as dynamical and idealized chaotic systems that cannot be described by the universal Turing machine model [55]. According to these considerations, they formulated the so-called *Thesis of Analog Computation* – an analogous to the Church-Turing thesis, but in the realm of analog computation – stating that no reasonable abstract analog device can be more powerful than first-order analog recurrent neural networks [59, 55]. A proper internal hierarchical classification of analog recurrent neural networks according to the Kolmogorov complexity of their underlying real synaptic weights has further been described in [5].

Besides, central in neural computation is the issue of noise, and a natural question to be addressed concerns the robustness of the computational power of neural networks subjected to various kinds of noise. In this context, it has been shown that the presence of analog noise would generally strongly reduce the computational power of the underlying systems to that of finite state automata, or even below [40, 41, 6]. On the other hand, the incorporation of some discrete source of stochasticity would rather tend to increase or maintain the capabilities of the neural systems [57].

But the neural models considered up to that point were generally oversimplified, lacking many biological features which may be essential to the information processing in the real brain. In particular, the *evolving capabilities* of biological networks had not been taken into consideration in the studies of the computational capabilities of neural models.

In fact, it is nowadays widely admitted that biological mechanisms like synaptic plasticity, cell birth and death, changes in connectivity, etc., are intimately related to the storage and encoding of memory traces in the central nervous system, and provide the basis for most models of learning and memory in neural networks [1, 42]. More precisely, the embryonic nervous system is initially driven by genetic programs that control neural stem cell proliferation, differentiation and migration through the actions of a limited set of trophic factors and guidance cues. After a relatively short period of stable synaptic density, a pruning process

begins: synapses are constantly removed, yielding a marked decrease in synaptic density due to apoptosis – genetically programmed cell death – and selective axon pruning [34]. Overproduction of a critical mass of synapses in each cortical area may be essential for their parallel emergence through competitive interactions between extrinsic afferent projections [15]. Background activity and selected patterns of afferent activity are likely to shape deeply the emergent circuit wiring [53]. Synapses can change their strength in response to the activity of both pre- and post-synaptic cells following spike timing dependent plasticity (STDP) rules [50]. Developmental and/or learning processes are likely to potentiate or weaken certain pathways through the network by affecting the number or efficacy of synaptic interactions between the neurons [33]. Despite the plasticity of these phenomena, it is rationale to suppose that whenever the same information is presented in the network, the same pattern of activity is evoked in a circuit of functionally interconnected neurons, referred to as *cell assembly* [29]. In cell assemblies interconnected in this way, some ordered sequences of interspike intervals will recur. Such recurring, ordered, and precise (in the order of few *ms*) interspike interval relationships are referred to as spatiotemporal patterns of discharges or preferred firing sequences. Several evidence exist of spatiotemporal firing patterns in behaving animals, from rats to primates [74, 54], where preferred firing sequences can be associated to specific types of stimuli or behaviors.

In the context of AI, the consideration of such evolving neural architectures in so-called *Evolving Connectionist Systems* (ECoS) has proven to be fruitful, and significantly increased in applications in the recent years [36, 76]. From a theoretical perspective, Turova and Villa showed that neural networks with embedded spike timing-dependent plasticity are able to exhibit a sustained level of activity under special choice of parameters [66, 67]. More recently, Cabessa and Siegelmann, introduced and studied the computational capabilities of a biologically oriented neural model where the synaptic weights, the connectivity pattern, and the number of neurons can evolve rather than stay static [10]. They proved that the so-called *evolving recurrent neural networks* are super-Turing, equivalent to static analog networks, irrespective of whether their synaptic weights are modeled by rational or real numbers. Consequently, the consideration of architectural evolving capabilities in a basic neural model provides an alternative and equivalent way to the incorporation of the power of the continuum towards the achievement of super-Turing computational capabilities.

Besides, in the general context of modern computation, the classical computational approach from Turing [64] has been argued to "no longer fully corresponds to the current notion of computing in modern systems" [73] – especially when it refers to bio-inspired complex information processing systems. In the brain (or in organic life in general), information is rather processed in an interactive way [78, 23], where previous experience must affect the perception of future inputs, and where older memories may themselves change with response to new inputs.

In fact, the cerebral cortex is not a single entity, but an impressive network formed by an order of tens of millions of neurons, most of them excitatory, and by about ten times more glial cells. Ninety percent of the inputs received by a cortical area

come from other areas of the cerebral cortex. As a whole, the cerebral cortex can be viewed as a machine talking to itself, and could be seen as one big feedback system subject to the relentless advance of entropy, which subverts the exchange of messages that is essential to continued existence [79]. This concept of interdependent communications systems, also known as systems theory, coupled with Wiener's assertion that a machine that changes its responses based on feedback is a machine that learns, defines the cerebral cortex as a cybernetic machine. Therefore, the focus of investigation is shifted from communication and control to *interaction*. Systems theory has traditionally focused more on the structure of systems and their models, whereas cybernetics has focused more on how systems function, that is to say how they control their actions, how they communicate with other systems or with their own components. However, structure and function of a system cannot be understood in separation, and cybernetics and systems theory should be viewed as two facets of the neuroheuristic approach [4, 61, 75].

Following these considerations, Cabessa and Villa initiated the study of the computational power of recurrent neural networks from the perspective of *interactive computation* [12]. They proved that various kinds of Boolean recurrent neural networks involved in a reactive computational scenario are computationally equivalent to Büchi and Muller automata. From this equivalence, they deduced a transfinite hierarchical classification of Boolean recurrent neural networks based on their attractor properties [12, 8]. Cabessa and Villa also provided a description of the super-Turing computational power of analog recurrent neural networks engaged in a similar reactive computational scenario [13]. Besides, Cabessa and Siegelmann provided a characterization of the Turing and super-Turing capabilities of rational and analog recurrent neural networks involved in a more bio-inspired interactive computational paradigm [11].

Finally, Cabessa proved that neural models combining the two crucial features of *evolvability* and *interactivity* were actually capable of super-Turing computational capabilities, irrespective of whether their synaptic weights are modeled by rational or real numbers [9, 14].

In this chapter, we first review the main results concerning the Turing and super-Turing capabilities of classical and interactive recurrent neural networks, and next provide a detailed proof of these last results stated in [9, 14].

2 Preliminaries

Given some finite alphabet Σ, we let Σ^*, Σ^+, Σ^n, and Σ^ω denote respectively the sets of finite words, non-empty finite words, finite words of length n, and infinite words, all of them over alphabet Σ. We also let $\Sigma^{\leq\omega} = \Sigma^* \cup \Sigma^\omega$ be the set of all possible words (finite or infinite) over Σ. The empty word is denoted λ.

For any $x \in \Sigma^{\leq\omega}$, the *length* of x is denoted by $|x|$ and corresponds to the number of letters contained in x. If x is non-empty, we let $x(i)$ denote the $(i+1)$-th letter of x, for any $0 \leq i < |x|$. The prefix $x(0)\cdots x(i)$ of x is denoted by $x[0{:}i]$, for any $0 \leq i < |x|$. For any $x \in \Sigma^*$ and $y \in \Sigma^{\leq\omega}$, the fact that x is a *prefix* (resp. *strict prefix*) of y is denoted by $x \subseteq y$ (resp. $x \subsetneq y$). If $x \subseteq y$, we let $y - x = y(|x|)\cdots y(|y|-1)$

be the *suffix* of y that is not common to x (if $x = y$, then $y - x = \lambda$). Moreover, the *concatenation* of x and y is denoted by $x \cdot y$.

Given some sequence of finite words $\{x_i : i \in \mathbb{N}\}$ such that $x_i \subseteq x_{i+1}$ for all $i \geq 0$, one defines the *limit* of the x_i's, denoted by $\lim_{i\geq 0} x_i$, as the unique finite or infinite word which is ultimately approached by the sequence of growing prefixes $\{x_i : i \geq 0\}$. Formally, if the sequence $\{x_i : i \in \mathbb{N}\}$ is eventually constant, i.e. there exists an index $i_0 \in \mathbb{N}$ such that $x_j = x_{i_0}$ for all $j \geq i_0$, then $\lim_{i\geq 0} x_i = x_{i_0}$, meaning that $\lim_{i\geq 0} x_i$ corresponds to the smallest finite word containing each word of $\{x_i : i \in \mathbb{N}\}$ as a finite prefix; if the sequence $\{x_i : i \in \mathbb{N}\}$ is not eventually constant, then $\lim_{i\geq 0} x_i$ corresponds to the unique infinite word containing each word of $\{x_i : i \in \mathbb{N}\}$ as a finite prefix.

A function $f : \Sigma^* \to \Sigma^*$ is called *monotone* if the relation $x \subseteq y$ implies $f(x) \subseteq f(y)$, for all $x, y \in \Sigma^*$. It is called *recursive* if it can be computed by some Turing machine. Throughout this paper, any function $\varphi : \Sigma^\omega \to \Sigma^{\leq\omega}$ mapping infinite words to finite or infinite words will be referred to as an *ω-translation*.

Note that any monotone function $f : \{0,1\}^* \to \{0,1\}^*$ induces "in the limit" an ω-translation $f_\omega : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by

$$f_\omega(x) = \lim_{i\geq 0} f(x[0{:}i])$$

for all $x \in \{0,1\}^\omega$. The monotonicity of f ensures that the value $f_\omega(x)$ is well-defined for all $x \in \{0,1\}^\omega$. In words, the value $f_\omega(x)$ corresponds to the finite or infinite word that is ultimately approached by the sequence of growing prefixes $\{f(x[0{:}i]) : i \geq 0\}$.

According to these definitions, an ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ will be called *continuous*[1] if there exists a monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$; it will be called *recursive continuous*[2] if there exists a monotone and recursive (i.e. Turing computable) function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$.

3 Interactive Computation

3.1 Historical Background

Interactive computation refers to the computational framework where systems may react or interact with each other as well as with their environment during the computation [78, 23]. This paradigm was theorized in contrast to classical computation [64] which rather proceeds in a function-based transformation of a given input

[1] The choice of this name comes from the fact that continuous functions over the Cantor space $\mathscr{C} = \{0,1\}^\omega$ can be precisely characterized as limits of monotone functions. We extend this definition in the present broader context of functions from $\{0,1\}^\omega$ to $\{0,1\}^{\leq\omega}$ that can also be expressed as limits of monotone functions.

[2] Our notion of a recursive continuous ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ is a transposition to the present context of the notion of a limit-continuous function $\varphi : \{0,1\}^\omega \to \{0,1\}^\omega$ defined in [68, Definition 12] and [72, Definition 13].

to a corresponding output (closed-box and amnesic fashion), and has been argued to "no longer fully correspond to the current notions of computing in modern systems" [73]. Interactive computation also provides a particularly appropriate framework for the consideration of natural and bio-inspired complex information processing systems [69, 73, 14].

Wegner first proposed a foundational approach to interactive computation [78]. In his work, he claimed that "interaction is more powerful than algorithms", in the sense that computations performed in an interactive way are capable of handling a wider range of problems than those performed in a classical way, namely by standard algorithms and Turing machines [77, 78].

In this context, Goldin et al. introduced the concept of a *persistent Turing machine* (PTM) as a possible extension of the classical Turing machine model to the framework of interactive computation [21, 22]. A *persistent Turing machine* consists of a multi-tape machines whose inputs and outputs are given as streams of tokens generated in a dynamical and sequential manner, and whose work tape is kept preserved during the transition from one interactive step to the next. In this sense, a PTM computation is sequentially interactive and history dependent. Goldin et al. further provided a transfinite hierarchical classification of PTMs according to their expressive power, and established that PTMs are more expressive (in a precise sense) than amnesic PTMs (an extension of classical Turing machines in their context of interactive computation), and hence also than classical Turing machines [21, 22].

All these consideration led Goldin and Wegner to formulate the so-called *Sequential Interaction Thesis*, a generalization of the Church-Turing Thesis in the realm of interactive computation, claiming that "any sequential interactive computation can be performed by a persistent Turing machine" [22, 24, 25, 26]. They argue that this hypothesis, when combined with their result that PTMs are more expressive than classical TMs, provides a formal proof of Wegner's conjecture that "interaction is more powerful than algorithms" [22, 24, 25, 26], and hence refutes what they call the Strong Church-Turing Thesis – different from the original Church-Turing Thesis –, stating any possible computation can be captured by some Turing machine, or in other words, that "models of computation more expressive than TMs are impossible" [24, 26].

Driven by similar motivations, Van Leeuwen and Wiedermann proposed a slightly different interactive framework where a general component interacts with its environment by translating an incoming input stream of bits into a corresponding output stream of bit in a sequential manner [68, 72]. In their study, they restrict themselves to deterministic components, and provide mathematical characterizations of interactively computable relations, interactively recognizable sets of inputs streams, interactively generated sets of output streams, and interactively computable translations.

In this context, they introduced the concept of an *interactive Turing machine* (I-TM), a relevant translation of the classical Turing machine model in their interactive framework [69]. They further introduced the concept of *interactive Turing machine with advice* (I-TM/A) as a relevant non-uniform computational model in the

context of interactive computation [69, 70]. Interactive Turing machines with advice were proven to be strictly more powerful than interactive Turing machines without advice [70, Proposition 5] and [69, Lemma 1], and were shown to be computationally equivalent to several other non-uniform models of interactive computation, like sequences of interactive finite automata, site machines, web Turing machines [69, 70], and more recently to interactive analog neural networks and interactive evolving neural networks [9, 11, 14].

These considerations led van Leeuwen and Wiedermann to formulate an *Interactive Extension of the Church-Turing Thesis* which states that "any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice" [70].

As opposed to Goldin and Wegner, van Leeuwen and Wiedermann consider that interactivity alone is not sufficient to break the Turing barrier, and rather consists of a different instead of a more powerful paradigm than the classical computational framework [69, 71, 73]. They write [73]:

> "From the viewpoint of computability theory, interactive computing e.g. with I-TMs does not lead to super-Turing computing power. Interactive computing merely extends our view of classically computable functions over finite domains to computable functions (translations) defined over infinite domains. Interactive computers simply compute something different from non-interactive ones because they follow a different scenario."

Here, we follow this point of view and adopt a similar approach to interactive computation as presented in [68, 72].

3.2 The Interactive Paradigm

The general interactive computational paradigm consists of a step by step exchange of information between a system and its environment [68, 72]. In order to capture the unpredictability of next inputs at any time step, the dynamically generated input streams need to be modeled by potentially infinite sequences of symbols (indeed, any interactive computation over a finite input stream can a posteriori be replayed in a non-interactive way producing the same output) [78, 25, 73].

Here, we consider a basic interactive computational scenario similar to that described for instance in [72]. At every time step, the environment first sends a non-empty input bit to the system (full environment activity condition), the system next updates its current state accordingly, and then answers by either producing a corresponding output bit or remaining silent. In other words, the system is not obliged to provide corresponding output bits at every time step, but might instead stay silent for a while (to express the need of some internal computational phase before producing a new output bit), or even staying silent forever (to express the case that it has died). Consequently, after infinitely many time steps, the system will have received an infinite sequence of consecutive input bits and translated it into a corresponding finite or infinite sequence of not necessarily consecutive output bits. In the sequel, we assume that every interactive system is deterministic.

Formally, given some interactive deterministic system $\mathscr{S}$, for any infinite input stream $s \in \{0,1\}^\omega$, we define the corresponding output stream $o_s \in \{0,1\}^{\leq\omega}$ of $\mathscr{S}$ as the finite or infinite subsequence of (non-λ) output bits produced by $\mathscr{S}$ after having processed input s. The deterministic nature of $\mathscr{S}$ ensures that the output stream o_s is unique. In this way, any interactive system $\mathscr{S}$ realizes an ω-translation $\varphi_\mathscr{S} : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ defined by $\varphi_\mathscr{S}(s) = o_s$, for each $s \in \{0,1\}^\omega$.

An ω-translation ψ is then called *interactively deterministically computable*, or simply *interactively computable* iff there exists an interactive deterministic system $\mathscr{S}$ such that $\varphi_\mathscr{S} = \psi$. Note that in this definition, we do absolutely not require for the system $\mathscr{S}$ to be driven by a Turing program nor to contain any computable component of whatever kind. We simply require that $\mathscr{S}$ is deterministic and performs ω-translations in conformity with our interactive paradigm, namely in a sequential interactive manner, as precisely described above.

3.3 Interactive Computable Functions

The specific nature of the interactive computational scenario imposes strong conditions on the ω-translations that can be performed by interactive deterministic systems in general. In fact, it can be proven that any interactively computable ω-translation is necessarily continuous. This result will be used in the sequel.

Proposition 1. *Let ψ be some ω-translation. If ψ is interactively computable, then it is continuous.*

Proof. Let ψ be an interactively computable ω-translation. Then by definition, there exists a deterministic interactive system $\mathscr{S}$ such that $\varphi_\mathscr{S} = \psi$. Now, consider the function $f : \{0,1\}^* \to \{0,1\}^*$ which maps every finite word u to the unique corresponding finite word produced by $\mathscr{S}$ after exactly $|u|$ steps of computation over input stream u provided bit by bit. Note that the deterministic nature of $\mathscr{S}$ ensures that the finite word $f(u)$ is indeed unique, and thus that the function f is well-defined.

We show that f is monotone. Suppose that $u \sqsubseteq v$. It follow that $v = u \cdot (v - u)$. Hence, according to our interactive scenario, the output strings produced by $\mathscr{S}$ after $|v|$ time steps of computation over input stream v, namely $f(v)$, simply consists of the output strings produced after $|u|$ time steps of computation over input u, namely $f(u)$, followed by the output strings produced after $|v-u|$ time steps of computation over input $v-u$. Consequently, $f(u) \sqsubseteq f(v)$, and therefore f is monotone.

We now prove that the ω-translation $\varphi_\mathscr{S}$ performed by the interactive system $\mathscr{S}$ corresponds to the the "limit" (in the sense of Section 2) of the monotone function f, i.e., that $\varphi_\mathscr{S} = f_\omega$. Towards this purpose, given some infinite input stream $s \in \{0,1\}^\omega$, we consider in turn the two possible cases where $\varphi_\mathscr{S}(s)$ is either an infinite or a finite word.

First, suppose that $\varphi_\mathscr{S}(s) \in \{0,1\}^\omega$. By definition, the word $\varphi_\mathscr{S}(s)$ corresponds to the output stream produced by $\mathscr{S}$ after having processed the whole infinite input s, and, for any $i \geq 0$, the word $f(s[0{:}i])$ corresponds to the output stream produced by $\mathscr{S}$ after $i+1$ time steps of computation over the input $s[0{:}i]$. According to our interactive scenario, $f(s[0{:}i])$ is a prefix of $\varphi_\mathscr{S}(s)$, for all $i \geq 0$ (indeed, once again,

what has been produced by $\mathscr{S}$ on s after infinitely many time steps, namely $\varphi_{\mathscr{S}}(s)$, consists of what has been produced by $\mathscr{S}$ on $s[0{:}i]$ after $i+1$ time steps, namely $f(s[0{:}i])$, followed by what has been produced by $\mathscr{S}$ on $s - s[0{:}i]$ after infinitely many time steps). Moreover, since $\varphi_{\mathscr{S}}(s) \in \{0,1\}^{\omega}$, it means that the sequence of partial output strings produced by $\mathscr{S}$ on input s after i time steps of computation is not eventually constant, i.e., $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$. Hence, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathscr{S}}(s) \in \{0,1\}^{\omega}$ for all $i \geq 0$ and $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$ ensure that $\varphi_{\mathscr{S}}(s)$ is the unique infinite word containing each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix, which is to say by definition that $\varphi_{\mathscr{S}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_{\omega}(s)$.

Secondly, suppose that $\varphi_{\mathscr{S}}(s) \in \{0,1\}^{*}$. By the very same argument as in the previous case, $f(s[0{:}i])$ is a prefix of $\varphi_{\mathscr{S}}(s)$, for all $i \geq 0$. Moreover, since $\varphi_{\mathscr{S}}(s) \in \{0,1\}^{*}$, the sequence of partial output strings produced by $\mathscr{S}$ on input s after i time steps of computation must become stationary from some time step j onwards, i.e. $\lim_{i\to\infty} |f(s[0{:}i])| < \infty$. Hence, the entire finite output stream $\varphi_{\mathscr{S}}(s)$ must necessarily have been produced after a finite amount of time, and thus $\varphi_{\mathscr{S}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$. Consequently, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathscr{S}}(s) \in \{0,1\}^{*}$ for all $i \geq 0$ and $\varphi_{\mathscr{S}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$ ensure that $\varphi_{\mathscr{S}}(s)$ is the smallest finite word that contains each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix, which is to say by definition that $\varphi_{\mathscr{S}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_{\omega}(s)$. Consequently, $\varphi_{\mathscr{S}}(s) = f_{\omega}(s)$ for any $s \in \{0,1\}^{\omega}$, meaning that $\varphi_{\mathscr{S}} = f_{\omega}$.

We proved that f is a monotone function satisfying $\varphi_{\mathscr{S}} = f_{\omega}$. This means by definition that $\varphi_{\mathscr{S}}$ is continuous. Since $\varphi_{\mathscr{S}} = \psi$, it follows that ψ is also continuous. □

3.4 Interactive Turing Machines

An *interactive Turing machine* consists of an interactive abstract device driven by a standard Turing machine program. It receives an infinite stream of bits as input and produces a corresponding stream of bits as output, step by step. The input and output bits are processed via corresponding input and output ports rather than tapes. Consequently, at every time step, the machine can no more operate on the output bits that have already been processed.[3] Furthermore, according to our interactive scenario, it is assumed that, at every time step, the environment sends a non-silent input bit to the machine, and the machine either answers by producing some corresponding output bit, or rather chooses to remain silent.

Formally, a deterministic *interactive Turing machine* (I-TM) $\mathscr{M}$ is defined as a tuple $\mathscr{M} = (Q, \Gamma, \delta, q_0)$, where Q is a finite set of states, $\Gamma = \{0,1,\lambda,\sharp\}$ is the alphabet of the machine, where $\sharp$ stands for the blank tape symbol, $q_0 \in Q$ is the initial state, and

$$\delta : Q \times \Gamma \times \{0,1\} \to Q \times \Gamma \times \{\leftarrow, \rightarrow, -\} \times \{0,1,\lambda\}$$

[3] In fact, allowing the machine to erase its previous output bits would lead to the consideration of much more complicated ω-translations.

is the transition function of the machine. The relation $\delta(q,x,b) = (q',x',d,b')$ means that if the machine $\mathscr{M}$ is in state q, the cursor of the tape is scanning the letter $x \in \{0,1,\sharp\}$, and the bit $b \in \{0,1\}$ is currently received at its input port, then $\mathscr{M}$ will go in next state q', it will make the cursor overwrite symbol x by $x' \in \{0,1,\sharp\}$ and then move to direction d, and it will finally output symbol $b' \in \{0,1,\lambda\}$ at its output port, where λ represents the fact the machine is not outputting any bit at that time step. An interactive Turing machine is illustrated in Figure 1.

According to this definition, any I-TM $\mathscr{M}$ induces an ω-translation $\varphi_{\mathscr{M}} : \{0,1\}^{\omega} \to \{0,1\}^{\leq\omega}$ mapping every infinite input stream s to the corresponding finite or infinite output stream o_s produced by $\mathscr{M}$. An ω-translation $\psi : \{0,1\}^{\omega} \to \{0,1\}^{\leq\omega}$ is said to be *realizable* by some interactive Turing machine iff there exists an I-TM $\mathscr{M}$ such that $\varphi_{\mathscr{M}} = \psi$.

Van Leeuwen and Wiedermann also introduced the concept of *interactive Turing machine with advice* as a relevant non-uniform computational model in the context of interactive computation [69, 70].

Formally, a deterministic *interactive Turing machine with advice* (I-TM/A) $\mathscr{M}$ consists of an interactive Turing machine provided with an advice mechanism, which comes in the form of an advice function $\alpha : \mathbb{N} \to \{0,1\}^*$. In addition, the machine $\mathscr{M}$ uses two auxiliary special tapes, an advice input tape and an advice output tape, as well as a designated advice state. During its computation, $\mathscr{M}$ has the possibility to write the binary representation of an integer m on its advice input tape, one bit at a time. Yet at time step n, the number m is not allowed to exceed n. During the computation, if the machine happens to enter its designated advice state at some time step, then the string $\alpha(m)$ is written on the advice output tape in one time step, replacing the previous content of the tape. The machine has the possibility to repeat this process as many time as needed during its infinite computation. An interactive Turing machine with a advice is illustrated in Figure 2.

Once again, according to our interactive scenario, any I-TM/A $\mathscr{M}$ induces an ω-translation $\varphi_{\mathscr{M}} : \{0,1\}^{\omega} \to \{0,1\}^{\leq\omega}$ which maps every infinite input stream s to the corresponding finite or infinite output stream o_s produced by $\mathscr{M}$. Finally, an ω-translation $\psi : \{0,1\}^{\omega} \to \{0,1\}^{\leq\omega}$ is said to be *realizable* by some interactive Turing machine with advice iff there exists an I-TM/A $\mathscr{M}$ such that $\varphi_{\mathscr{M}} = \psi$.

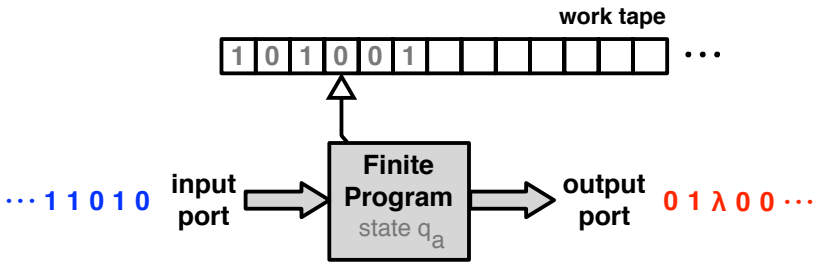

Fig. 1 An interactive Turing machine

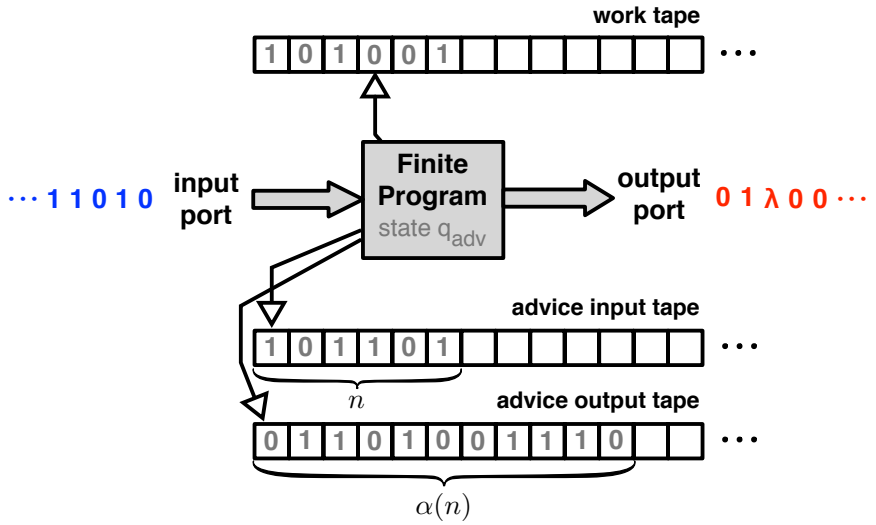

Fig. 2 An interactive Turing machine with advice

For sake of completeness, we provide a proof that I-TM/A are strictly more powerful than I-TM. Accordingly, we say that I-TM/A are *super-Turing*. The result has already been mentioned in [70, Proposition 5] and [69, Lemma 1]

Proposition 2. *I-TM/As are strictly more powerful than I-TMs.*

Proof. We prove that there exists an ω-translation ψ which is realizable by some I-TM/A, yet by no I-TM. Consider a non-Turing computable function $\alpha : \mathbb{N} \to \{0,1\}^*$. Note that such a function obviously exists since there are $2^{\aleph_0}$ (i.e. uncountably many) distinct functions of that form whereas there are only $\aleph_0$ (i.e. countably many) possible Turing machines. Consider the ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ which maps every infinite input stream s, necessarily writable of the form $s = 0^* b_0 0^+ b_1 0^+ b_2 0^+ b_3 \cdots$ where b_i's denote the blocks of 1's occurring between the 0's, to the corresponding finite or infinite word given by $\psi(s) = \alpha(|b_0|)\alpha(|b_1|)\alpha(|b_2|)\alpha(|b_3|)\cdots$ (if s has suffix 0^ω, then $\psi(s)$ is finite).

The ω-translation ψ is clearly realizable by some I-TM/A $\mathscr{M}$ with advice function α. Indeed, on every input stream $s \in \{0,1\}^\omega$, the machine $\mathscr{M}$ stores the successive blocks $b_0, b_1, b_2, \ldots$ of 1's occurring in s, and, for every such block b_i, first computes the length $|b_i|$, writes it in binary on its advice tape, then calls the advice value $\alpha(|b_i|)$ (or waits enough time steps in order to have the right to call it), and finally outputs the value $\alpha(|b_i|)$, before reiterating the procedure with respect to the next block b_{i+1}. In this way, $\mathscr{M}$ realizes ψ.

On the other hand, the ω-translation ψ is not realizable by any I-TM. Indeed, towards a contradiction, suppose it is realizable by some I-TM $\mathscr{M}$. Then, consider the classical Turing machine $\mathscr{M}'$ which, on every finite input r of the form $r = 1^k$,

proceeds exactly like $\mathscr{M}$ would have on any infinite input beginning by r, thus outputs $\alpha(k)$, and diverges on every other finite input. The existence of this classical TM $\mathscr{M}'$ shows that the function α is Turing computable, a contradiction. □

Moreover, a precise characterization of the computational powers of I-TMs and I-TM/As can be given. In fact, the I-TMs and I-TM/As realize precisely the classes of recursive continuous and continuous ω-translations, respectively. The following results are proven in [11]. Since these proofs can be easily deduced from those of previous Proposition 1 and forthcoming Lemma 1, we can include them hereafter.

Proposition 3. *Let ψ be some ω-translation.*

a) ψ is realizable by some I-TM iff ψ is recursive continuous.
b) ψ is realizable by some I-TM/A iff ψ is continuous.

Proof. The proofs of points (a) and (b) rely on previous Proposition 1 and forthcoming Lemma 1.

Point (a). Let ψ be some ω-translation realized by some I-TM $\mathscr{M}$. This means that $\psi = \varphi_{\mathscr{M}}$. Now, consider the function $f : \{0,1\}^* \to \{0,1\}^*$ which maps every finite word u to the unique corresponding finite word produced by $\mathscr{M}$ after exactly $|u|$ steps of computation over input stream u provided bit by bit. Since $\mathscr{M}$ is driven by a classical TM program, f is recursive. Moreover, by the exact same argument as in the proof of Proposition 1, f is monotone, and $f_\omega = \varphi_{\mathscr{M}} = \psi$. Consequently, ψ is recursive continuous.

Conversely, let ψ be a recursive continuous ω-translation. Then there exists some recursive monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$. Now consider the forthcoming infinite Procedure 1 (proof of Lemma 1) where the three instructions "decode $s[0{:}i]$ from x", "access to the value q_{i+1}", and "decode $f(s[0{:}i])$ from q_{i+1}" are replaced by the following one: "compute $f(s[0{:}i])$". Since f is recursive, this slightly modified version of Procedure 1 can clearly be performed by an I-TM $\mathscr{M}$. The machine $\mathscr{M}$ outputs the current word $v-u$ bit by bit every time it reaches up the instruction "output $v-u$ bit by bit", and otherwise, keeps outputting λ symbols while simulating any other internal computational steps. By the exact same argument as the one presented in the proof of Lemma 1, one has that $\varphi_{\mathscr{M}} = f_\omega = \psi$, meaning that ψ is realized by $\mathscr{M}$.

Point (b). Let ψ be some ω-translation realized by some I-TM/A $\mathscr{M}$. By definition, ψ is interactively computable. By Proposition 1, ψ is continuous.

Conversely, let ψ be a continuous ω-translation. Then there exists some monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such that $f_\omega = \psi$. First of all, we consider the function $\alpha : \mathbb{N} \to \{0,1\}^*$ which maps every integer n to the finite binary word w_n described in the beginning of the proof of forthcoming Lemma 1. Now consider the forthcoming infinite Procedure 1 (proof of Lemma 1) where the three instructions "decode $s[0{:}i]$ from x", "access to the value q_{i+1}", and "decode $f(s[0{:}i])$ from q_{i+1}" are replaced by the two following ones: "query $\alpha(i+1) = w_{i+1}$" and "extract the subword $f(s[0{:}i])$ from w_{i+1}". This slightly modified version of Procedure 1 can clearly be performed by an I-TM/A $\mathscr{M}$ with advice function α. Every time $\mathscr{M}$ encounters the instruction "query $\alpha(i+1) = w_{i+1}$", it makes an extra-recursive call

to its advice value $\alpha(i+1)$; otherwise, $\mathscr{M}$ simulates every other recursive step by means of its classical Turing program. Moreover, $\mathscr{M}$ outputs the current word $v-u$ bit by bit every time it reaches up the instruction "output $v-u$ bit by bit", and otherwise keeps outputting λ symbols while simulating any other internal computational steps. By the exact same argument as the one presented in the proof of forthcoming Lemma 1, one has that $\varphi_{\mathscr{M}} = f_\omega = \psi$, meaning that ψ is realized by $\mathscr{M}$. □

4 Interactive Recurrent Neural Networks

4.1 Recurrent Neural Networks

In this work, we consider a classical model of a first-order recurrent neural network, as presented for instance in [59, 60, 55, 56].

A *(first-order) recurrent neural network* (RNN) consists of a synchronous network of neurons (or processors) related together in a general architecture. The network contains a finite number of neurons $(x_i)_{i=1}^N$, M parallel input neurons $(u_i)_{i=1}^M$, and P designated output neurons among the N. The input and output neurons are used to transmit the information from the environment to the network or from the network to the environment, respectively. At each time step, the activation value of every neuron is updated by applying a linear-sigmoid function to some weighted affine combination of values of other neurons or inputs at previous time step.

Formally, given the activation values of the internal and input neurons $(x_j)_{j=1}^N$ and $(u_j)_{j=1}^M$ at time t, the activation value of each neuron x_i at time $t+1$ is then updated by the following equation

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

where a_{ij}, b_{ij}, and c_i are numbers describing the weighted synaptic connections and weighted bias of the network, and σ is the classical saturated-linear activation function defined by

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

Besides, Cabessa and Siegelmann introduced the model of an *evolving recurrent neural network* (Ev-RNN) as a RNN whose synaptic weights have the possibility to evolve over time rather than remaining static [10]. Formally, an *evolving recurrent neural network* (Ev-RNN) consists of an RNN whose dynamics is given by the following equation:

$$x_i(t+1) = \sigma\left(\sum_{j=1}^{N} a_{ij}(t)\cdot x_j(t) + \sum_{j=1}^{M} b_{ij}(t)\cdot u_j(t) + c_i(t)\right), \quad i = 1,\ldots,N \tag{2}$$

where $a_{ij}(t)$, $b_{ij}(t)$, and $c_i(t)$ are bounded and time-dependent synaptic weights, and σ is the classical saturated-linear activation function.

The time dependence of the synaptic weights determines the evolving capabilities of the network. The boundedness condition expresses the fact that the synaptic strengths are confined into a certain range of values imposed by the biological constitution of the neurons. It formally states that there exist an upper and a lower bound s and s' such that $a_{ij}(t), b_{ij}(t), c_i(t) \in [s, s']$ for every $t \geq 0$.

Note that this evolving neural model can describe important dynamics other than the sole synaptic plasticity. For instance, creation or deterioration of synapses can be modeled by switching the corresponding synaptic weights on or off, respectively, and cell birth and death are modeled by simultaneously switching on or off the adjacent synaptic weights of a given cell, respectively.

According to these definitions, four models of RNNs can be considered according to whether their underlying synaptic weights are either rational or real numbers of either static or evolving nature. More precisely, a network will be called *rational* if all its weights are rational numbers, and *real* if all its weights are real numbers. It will also be called *static* if all its weights remain static over time, and *evolving* if its weights are time dependent. According to these definitions, the corresponding notions of *static rational* (St-RNN[$\mathbb{Q}$]), *static real* (St-RNN[$\mathbb{R}$]), *evolving rational* (Ev-RNN[$\mathbb{Q}$]), and *evolving real* (Ev-RNN[$\mathbb{R}$]) recurrent neural networks will be employed.

Observe that since rational numbers are included in real numbers, any rational network is also a real network by definition. Moreover, since static weights are particular cases of evolving weights where the evolving patterns remain constant over time, it follows that any static network is also an evolving network. The converses of these two affirmations are obviously not true. Hence, the class of St-RNN[$\mathbb{Q}$]s corresponds precisely to the intersection of the classes of St-RNN[$\mathbb{R}$]s and Ev-RNN[$\mathbb{Q}$]s, and all the latter three classes are included in the class of Ev-RNN[$\mathbb{R}$]s, as illustrated in Figure 3.

4.2 Recurrent Neural Networks and Interactive Computation

In order to stay consistent with the interactive scenario presented in Section 3.2, we define a model of an *interactive recurrent neural network* (I-RNN) which adheres to a rigid encoding of the way inputs and outputs are interactively processed between the environment and the network. This model has already been considered in [11, 9, 14].

An *interactive recurrent neural network* (I-RNN) consists of RNN provided with a single input cell u as well as two binary output cells[4], a data cell y_d and validation cell y_v. The role of the input cell u is to transmit to the network the infinite input stream of bits sent by the environment. At each time step $t \geq 0$, the cell u admits an activation value $u(t)$ belonging to $\{0,1\}$ (the full environment activity condition

[4] The binary requirement of the output cells y_d and y_v means that the network is designed such that for every input and every time step t, one has $y_d(t) \in \{0,1\}$ and $y_v(t) \in \{0,1\}$.

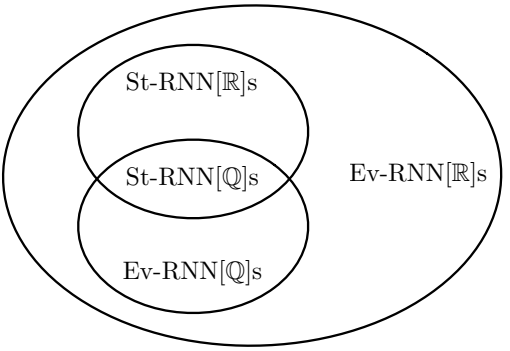

Fig. 3 Inclusion relations between the four classes of St-RNN[$\mathbb{Q}$]s, St-RNN[$\mathbb{R}$]s, Ev-RNN[$\mathbb{Q}$]s, and Ev-RNN[$\mathbb{R}$]s

forces that $u(t)$ never equals λ). Moreover, the role of the data cell y_d is to carry the output stream of the network to the environment, while the role of the validation cell y_v is to describe when the data cell is active and when it is silent. Accordingly, the output stream transmitted by the network to the environment will be defined as the (finite or infinite) subsequence of successive data bits that occur simultaneously with positive validation bits.

Formally, any I-RNN $\mathscr{N}$ will be supposed to have its initial activation values set to zero, i.e. $x_i(0) = 0$, for $i = 1, \ldots, N$. Then any infinite input stream

$$s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$$

transmitted to input cell u induces via equations (1) or (2) a corresponding pair of infinite streams transmitted by cells y_d and y_v

$$(y_d(0)y_d(1)y_d(2)\cdots, y_v(0)y_v(1)y_v(2)\cdots) \in \{0,1\}^\omega \times \{0,1\}^\omega.$$

The output stream of $\mathscr{N}$ associated to input s is then given by the finite or infinite subsequence o_s of successive data bits that occur simultaneously with positive validation bits, namely

$$o_s = \langle y_d(i) : i \in \mathbb{N} \text{ and } y_v(i) = 1 \rangle \in \{0,1\}^{\leq \omega}.$$

Hence, any I-RNN $\mathscr{N}$ naturally induces an ω-translation $\varphi_\mathscr{N} : \{0,1\}^\omega \to \{0,1\}^{\leq \omega}$ defined by $\varphi_\mathscr{N}(s) = o_s$, for each $s \in \{0,1\}^\omega$. Finally, an ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq \omega}$ is said to be *realizable* by some I-RNN iff there exists some I-RNN $\mathscr{N}$ such that $\varphi_\mathscr{N} = \psi$.

In this work, four models of I-RNNs will be considered according to whether their underlying synaptic weights are either rational or real numbers of either static or evolving nature. More precisely, the four notions of an *interactive static rational*

(I-St-RNN[$\mathbb{Q}$]), *interactive static real* (I-St-RNN[$\mathbb{R}$]), *interactive evolving rational* (I-Ev-RNN[$\mathbb{Q}$]), *interactive evolving real* (I-Ev-RNN[$\mathbb{R}$]) recurrent neural networks will be employed. An I-RNN is illustrated in Figure 4.

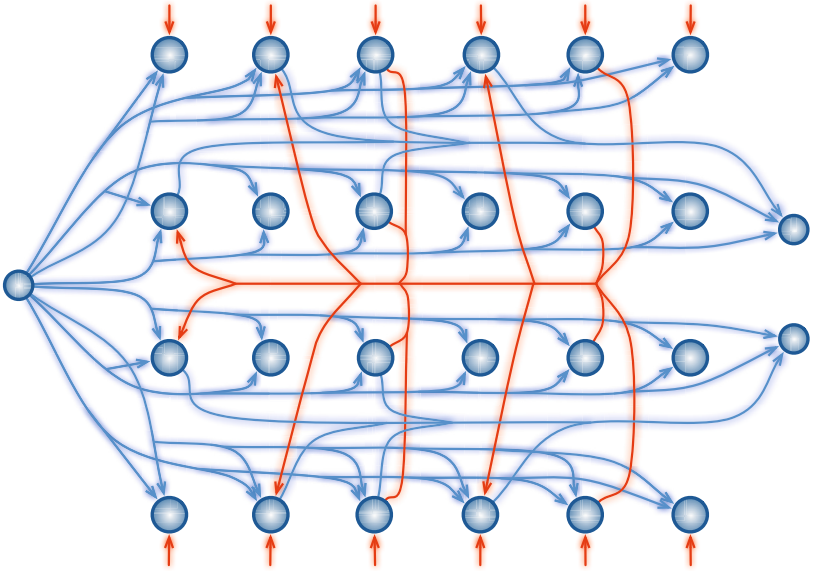

Fig. 4 An interactive recurrent neural network (I-RNN). The single neuron at the very left side represents the input cell. The two little neurons at the very right side represent the output data and validation cells. The forward and recurrent synaptic connections are represented in blue and red, respectively. The background activity connections are represented in red also. The shaded style of the synaptic connections illustrate the fact that the synaptic weights might evolve over time, in the case of an evolving interactive RNN.

5 Computational Power of Classical Neural Networks

For the sake of clarity, we recall the main results concerning the computational powers of recurrent neural networks in the case of classical (i.e. non-interactive) computation. In this context, static rational RNNs were proven to be Turing equivalent [60], whereas static real (or analog) RNNs were proven to be super-Turing [59]. Furthermore, evolving RNNs were shown to be also super-Turing, irrespective of whether their synaptic weights are modeled by rational or real numbers [10]. The three following theorems state these results in details. We now focus on particular each result.

First of all, rational-weighted RNNs were proven to be computationally equivalent to Turing machines (TMs) [60]. Indeed, on the one hand, any function determined by Equation (1) and involving rational weights is necessarily recursive, and thus can be computed by some TM. On the other hand, it was shown that any Turing

machine can be simulated in linear time by some rational RNN. The result can be expressed as follows.

Theorem 1. *St-RNN[$\mathbb{Q}$]s are Turing equivalent. More precisely, a language $L \subseteq \{0,1\}^+$ is decidable by some St-RNN[$\mathbb{Q}$] if and only if L is decidable by some TM, i.e., if and only if L is recursive.*

Secondly, real-weighted (or analog) RNNs were shown to be super-Turing, namely strictly more powerful than Turing machines, and hence also than rational RNNs. More precisely, real RNNs are capable of deciding all possible languages in exponential time of computation. When restricted to polynomial time of computation, real RNNs are computationally equivalent to Turing machines with polynomial advice[5] (TM/poly(A)), and hence decide the complexity class of languages **P/poly** [59]. Since **P/poly** strictly includes the class **P** and contains non-recursive languages, it follows that the real networks are capable of super-Turing computational power already from polynomial time of computation. Consequently, the translation from the rational- to the real-weighted context does add to the computational power of the RNNs. These results are summarized in the following theorem.

Theorem 2. *St-RNN[$\mathbb{R}$]s are super-Turing. More precisely, any language $L \subseteq \{0,1\}^+$ can be decided in exponential time by some St-RNN[$\mathbb{R}$]. Moreover, a language $L \subseteq \{0,1\}^+$ is decidable in polynomial time by some St-RNN[$\mathbb{R}$] if and only if L is decidable in polynomial time by some TM/poly(A), i.e., if and only if $L \in$ **P/poly**.*

Thirdly, evolving RNNs were shown to be also super-Turing, irrespective of whether their synaptic weights are modeled by rational or real numbers [10]. Hence, the translation from static rational to the evolving rational context does also bring up additional computational power to the networks. However, as opposed to the static context, the translation from the evolving rational to the evolving real does not increase further the capabilities of the networks.

Theorem 3. *Ev-RNN[$\mathbb{Q}$]s and Ev-RNN[$\mathbb{R}$]s are super-Turing equivalent. More precisely, any language $L \subseteq \{0,1\}^+$ can be decided in exponential time by some Ev-RNN[$\mathbb{Q}$] or by some Ev-RNN[$\mathbb{R}$]. Moreover, a language $L \subseteq \{0,1\}^+$ is decidable in polynomial time by some Ev-RNN[$\mathbb{Q}$] or by some Ev-RNN[$\mathbb{R}$] if and only if L is decidable in polynomial time by some TM/poly(A), i.e., if and only if $L \in$ **P/poly**.*

The computational capabilities of classical RNNs stated in previous theorems 1, 2, and 3 are summarized in Table 1 below.

[5] We recall that a *Turing machine with advice* (TM/A) consists of a classical Turing machine provided with an additional advice function $\alpha : \mathbb{N} \to \{0,1\}^+$ as well as an additional advice tape, and such that, on every input u of length n, the machine first copies the advice word $\alpha(n)$ on its advice tape and then continues its computation according to its finite Turing program. A *Turing machine with polynomial-bounded advice* (TM/poly(A)) consists of a TM/A whose advice length is bounded by some polynomial. The complexity classes **P** and **P/poly** represents the set of all languages decidable in polynomial time by some TM and some TM/poly(A), respectively.

Table 1 Computational power of static and evolving RNNs according to the nature of their synaptic weights

	Static RNNs	Evolving RNNs
$\mathbb{Q}$	Turing	super-Turing
$\mathbb{R}$	super-Turing	super-Turing

6 Computational Power of Interactive Static Neural Networks

Cabessa and Villa initiated the study of the computational power of RNNs involved in a reactive computational context [13]. They proved that deterministic and non-deterministic real RNNs working on infinite input streams are strictly more expressive than Turing machines equipped with Büchi or Muller conditions, respectively. More recently, Cabessa and Siegelmann studied the computational power of RNNs involved in an interactive computational framework similar the one presented here.

First, they proved that interactive rational RNNs are Turing-equivalent, and hence realize the class of recursive continuous ω-translations [11]. The results is formally expressed as follows.

Theorem 4. *I-St-RNN[$\mathbb{Q}$] are Turing-equivalent. More precisely, for any ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$, the following conditions are equivalent:*

a) ψ is realizable by some I-St-RNN[$\mathbb{Q}$];
b) ψ is realizable by some I-TM;
c) ψ is recursive continuous.

Second, they showed that interactive real RNNs are super-Turing. They are computationally equivalent to I-TM/A, and realize the class of continuous ω-translations [11]. Hence, similarly to the classical case, the translation from the rational- to the real-weighted context does bring additional computational power to the neural networks.

Theorem 5. *I-St-RNN[$\mathbb{R}$] are super-Turing. More precisely, for any ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$, the following conditions are equivalent:*

a) ψ is realizable by some I-St-RNN[$\mathbb{R}$];
b) ψ is realizable by some I-TM/A;
c) ψ is continuous.

Theorems 4 and 5 provide a generalization of theorems 1 and 2 to the interactive context. Note that the equivalences between point b and c of theorems 4 and 5 are given by Proposition 3. According to these results, for the classical as for the interactive computational framework, the translation from the static rational- to the static real-weighted context, or in other words, the incorporation of some power of continuum in the model, does bring additional capabilities to the neural networks.

7 Computational Power of Interactive Evolving Neural Networks

In this section, we prove that interactive evolving RNNs are super-Turing, irrespective of whether their synaptic weights are modeled by rational or real numbers. More precisely, both models of interactive rational and interactive real RNNs are computationally equivalent to interactive Turing machines with advice, and realize the class of continuous ω-translations. Consequently, in both classical and interactive frameworks, the translation from static rational to the evolving rational context does bring additional computational power to the networks. Once again, the translation from the evolving rational to the evolving real does not increase further the capabilities of the networks. These results provide a generalization of Theorem 3 to the context of interactive computation. They show that the power of evolution provides the possibility to break the Turing barrier of computation. A concise form of these results has already appeared in [9, 14]. The results are proven here in details.

Theorem 6. *I-Ev-RNN[$\mathbb{Q}$]s and I-Ev-RNN[$\mathbb{R}$]s are super-Turing. More precisely, for any ω-translation $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$, the following conditions are equivalent:*

a) ψ is realizable by some I-Ev-RNN[$\mathbb{Q}$];
b) ψ is realizable by some I-Ev-RNN[$\mathbb{R}$];
c) ψ is realizable by some I-TM/A;
d) ψ is continuous.

Proof. The implication "a $\to$ b" holds by definition. The three implications "a $\to$ d", "b $\to$ d", and "c $\to$ d" are given by Proposition 1. The equivalence "d $\leftrightarrow$ c" is provided by Proposition 3(a). The implication "d $\to$ a" is given by forthcoming Lemma 1. By combining all these implications and equivalences, the equivalences between points a, b, c and d are obtained. □

Lemma 1. *Let $\psi : \{0,1\}^\omega \to \{0,1\}^{\leq\omega}$ be some continuous ω-translation. Then ψ is realizable by some I-Ev-RNN[$\mathbb{Q}$].*

Proof. Let ψ be a continuous function. Then there exists some monotone function $f : \{0,1\}^* \to \{0,1\}^*$ such such that $f_\omega = \psi$. We begin by encoding all possible values of f into successive distinct rational numbers. Towards this purpose, for any $n > 0$, we let $w_{n,1}, \ldots, w_{n,2^n}$ be the lexicographical enumeration of all binary words of length n, and we let $w_n \in \{0,1,2\}^*$ be the finite word given by $w_n = 2 \cdot f(w_{n,1}) \cdot 2 \cdot f(w_{n,2}) \cdot 2 \cdots 2 \cdot f(w_{n,2^n}) \cdot 2$. Then, we consider the following rational encoding of the word w_n

$$q_n = \sum_{i=1}^{|w_n|} \frac{2 \cdot w_n(i) + 1}{6^i}.$$

Note that $q_n \in]0,1[$ for all $n > 0$. Also, the encoding procedure ensures that $q_n \neq q_{n+1}$, since $w_n \neq w_{n+1}$, for all $n > 0$. Moreover, it can be shown that the finite word w_n can be decoded from the value q_n by some Turing machine, or equivalently,

by some rational recurrent neural network [59, 60]. In this way, for any $n > 0$, the number q_n provides a rational encoding of the images by f of all words of length n.

Now, we consider the infinite Procedure 1 described below. This procedure receives as input an infinite stream $s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$ provided bit by bit, and eventually produces as output a corresponding finite or infinite stream of bits. The procedure consists of two infinite subroutines running in parallel. The first subroutine stores each input bit $s(t)$ occurring at every time step t. The second subroutine performs an infinite loop. More precisely, at stage $i+1$, the procedure considers the value $f(s[0{:}i{+}1])$. By monotonicity of f, the word $f(s[0{:}i{+}1])$ extends $f(s[0{:}i])$. If this extension is strict, the procedure output the difference word $f(s[0{:}i{+}1]) - f(s[0{:}i])$ bit by bit. Otherwise, the procedure simply outputs the empty word λ. Note that the only non-recursive instruction of Procedure 1 is "access to the value q_{i+1}".

Procedure 1.

input: infinite input stream $s = s(0)s(1)s(2)\cdots \in \{0,1\}^\omega$ provided bit by bit
initialization: $i \leftarrow 0$, $x \leftarrow \lambda$, $u \leftarrow \lambda$, $v \leftarrow \lambda$

SUBROUTINE 1:
for all $t \geq 0$ **do**
  $x \leftarrow x \cdot s(t)$ `// concatenation of the current bit s(t) to x`
end for

SUBROUTINE 2:
loop
  decode $s[0{:}i]$ from x
  access to the value q_{i+1} `// non-recursive instruction`
  decode $f(s[0{:}i])$ from q_{i+1}
  $v \leftarrow f(s[0{:}i])$
  if $u \subsetneq v$ **then**
    output $v - u$ bit by bit
  else
    output λ
  end if
  $i \leftarrow i+1$
  $u \leftarrow v$
end loop

We now show that there indeed exists some I-Ev-RNN[$\mathbb{Q}$] $\mathcal{N}$ which performs Procedure 1. The network $\mathcal{N}$ consists of one evolving and one static rational sub-network connected together. The evolving sub-network will be in charge of the execution of the only non-recursive instruction "access to the value q_{i+1}", and the static sub-network will be in charge of the execution of all other recursive instructions of Procedure 1.

More precisely, the evolving rational-weighted part of $\mathcal{N}$ is made up of a single designated processor x_e. The neuron x_e receives as sole incoming synaptic

connection a background activity of evolving intensity $c_e(t)$. The synaptic weight $c_e(t)$ successively takes the rational bounded values $q_1, q_2, q_3, \ldots$, by switching from value q_k to q_{k+1} after every N_k time steps, for some large enough $N_k > 0$ to be described. In this way, every time some new value q_{i+1} appears as a background activity of neuron x_e, the network stores it in a designated neuron in order to be able to perform the instruction "access to the value q_{i+1}" when required.

The static rational-weighted part of $\mathscr{N}$ is designed in order to perform the successive recursive steps of Procedure 1, every time some new value q_{i+1} has appeared by means of the activation value of neuron x_e. The equivalence result between rational-weighted RNNs and TMs ensures that such a static rational-weighted sub-network of $\mathscr{N}$ performing these recursive step can indeed always be constructed [59]. Moreover, for each $k > 0$, the time interval N_k between the apparition of the synaptic weights q_k and q_{k+1} is chosen large enough in order to be able to perform all the aforementioned recursive steps.

Finally, the network $\mathscr{N}$ is designed in such a way that it outputs via its data and validation cells y_d and y_v the finite word $v-u$ every time it simulates the instruction "output $v-u$ bit by bit" of Procedure 1. The network keeps outputting λ symbols every time it simulates any other internal instruction of Procedure 1.

It remains to prove that the network $\mathscr{N}$ realizes ψ, i.e. that $\varphi_{\mathscr{N}} = \psi$. Note that, for any input stream $s \in \{0,1\}^\omega$, the finite word that has been output at the end of each instruction "output $v-u$ bit by bit" corresponds precisely to the finite word $f(s[0{:}i])$ currently stored in the variable v. Hence, after infinitely many time steps, the finite or infinite word $\varphi_{\mathscr{N}}(s)$ output by $\mathscr{N}$ contains each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix. In other words, $f(s[0{:}i]) \subseteq \varphi_{\mathscr{N}}(s)$ for all $i \geq 0$.

We now consider in turn the two possible cases where $\varphi_{\mathscr{N}}(s)$ is either infinite or finite. First, if $\varphi_{\mathscr{N}}(s)$ is infinite, then it means that Procedure 1 has never stopped outputting new bits from some time step onwards, i.e., $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$. Consequently, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathscr{N}}(s) \in \{0,1\}^\omega$ for all $i \geq 0$ and $\lim_{i\to\infty} |f(s[0{:}i])| = \infty$ ensure that $\varphi_{\mathscr{N}}(s)$ is the unique infinite word containing each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix, which is to say by definition that $\varphi_{\mathscr{N}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_\omega(s)$. Second, if $\varphi_{\mathscr{N}}(s)$ is finite, it means that Procedure 1 has stopped outputting new bits from some time step onwards, and hence $\varphi_{\mathscr{N}}(s) = f(s[0{:}j])$ for some $j \geq 0$. In this case, the two properties $f(s[0{:}i]) \subseteq \varphi_{\mathscr{N}}(s) \in \{0,1\}^*$ for all $i \geq 0$ and $\varphi_{\mathscr{N}}(s) \in \{f(s[0{:}i]) : i \geq 0\}$ ensure that $\varphi_{\mathscr{N}}(s)$ is the smallest finite word that contains each word of $\{f(s[0{:}i]) : i \geq 0\}$ as a finite prefix, which is to say by definition that $\varphi_{\mathscr{N}}(s) = \lim_{i\geq 0} f(s[0{:}i]) = f_\omega(s)$.

Therefore, $\varphi_{\mathscr{N}} = f_\omega$, and since $f_\omega = \psi$, it follows that $\varphi_{\mathscr{N}} = \psi$, meaning that ψ is realized by $\mathscr{N}$. This concludes the proof. □

Finally, the computational capabilities of interactive RNNs, stated by previous theorems 4, 5, and 6 follow the same pattern as those of classical RNNs. The results are summarized in Table 2 below.

Table 2 Computational power of interactive static and evolving RNNs according to the nature of their synaptic weights

	Interactive Static RNNs	Interactive Evolving RNNs
$\mathbb{Q}$	Turing	super-Turing
$\mathbb{R}$	super-Turing	super-Turing

8 Universality

Theorems 5 and 6 together with Proposition 1 show that the four models of I-St-RNN[$\mathbb{R}$]s, I-Ev-RNN[$\mathbb{Q}$]s, I-Ev-RNN[$\mathbb{R}$]s, and I-TM/As are capable to capture all possible computations performable by some deterministic interactive system. More precisely, for any possible interactive deterministic systems $\mathscr{S}$, there exists an I-St-RNN[$\mathbb{R}$] $\mathscr{N}_1$, an I-Ev-RNN[$\mathbb{Q}$] $\mathscr{N}_2$, an I-Ev-RNN[$\mathbb{R}$] $\mathscr{N}_3$, and an I-TM/A $\mathscr{M}$ such that $\varphi_{\mathscr{N}_1} = \varphi_{\mathscr{N}_2} = \varphi_{\mathscr{N}_3} = \varphi_{\mathscr{M}} = \varphi_{\mathscr{S}}$. In this sense, those four models of interactive computation are called *universal*.

Theorem 7. *The four models of computations that are I-St-RNN[$\mathbb{R}$]s, I-Ev-RNN[$\mathbb{Q}$]s, I-Ev-RNN[$\mathbb{R}$]s, and I-TM/As, are super-Turing universal.*

Proof. Let $\mathscr{S}$ be some deterministic interactive system. By Proposition 1, $\varphi_{\mathscr{S}}$ is continuous. By Theorems 5 and 6, $\varphi_{\mathscr{S}}$ is realizable by some I-St-RNN[$\mathbb{R}$], by some I-Ev-RNN[$\mathbb{Q}$], by some I-Ev-RNN[$\mathbb{R}$], and by some I-TM/A. □

These results can be understood as follows: similarly to the classical framework, where every possible partial function from integers to integers can be computed by some Turing machine with oracle [64], in the interactive framework, every possible ω-translation performed in an interactive way can be computed by some interactive Turing machine with advice, or equivalently, by some interactive analog or evolving recurrent neural network. Alternatively put, as in the classical framework, where the model of a Turing machine with oracle exhausts the class of all possible partial functions from integers to integers, in the interactive framework, the model of an interactive Turing machine with advice or those of an interactive analog or evolving recurrent neural network also exhaust the class of all possible ω-translations performed in an interactive way.

9 Discussion

We showed that interactive rational- and real-weighted RNNs are Turing-equivalent and super-Turing, respectively (theorems 4, 5). Furthermore, interactive evolving RNNs are also super-Turing, irrespective of whether their synaptic weights are modeled by rational or real numbers (Theorem 6). The comparison between theorems 1,

2, 3 and theorems 4, 5, 6 shows that the computational powers of RNNs involved in a classical or in an interactive computational framework follow similar patterns of characterization. These results are summarized in tables 1 and 2, respectively.

These achievements show that in both classical and interactive computational framework, the translations from the static rational to the static real context, as well as from the static rational to the evolving rational context, do bring additional power to the underlying neural networks. By contrast, the two other translations from the evolving rational to the evolving real context, as well as from the static real to the evolving real context, do not increase further the capabilities of the neural networks.

Furthermore, according to theorems 1, 2, 3, 4, 5, 6, the computational capabilities of all neural models studied so far are shown to be upper bounded by those of the Turing machine with advice model. In the classical computational context, these considerations support the *Thesis of Natural Computation*, which states that every natural computational phenomenon can be captured by the Turing machine with polynomial advice model [59]. In the interactive framework, they support the *Church-Turing Thesis of Interactive Computation* which claims that "any (non-uniform interactive) computation can be described in terms of interactive Turing machines with advice" [70].

Hence, similarly to the Turing machine model which represents a definitely relevant conceptualization of current algorithmic, the interactive Turing machine with advice model also seems to encompass a particularly suitable conceptualization of brain computation, or even of natural computation in general[55, 7, 73], since it is capable to capture crucial features, like analogue considerations [60], evolvability[10, 9, 14], chaotic behaviors [63], that are impossible to be achieved via the simple Turing machine model.

The results also show that the incorporation of general *evolving capabilities* in a neuronal-based computational model naturally leads to the emergence of super-Turing computational capabilities. In fact, tables 1 and 2 show that the incorporation of either *evolving capabilities* or some *power of the continuum* in a basic neural model provides an alternative and equivalent way towards the achievement of super-Turing computational capabilities. Although being mathematically equivalent in this sense, these two features are nevertheless conceptually well distinct. While the power of the continuum is a pure conceptualisation of the mind, the evolving capabilities of the networks are, by contrast, observable in nature.

These achievements support the claim that the general mechanism of plasticity is crucially involved in the computational and dynamical capabilities of biological neural networks, and in this sense, provides a new theoretical complement to the numerous experimental studies emphasizing the importance of the general mechanism of plasticity in brain's information processing [1, 18, 30]. They further suggest that some intrinsic computational capabilities of the brain might lie beyond the scope of Turing-equivalent models of computation, and hence surpass the potentialities every current standard artificial models of computation.

Finally, we believe that the present work presents some interest far beyond the question of the existence of hypercomputational capabilities in nature [16, 17]. Comparative studies about the computational power of more and more biologically oriented neural models might ultimately bring further insight to the understanding of the intrinsic natures of biological as well as artificial intelligences. Furthermore, foundational approaches to alternative models of computation might in the long term not only lead to relevant theoretical considerations, but also to practical applications. Similarly to the theoretical work from Turing which played a crucial role in the practical realization of modern computers, further foundational considerations of alternative models of computation will certainly contribute to the emergence of novel computational technologies and computers, and step by step, open the way to the next computational era.

References

1. Abbott, L.F., Nelson, S.B.: Synaptic plasticity: taming the beast. Nat. Neurosci. 3(suppl.), 1178–1183 (2000)
2. Alon, N., Dewdney, A.K., Ott, T.J.: Efficient simulation of finite automata by neural nets. J. ACM 38(2), 495–514 (1991)
3. Alquźar, R., Alberto, S.: An algebraic framework to represent finite state machines in single-layer recurrent neural networks. Neural Computation 7(5), 931–949 (1995)
4. Arbib, M.A.: On Modelling the Nervous System. In: von Gierke, H.E., Keidel, W.D., Oestreicher, H.L. (eds.) Principles and Practice of Bionics, Proc. 44th. AGARD—Conference Brüssel, ch. 1-2, pp. 43–58. The Advisory Group for Aerospace Research and Development, NATO (1970)
5. Balcázar, J.L., Gavaldà, R., Siegelmann, H.T.: Computational power of neural networks: a characterization in terms of kolmogorov complexity. IEEE Transactions on Information Theory 43(4), 1175–1183 (1997)
6. Ben-Hur, A., Roitershtein, A., Siegelmann, H.T.: On probabilistic analog automata. Theor. Comput. Sci. 320(2-3), 449–464 (2004)
7. Bournez, O., Cosnard, M.: On the computational power of dynamical systems and hybrid systems. Theoretical Computer Science 168(2), 417–459 (1996)
8. Cabessa, J., Villa, A.E.P.: An attractor-based complexity measurement of boolean recurrent neural networks. Plos One (to appear, 2014)
9. Cabessa, J.: Interactive evolving recurrent neural networks are super-Turing. In: Filipe, J., Fred, A.L.N. (eds.) ICAART (1), pp. 328–333. SciTePress (2012)
10. Cabessa, J., Siegelmann, H.T.: Evolving recurrent neural networks are super-Turing. In: IJCNN, pp. 3200–3206. IEEE (2011)
11. Cabessa, J., Siegelmann, H.T.: The computational power of interactive recurrent neural networks. Neural Computation 24(4), 996–1019 (2012)
12. Cabessa, J., Villa, A.E.P.: A hierarchical classification of first-order recurrent neural networks. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 142–153. Springer, Heidelberg (2010)
13. Cabessa, J., Villa, A.E.P.: The expressive power of analog recurrent neural networks on infinite input streams. Theor. Comput. Sci. 436, 23–34 (2012)

14. Cabessa, J., Villa, A.E.P.: The super-Turing computational power of interactive evolving recurrent neural networks. In: Mladenov, V., Koprinkova-Hristova, P., Palm, G., Villa, A.E.P., Appollini, B., Kasabov, N. (eds.) ICANN 2013. LNCS, vol. 8131, pp. 58–65. Springer, Heidelberg (2013)
15. Chechik, G., Meilijson, I., Ruppin, E.: Neuronal regulation: A mechanism for synaptic pruning during brain maturation. Neural Comput. 11, 2061–2080 (1999)
16. Copeland, B.J.: Hypercomputation. Minds Mach. 12(4), 461–502 (2002)
17. Copeland, B.J.: Hypercomputation: philosophical issues. Theor. Comput. Sci. 317(1-3), 251–267 (2004)
18. Destexhe, A., Marder, E.: Plasticity in single neuron and circuit computations. Nature 431(7010), 789–795 (2004)
19. Franklin, S., Garzon, M.: Neural computability. In: Omidvar, O. (ed.) Progress in Neural Networks, pp. 128–144. Ablex, Norwood (1989)
20. Garzon, M., Franklin, S.: Neural computability II. In: Omidvar, O. (ed.) Proceedings of the Third International Joint Conference on Neural Networks, pp. 631–637. IEEE (1989)
21. Goldin, D.Q.: Persistent Turing machines as a model of interactive computation. In: Schewe, K.-D., Thalheim, B. (eds.) FoIKS 2000. LNCS, vol. 1762, pp. 116–135. Springer, Heidelberg (2000)
22. Goldin, D., Smolka, S.A., Attie, P.C., Sonderegger, E.L.: Turing machines, transition systems, and interaction. Inf. Comput. 194, 101–128 (2004)
23. Goldin, D., Smolka, S.A., Wegner, P.: Interactive Computation: The New Paradigm. Springer-Verlag New York, Inc., Secaucus (2006)
24. Goldin, D., Wegner, P.: The Church-Turing thesis: Breaking the myth. In: Cooper, S.B., Löwe, B., Torenvliet, L. (eds.) CiE 2005. LNCS, vol. 3526, pp. 152–168. Springer, Heidelberg (2005)
25. Goldin, D., Wegner, P.: Principles of interactive computation. In: Goldin, D., Smolka, S.A., Wegner, P. (eds.) Interactive Computation, pp. 25–37. Springer, Heidelberg (2006)
26. Goldin, D., Wegner, P.: The interactive nature of computing: Refuting the strong Church-Turing thesis. Minds Mach. 18, 17–38 (2008)
27. Goudreau, M.W., Giles, C.L., Chakradhar, S.T., Chen, D.: First-order versus second-order single-layer recurrent neural networks. IEEE Transactions on Neural Networks 5(3), 511–513 (1994)
28. Hartley, R., Szu, H.: A comparison of the computational power of neural network models. In: Butler, C. (ed.) Proceedings of the IEEE First International Conference on Neural Networks, pp. 17–22. IEEE (1987)
29. Hebb, D.O.: The organization of behavior: a neuropsychological theory. John Wiley & Sons Inc. (1949)
30. Holtmaat, A., Svoboda, K.: Experience-dependent structural synaptic plasticity in the mammalian brain. Nat. Rev. Neurosci. 10(9), 647–658 (2009)
31. Horne, B.G., Hush, D.R.: Bounds on the complexity of recurrent neural network implementations of finite state machines. Neural Networks 9(2), 243–252 (1996)
32. Hyötyniemi, H.: Turing machines are recurrent neural networks. In: Proceedings of STEP 1996, pp. 13–24. Finnish Artificial Intelligence Society (1996)
33. Iglesias, J., Villa, A.E.P.: Emergence of preferred firing sequences in large spiking neural networks during simulated neuronal development. Int. J. Neural Syst. 18(4), 267–277 (2008)

34. Innocenti, G.M., Price, D.J.: Exuberance in the development of cortical networks. Nature Rev. Neurosci. 6, 955–965 (2005)
35. Kaneko, K., Tsuda, I.: Chaotic itinerancy. Chaos 13(3), 926–936 (2003)
36. Kasabov, N.: Evolving connectionist systems - the knowledge engineering approach, 2nd edn. Springer (2007)
37. Kilian, J., Siegelmann, H.T.: The dynamic universality of sigmoidal neural networks. Inf. Comput. 128(1), 48–56 (1996)
38. Kleene, S.C.: Representation of events in nerve nets and finite automata. In: Shannon, C., McCarthy, J. (eds.) Automata Studies, pp. 3–41. Princeton University Press, Princeton (1956)
39. Kremer, S.C.: On the computational power of elman-style recurrent networks. IEEE Transactions on Neural Networks 6(4), 1000–1004 (1995)
40. Maass, W., Orponen, P.: On the effect of analog noise in discrete-time analog computations. Neural Comput. 10(5), 1071–1095 (1998)
41. Maass, W., Sontag, E.D.: Analog neural nets with gaussian or other common noise distributions cannot recognize arbitary regular languages. Neural Comput. 11(3), 771–782 (1999)
42. Martin, S.J., Grimwood, P.D., Morris, R.G.M.: Synaptic plasticity and memory: An evaluation of the hypothesis. Annu. Rev. Neurosci. 23(1), 649–711 (2000); PMID: 10845078
43. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysic 5, 115–133 (1943)
44. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Inc., Englewood Cliffs (1967)
45. Minsky, M.L., Papert, S.: Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge (1969)
46. Neto, J.P., Siegelmann, H.T., Costa, J.F., Araujo, C.P.S.: Turing universality of neural nets (revisited). In: Moreno-Díaz, R., Pichler, F. (eds.) EUROCAST 1997. LNCS, vol. 1333, pp. 361–366. Springer, Heidelberg (1997)
47. von Neumann, J.: The computer and the brain. Yale University Press, New Haven (1958)
48. Omlin, C.W., Giles, C.L.: Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants. Neural Computation 8(4), 675–696 (1996)
49. Pollack, J.B.: On Connectionist Models of Natural Language Processing. PhD thesis, Computing Reseach Laboratory, New Mexico State University, Las Cruces, NM (1987)
50. Roberts, P.D., Bell, C.C.: Spike timing dependent synaptic plasticity in biological systems. Biol. Cybern. 87, 392–403 (2002)
51. Rosenblatt, F.: The perceptron: A perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York (1957)
52. Schmidhuber, J.: Dynamische neuronale Netze und das fundamentale raumzeitliche Lernproblem (Dynamic neural nets and the fundamental spatio-temporal credit assignment problem). PhD thesis, Institut für Informatik, Technische Universität München (1990)
53. Shatz, C.J.: Impulse activity and the patterning of connections during CNS development. Neuron 5, 745–756 (1990)
54. Shmiel, T., Drori, R., Shmiel, O., Ben-Shaul, Y., Nadasdy, Z., Shemesh, M., Teicher, M., Abeles, M.: Neurons of the cerebral cortex exhibit precise interspike timing in correspondence to behavior. Proc. Natl. Acad. Sci.d U S A 102(51), 18655–18657 (2005)
55. Siegelmann, H.T.: Computation beyond the Turing limit. Science 268(5210), 545–548 (1995)

56. Siegelmann, H.T.: Neural networks and analog computation: beyond the Turing limit. Birkhauser Boston Inc., Cambridge (1999)
57. Siegelmann, H.T.: Stochastic analog networks and computational complexity. J. Complexity 15(4), 451–475 (1999)
58. Siegelmann, H.T.: Neural and super-Turing computing. Minds Mach. 13(1), 103–114 (2003)
59. Siegelmann, H.T., Sontag, E.D.: Analog computation via neural networks. Theor. Comput. Sci. 131(2), 331–360 (1994)
60. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. J. Comput. Syst. Sci. 50(1), 132–150 (1995)
61. Taylor, J.G., Villa, A.E.P.: The "Conscious I": A Neuroheuristic Approach to the Mind. In: Baltimore, D., Dulbecco, R., Jacob, F., Montalcini, R.L. (eds.) Frontiers of Life, vol. III, pp. 349–270. Academic Press (2001) ISBN: 0-12-077340-6
62. Tsuda, I.: Chaotic itinerancy as a dynamical basis of hermeneutics of brain and mind. World Futures 32, 167–185 (1991)
63. Tsuda, I.: Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. Behav. Brain Sci. 24(5), 793–847 (2001)
64. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc. 2(42), 230–265 (1936)
65. Turing, A.M.: Intelligent machinery. Technical report, National Physical Laboratory, Teddington, UK (1948)
66. Turova, T.S.: Structural phase transitions in neural networks. Math. Biosci. Eng. 11(1), 139–148 (2014)
67. Turova, T.S., Villa, A.E.P.: On a phase diagram for random neural networks with embedded spike timing dependent plasticity. Biosystems 89(1-3), 280–286 (2007)
68. van Leeuwen, J., Wiedermann, J.: On algorithms and interaction. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, pp. 99–113. Springer, Heidelberg (2000)
69. van Leeuwen, J., Wiedermann, J.: Beyond the Turing limit: Evolving interactive systems. In: Pacholski, L., Ružička, P. (eds.) SOFSEM 2001. LNCS, vol. 2234, pp. 90–109. Springer, Heidelberg (2001)
70. van Leeuwen, J., Wiedermann, J.: The Turing machine paradigm in contemporary computing. In: Engquist, B., Schmid, W. (eds.) Mathematics Unlimited - 2001 and Beyond. LNCS, pp. 1139–1155. Springer, Heidelberg (2001)
71. van Leeuwen, J., Wiedermann, J.: The emergent computational potential of evolving artificial living systems. AI Commun. 15, 205–215 (2002)
72. van Leeuwen, J., Wiedermann, J.: A theory of interactive computation. In: Goldin, D., Smolka, S.A., Wegner, P. (eds.) Interactive Computation, pp. 119–142. Springer, Heidelberg (2006)
73. Wiedermann, J., van Leeuwen, J.: How we think of computing today. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) CiE 2008. LNCS, vol. 5028, pp. 579–593. Springer, Heidelberg (2008)
74. Villa, A.E.P., Tetko, I.V., Hyland, B., Najem, A.: Spatiotemporal activity patterns of rat cortical neurons predict responses in a conditioned task. Proc. Natl. Acad. Sci. U S A 96(3), 1106–1111 (1999)
75. Villa, A.E.P.: Neural Coding in the Neuroheuristic Perspective. In: Barbieri, M. (ed.) The Codes of Life: The Rules of Macroevolution, ch. 16. Biosemiotics, vol. 1, pp. 357–377. Springer, Berlin (2008)
76. Watts, M.J.: A decade of kasabov's evolving connectionist systems: A review. IEEE Transactions on Systems, Man, and Cybernetics, Part C 39(3), 253–269 (2009)

77. Wegner, P.: Why interaction is more powerful than algorithms. Commun. ACM 40, 80–91 (1997)
78. Wegner, P.: Interactive foundations of computing. Theor. Comput. Sci. 192, 315–351 (1998)
79. Wiener, N.: Cybernetics Or Control And Communication In The Animal And The Machine. John Wiley & Sons Inc. (1948)