Journal of Computer and System Sciences ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Computational capabilities of analog and evolving neural networks over infinite input streams

Jérémie Cabessa [a,*], Olivier Finkel [b]

[a] *Laboratoire d'économie mathématique et de microéconomie appliquée (LEMMA), Université Paris 2 – Panthéon-Assas, 4 Rue Blaise Desgoffe, 75006 Paris, France*

[b] *Institut de Mathématiques de Jussieu – Paris Rive Gauche, CNRS et Université Paris Diderot, UFR de mathématiques case 7012, 75205 Paris Cedex 13, France*

ARTICLE INFO

Article history:
Received 20 March 2018
Accepted 20 November 2018
Available online xxxx

Keywords:
Recurrent neural networks
Analog computation
Infinite computation
Attractors
Turing machines
Turing machines with oracles
Super-Turing
ω-languages
Borel sets
Analytic sets

ABSTRACT

Analog and evolving recurrent neural networks are super-Turing powerful. Here, we consider analog and evolving neural nets over infinite input streams. We then characterize the topological complexity of their ω-languages as a function of the specific analog or evolving weights that they employ. As a consequence, two infinite hierarchies of classes of analog and evolving neural networks based on the complexity of their underlying weights can be derived. These results constitute an optimal refinement of the super-Turing expressive power of analog and evolving neural networks. They show that analog and evolving neural nets represent natural models for oracle-based infinite computation.

1. Introduction

Understanding the computational and dynamical capabilities of biological neural networks is an issue of major importance, with repercussions in the fields of theoretical neuroscience, bio-inspired computing, artificial intelligence, robotics and philosophy.

In this context, the theoretical approach to neural computation consists of studying the computational power of neural network models from the perspective of automata theory. The capabilities of neural networks are known to be related to the kind of activation functions used by the neurons, to the nature of their synaptic connections, to the eventual presence of noise in the model, and to the possibility for the neural architecture to evolve over time. The computational capabilities of diverse neural models have been shown to range from the finite automaton level [34,30,35,41], up to the Turing [50,39,22,45,29,25,37] or even to the super-Turing degree [44,4,42,43,8,10] (for detailed survey, see [46]).

More specifically, real-weighted neural networks, also referred to as *analog neural nets*, are strictly more powerful than Turing machines. In exponential time of computation, they can decide any possible discrete language. In polynomial time of computation, they are equivalent to Turing machines with polynomially bounded advice, and hence decide the complexity class $P/poly$ [44,42,43]. Interestingly, the super-Turing computational capabilities of analog networks can be finely

* Corresponding author.
E-mail addresses: jeremie.cabessa@u-paris2.fr (J. Cabessa), finkel@math.univ-paris-diderot.fr (O. Finkel).

https://doi.org/10.1016/j.jcss.2018.11.003
0022-0000/

characterized in terms of the Kolmogorov complexity of their underlying synaptic real weights. A proper infinite hierarchy of classes of analog neural nets with real weights of increasing Kolmogorov complexity has been obtained [4]. Besides this, it has been shown that neural networks employing time-dependent synaptic weights, called *evolving neural nets*,[1] are computationally equivalent to the analog ones. This computational equivalence holds irrespectively of whether the synaptic weights of networks are modeled by rational or real numbers and their patterns of evolution restricted to binary updates or expressed by more general form of updating [8,10].

Based on biological and computational considerations, these studies have been extended to alternative paradigms of computation where the networks process infinite rather than finite input streams [8,9,11–13,10,15,14,16,17,5,6,18]. This approach conciliates two important biological and computer scientist perspectives about neural attractor dynamics on the one hand [2] and non-terminating computational processes on the other [49,38]. The networks are provided with Boolean input and output cells carrying out the discrete exchange of information with their environment. When subjected to some infinite input stream, the outputs of the networks eventually get trapped into some attractor dynamics. The set of input streams inducing a meaningful attractor dynamics is the neural ω-language recognized by the network. The expressive power of the networks is then characterized by the topological complexity of their underlying neural ω-languages.

Within this framework, the Boolean neural networks provided with certain type specification of their attractors are computationally equivalent to Büchi or Muller automata [11,14]. As a consequence, a novel attractor-based measure of complexity for Boolean neural networks has been obtained. This complexity measure refers to the ability of the networks to perform more or less complicated classification tasks of their input streams via the manifestation of meaningful or spurious attractor dynamics.

The sigmoidal neural networks are strictly more powerful than their Boolean counterparts. The static rational-weighted neural networks are computationally equivalent to Muller Turing machines. In the deterministic and nondeterministic cases, these networks recognize the (lightface) topological classes of $BC(\Pi_2^0)$ and Σ_1^1 neural ω-languages, respectively [16,18]. By contrast, the static real-weighted (or analog) neural networks are super-Turing. In the deterministic and nondeterministic cases, they recognize the (boldface) topological classes of $BC(\mathbf{\Pi_2^0})$ and $\mathbf{\Sigma_1^1}$ neural ω-languages, respectively [5,6,16,18]. In addition, the evolving neural networks are computationally equivalent to the static analog ones. This equivalence holds irrespectively of whether the static and evolving weights of the networks are modeled by rational or real numbers, and the patterns of evolution restricted to binary updates or expressed by more general forms of updating.

In this paper, we provide an optimal refinement of these results and complete our study undertaken in [7], where only the case of evolving neural nets is treated in a more succinct way. We fully characterize the expressive power of analog and evolving networks according to the specific analog and evolving weights that they employ. Without loss of generality, we focus on analog or evolving networks using only one analog or one evolving weight, respectively. For any $\alpha \in 2^\omega$ with corresponding encoding $r_\alpha \in \mathbb{R}$, we show that deterministic and nondeterministic analog or evolving networks employing either the single static analog weight r_α or the single evolving weight α recognize the (lightface) relativized topological classes of $BC(\Pi_2^0)(\alpha)$ and $\Sigma_1^1(\alpha)$ ω-languages, respectively. As a consequence, we show the existence of two infinite refined hierarchies of classes of analog and evolving neural nets based on the complexity of their underlying analog and evolving weights. These hierarchies contain chains of length ω_1 and antichains of uncountable size.

From the point of view of theoretical computer science, these results constitute a generalization of the fundamental hierarchy of classes of analog networks based on the Kolmogorov complexity of their underlying analog weights [4]. They provide an optimal refinement of the super-Turing expressive power of analog and evolving neural networks working on infinite input streams. They also show that analog and evolving neural networks represent natural models for oracle-based infinite computation, beyond the Turing limits. From a biological point of view, these achievements may constitute a theoretical foundation of the primary role played by synaptic plasticity in the computational capabilities of neural networks [1,33,40,19].

2. Preliminaries

Given a finite set X, referred to as an *alphabet*, we let X^* and X^ω denote the sets of finite sequences (or *finite words*) and infinite sequences (or *infinite words*) of elements of X. A set $L \subseteq X^*$ or $L \subseteq X^\omega$ is called a *language* or an *ω-language*, respectively.

We assume the reader to be familiar with basic considerations about *Turing machines* (TM). A *Muller Turing machine* is a TM working on infinite words. It is defined as a pair $(\mathcal{M}, \mathcal{T})$, where $\mathcal{M}$ is a classical multitape TM whose input tape is associated with a one way read-only head, and the Muller table $\mathcal{T} = \{T_1, \ldots, T_k\}$ is a finite collection of sets of states of $\mathcal{M}$. In the deterministic (resp., non deterministic) context, an infinite word s is accepted by $(\mathcal{M}, \mathcal{T})$ if and only if the unique infinite run (resp. there exists an infinite run) of $\mathcal{M}$ on s induces (resp. which induces) a set of states that are visited infinitely often T_i which belongs to $\mathcal{T}$. The set of all infinite words accepted by $(\mathcal{M}, \mathcal{T})$ is the ω-language recognized by $(\mathcal{M}, \mathcal{T})$. For any infinite word α, a *Muller Turing machine with oracle α* is a Muller Turing machine having an additional oracle tape with α written on it.

[1] Throughout this paper, the expressions *evolving neural networks* refers to neural networks with time-dependent synaptic weights, along the lines of [10,6,18]. This expression is not to be understood in the sense of *Evolving Connectionist Systems (ECoS)* [26] nor in that of *Evolving Neural Networks through Augmenting Topologies (NEAT)* [48].

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

In the sequel, any space of the form X^ω is assumed to be equipped with the product topology of the discrete topology on X. Accordingly, the basic open sets of X^ω are of the form $p \cdot X^\omega$, for some $p \in X^*$. The general open sets are countable unions of basic open sets. In particular, the space of infinite words of bits (Cantor space) and that of infinite words of N-dimensional Boolean vectors will be denoted by $2^\omega = \{0,1\}^\omega$ and $(\mathbb{B}^N)^\omega$, respectively. They are assumed to be equipped with the above mentioned topology.

Let $(\mathcal{X}, \mathcal{T})$ be one of the above topological spaces, or a product of such spaces. The class of *Borel subsets* of $\mathcal{X}$, denoted by $\mathbf{\Delta_1^1}$ (boldface), is the σ-algebra generated by $\mathcal{T}$, i.e., the smallest collection of subsets of $\mathcal{X}$ containing all open sets and closed under countable union and complementation. For every non-null countable ordinal $\alpha < \omega_1$, where ω_1 is the first uncountable ordinal, the Borel classes $\mathbf{\Sigma_\alpha^0}$, $\mathbf{\Pi_\alpha^0}$ and $\mathbf{\Delta_\alpha^0}$ of $\mathcal{X}$ are defined as follows:

- $\mathbf{\Sigma_1^0}$ is the class of open subsets of $\mathcal{X}$ (namely $\mathcal{T}$)
- $\mathbf{\Pi_1^0}$ is the class of closed subsets of $\mathcal{X}$, i.e., that of complements of open sets
- $\mathbf{\Sigma_\alpha^0}$ is the class of countable unions of subsets of $\mathcal{X}$ in $\bigcup_{\gamma<\alpha} \mathbf{\Pi_\gamma^0}$
- $\mathbf{\Pi_\alpha^0}$ is the class of countable intersections of subsets of $\mathcal{X}$ in $\bigcup_{\gamma<\alpha} \mathbf{\Sigma_\gamma^0}$.
- $\mathbf{\Delta_\alpha^0} = \mathbf{\Sigma_\alpha^0} \cap \mathbf{\Pi_\alpha^0}$

The classes $\mathbf{\Sigma_\alpha^0}$, $\mathbf{\Pi_\alpha^0}$ and $\mathbf{\Delta_\alpha^0}$ provide a stratification of the class of Borel sets known as the *Borel hierarchy*. One has $\mathbf{\Delta_1^1} = \bigcup_{\alpha<\omega_1} \mathbf{\Sigma_\alpha^0} = \bigcup_{\alpha<\omega_1} \mathbf{\Pi_\alpha^0}$ [28]. The *rank* of a Borel set $A \subseteq \mathcal{X}$ is the smallest ordinal α such that $A \in \mathbf{\Sigma_\alpha^0} \cup \mathbf{\Pi_\alpha^0}$. It is commonly considered as a relevant measure of the topological complexity of Borel sets. The class of sets obtained as finite Boolean combinations (unions, intersections and complementations) of $\mathbf{\Pi_2^0}$-sets is denoted by $BC(\mathbf{\Pi_2^0})$.

Analytic sets are obtained as projections of either $\mathbf{\Pi_2^0}$-sets or general Borel sets [28]. More precisely, a set $A \subseteq \mathcal{X}$ is *analytic* if there exists some $\mathbf{\Pi_2^0}$-set $B \subseteq \mathcal{X} \times 2^\omega$ such that $A = \{x \in \mathcal{X} : (x,\beta) \in B,$ for some $\beta \in 2^\omega\} = \pi_1(B)$ [28]. The class of analytic sets is denoted by $\mathbf{\Sigma_1^1}$. It strictly contains that of Borel sets, i.e., $\mathbf{\Delta_1^1} \subsetneq \mathbf{\Sigma_1^1}$ [28].

The *effective* (lightface) counterpart of the Borel and analytic classes, denoted by $\Sigma_n^0, \Pi_n^0, \Delta_n^0$ as well as Δ_1^1 and Σ_1^1, are obtained by a similar effective construction, but starting from the class Σ_1^0 of effective open sets [36]. The class of finite Boolean combinations of Π_2^0-sets, denoted by $BC(\Pi_2^0)$ (lightface), and that of effective analytic sets, denoted by Σ_1^1 (lightface), correspond to the collections of ω-languages recognizable by deterministic and nondeterministic Muller Turing machines, respectively [47]. One has $BC(\Pi_2^0) \subsetneq BC(\mathbf{\Pi_2^0})$ and $\Sigma_1^1 \subsetneq \mathbf{\Sigma_1^1}$.

Any topological class Γ of the topological space $\mathcal{X}$ will also be written as $\Gamma \restriction \mathcal{X}$, whenever the underlying space $\mathcal{X}$ is needed to be specified. In addition, for any point $x \in \mathcal{X}$, we will use the notation $x \in \Gamma$ to mean that $\{x\} \in \Gamma$. Besides, any product space $\mathcal{X} \times \mathcal{Y}$ is assumed to be equipped with the product topology. If $A \subseteq \mathcal{X} \times \mathcal{Y}$ and $y \in \mathcal{Y}$, the *y-section* of A is defined by $A_y = \{x \in \mathcal{X} : (x,y) \in A\}$. For any class Γ being equal to Σ_1^0, $BC(\Pi_2^0)$, Σ_1^1, or Π_1^1 with underlying product space $\mathcal{X} \times \mathcal{Y}$ and for any $y \in \mathcal{Y}$, the *relativization of* Γ *to* y, denoted by $\Gamma(y)$, is the class of all y-sections of sets in Γ. In other words, $A \in \Gamma(y) \restriction \mathcal{X}$ if and only if there exists $B \in \Gamma \restriction \mathcal{X} \times \mathcal{Y}$ such that $A = B_y$. Moreover, we denote as usual $\Delta_1^1(y) = \Sigma_1^1(y) \cap \Pi_1^1(y)$ [36, p. 118].

For any $\alpha \in 2^\omega$, one can show that the relativized classes $BC(\Pi_2^0)(\alpha)$ and $\Sigma_1^1(\alpha)$ correspond to the collections of ω-languages recognizable by deterministic and nondeterministic Muller Turing machine with oracle α, respectively. In addition, it can be shown that $x \in \Sigma_1^0(\alpha)$ if and only if the successive letters of x can be produced step by step by some Turing machine with oracle α. Besides, one has $x \in \Sigma_1^1(\alpha)$ iff $x \in \Delta_1^1(\alpha)$, for any $\alpha \in 2^\omega$ [36].

Finally, the spaces $(\mathbb{B}^M)^\omega \times 2^\omega$ and $(\mathbb{B}^{M+1})^\omega$ are isomorphic via the natural identification. Accordingly, subsets of these spaces will be identified without it being explicitly mentioned.

3. Recurrent neural networks on infinite input streams

We consider first-order recurrent neural networks composed of Boolean input cells, Boolean output cells and sigmoidal internal cells. The sigmoidal internal neurons introduce the biological source of nonlinearity which is crucial to neural computation. They provide the possibility to surpass the capabilities of finite state automata, or even of Turing machines. The Boolean input and output cells carry out the exchange of discrete information between the network and the environment. When some infinite input stream is supplied, the output cells eventually enter into some attractor dynamics. The expressive power of the networks is related to the attractor dynamics of their Boolean output cells.

3.1. Deterministic case

A *deterministic (first-order) recurrent neural network* (D-RNN) consists of a synchronous network of neurons related together in a general architecture. It is composed of M Boolean input cells $(u_i)_{i=1}^M$, N sigmoidal internal neurons $(x_i)_{i=1}^N$, and P Boolean output cells $(y_i)_{i=1}^P$. The dynamics of the network is computed as follows: given the activation values of the input and internal neurons $(u_j)_{j=1}^M$ and $(x_j)_{j=1}^N$ at time t, the activation value of each internal and output neuron x_i and y_i at

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

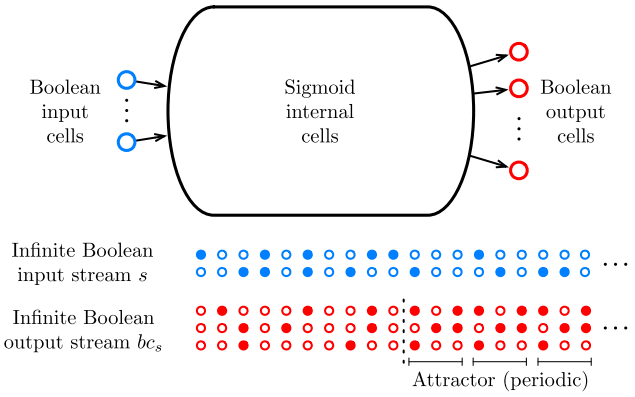

Fig. 1. Illustration of the computational process performed by some D-RNN. The infinite Boolean input stream $s = \vec{u}(0)\vec{u}(1)\vec{u}(2)\cdots \in (\mathbb{B}^M)^\omega$ induces a corresponding Boolean output stream – or Boolean computation – $bc_s = \vec{y}(0)\vec{y}(1)\vec{y}(2)\cdots \in (\mathbb{B}^P)^\omega$. The filled and empty circles represent active and quiet Boolean cells, respectively. From some time step onwards, a certain set of output states begins to repeat infinitely often, which corresponds to the attractor dynamics associated with input stream s. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

time $t+1$ is updated by the following equations, respectively:

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

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

where $a_{ij}(t)$, $b_{ij}(t)$, and $c_i(t)$ are the time dependent *synaptic weights* and *bias* of the network at time t, and σ and θ are the linear-sigmoid[2] and Heaviside step activation functions defined by

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

A synaptic weight or a bias w will be called *static* if it remains constant over time, i.e., if $w(t) = c$ for all $t \ge 0$. It will be called *bi-valued evolving* if it varies among two possible values over time, i.e., if $w(t) \in \{0,1\}$ for all $t \ge 0$. It will be called *general evolving* otherwise. A D-RNN is illustrated in Fig. 1.

According to these considerations, the dynamics of any D-RNN $\mathcal{N}$ is given by the function $f_{\mathcal{N}} : \mathbb{B}^M \times \mathbb{B}^N \to \mathbb{B}^N \times \mathbb{B}^P$ defined by

$$f_{\mathcal{N}}\left(\vec{u}(t), \vec{x}(t)\right) = \left(\vec{x}(t+1), \vec{y}(t+1)\right)$$

where the components of $\vec{x}(t+1)$ and $\vec{y}(t+1)$ are given by Equations (1) and (2), respectively.

Consider some D-RNN $\mathcal{N}$ provided with M Boolean input cells, N sigmoidal internal cells, and P Boolean output cells. For each time step $t \ge 0$, the *state* of $\mathcal{N}$ at time t consists of a pair of the form

$$\langle \vec{x}(t), \vec{y}(t) \rangle \in [0,1]^N \times \mathbb{B}^P .$$

The second element of this pair, namely $\vec{y}(t)$, is the *output state* of $\mathcal{N}$ at time t.

Assuming the initial state of the network to be $\langle \vec{x}(0), \vec{y}(0) \rangle = \langle \vec{0}, \vec{0} \rangle$, any infinite input stream

$$s = \left(\vec{u}(t)\right)_{t\in\mathbb{N}} = \vec{u}(0)\vec{u}(1)\vec{u}(2)\cdots \in \left(\mathbb{B}^M\right)^\omega$$

induces via Equations (1) and (2) an infinite sequence of consecutive states

$$c_s = \left(\langle \vec{x}(t), \vec{y}(t) \rangle\right)_{t\in\mathbb{N}} = \langle \vec{x}(0), \vec{y}(0) \rangle \langle \vec{x}(1), \vec{y}(1) \rangle \cdots \in \left([0,1]^N \times \mathbb{B}^P\right)^\omega$$

[2] The seminal results concerning the computational power of rational- and real-weighted neural networks have been obtained in this context of linear-sigmoid functions [44,45]. It has then been shown that these results remain valid for any other kind of sigmoidal activation function satisfying the properties mentioned in [29, Section 4].

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

which is the *computation* of $\mathcal{N}$ induced by s. The corresponding infinite sequence of output states

$$bc_s = \left(\vec{y}(t)\right)_{t\in\mathbb{N}} = \vec{y}(0)\vec{y}(1)\vec{y}(2)\cdots \in \left(\mathbb{B}^P\right)^\omega$$

is the *Boolean computation* of $\mathcal{N}$ induced by s. The computation of such a D-RNN is illustrated in Fig. 1.

Note that any D-RNN $\mathcal{N}$ with P Boolean output cells can only have 2^P – i.e., finitely many – possible distinct output states. Consequently, any Boolean computation bc_s necessarily consists of a finite prefix of output states followed by an infinite suffix of output states that repeat infinitely often – yet not necessarily in a periodic manner – denoted by $\inf(bc_s)$. A set of states of the form $\inf(bc_s) \subseteq \mathbb{B}^P$ will be called an *attractor* of $\mathcal{N}$ [14]. A precise definition can be given as follows:

Definition 1. Let $\mathcal{N}$ be some D-RNN. A set $A = \{\vec{y}_0, \ldots, \vec{y}_k\} \subseteq \mathbb{B}^P$ is an *attractor* for $\mathcal{N}$ if there exists some infinite input stream s such that the corresponding Boolean computation bc_s satisfies $\inf(bc_s) = A$.

In words, an attractor of $\mathcal{N}$ is a set of output states into which the Boolean computation of the network could become forever trapped – yet not necessarily in a periodic manner. An attractor of some D-RNN is illustrated in Fig. 1.

In this work, we further suppose that the networks' attractors can be of two distinct types, namely either *accepting* or *rejecting*. The classification of attractors into meaningful (accepting) or spurious (rejecting) types is an issue of significant importance in neural network studies [14]; however, it is not the subject of this work. Here, we rather consider that the type specification of the networks' attractors has already been established, e.g., according to some neurophysiological criteria or computational requirements. Hence, from this point onwards, we always assume that a D-RNN is provided with an associated classification of all of its attractors into accepting and rejecting types.

This classification of attractors leads to the following Muller-like acceptance condition: given some D-RNN $\mathcal{N}$, an infinite input stream $s \in (\mathbb{B}^M)^\omega$ is *accepted* $\mathcal{N}$ if $\inf(bc_s)$ is an accepting attractor; it is *rejected* by $\mathcal{N}$ if $\inf(bc_s)$ is a rejecting attractor. The set of all accepted input streams of $\mathcal{N}$ is the *neural ω-language recognized by* $\mathcal{N}$, denoted by $L(\mathcal{N})$. A set $L \subseteq (\mathbb{B}^M)^\omega$ is said to be *recognizable* by some D-RNN if there exists a network $\mathcal{N}$ such that $L(\mathcal{N}) = L$.

We consider six different models of D-RNNs, according to the nature of their synaptic weights:

1. The class of *deterministic static rational neural nets* refers to the D-RNNs whose all weights are static rational values. It is denoted by D-St-RNN[$\mathbb{Q}$]s.
2. The class of *deterministic static real (or analog) neural nets* refers to the D-RNNs whose all weights are static real values. It is denoted by D-St-RNN[$\mathbb{R}$]s. For the purpose of our study, we stratify this class into uncountably many subclasses, each one being defined according to some specific real weights involved in the networks. Formally, for each $r_1, \ldots, r_k \in \mathbb{R}$, the subclass of networks containing $r_1, \ldots, r_k$ as real weights[3] and all other ones being rational is denoted by D-St-RNN[$\mathbb{Q}, r_1, \ldots, r_k$]s.
3. The class of *deterministic bi-valued evolving rational neural nets* refers to the D-RNNs whose all non-static weights are bi-valued evolving and all static weight are rational. It is denoted by D-Ev$_2$-RNN[$\mathbb{Q}$]s. For each $\alpha_1, \ldots, \alpha_k \in 2^\omega$, the subclass of networks containing $\alpha_1, \ldots, \alpha_k$ as sole bi-valued evolving weights, all other ones being static rational, is denoted by D-Ev$_2$-RNN[$\mathbb{Q}, \alpha_1, \ldots, \alpha_k$]s.
4. The class of *deterministic (general) evolving rational neural nets* refers to the D-RNNs whose all static and evolving weights are rational. It is denoted by D-Ev-RNN[$\mathbb{Q}$]s.
5. The class of *deterministic bi-valued evolving real neural nets* refers to the D-RNNs whose all non-static weights are bi-valued evolving and all static weight are real. It is denoted by D-Ev$_2$-RNN[$\mathbb{R}$]s.
6. The class of *deterministic (general) evolving real neural nets* refers to the D-RNNs whose all static and evolving weights are real. It is denoted by D-Ev-RNN[$\mathbb{R}$]s.

3.2. Nondeterministic case

We also consider nondeterministic recurrent neural networks, as introduced in [44,45]. The nondeterminism is expressed by means of an external binary guess stream processed via some additional Boolean guess cell.

Formally, a *nondeterministic (first-order) recurrent neural network* (N-RNN) consists of a recurrent neural network $\mathcal{N}$ as described in previous Section 3.1, except that it contains $M+1$ Boolean input cells $(u_i)_{i=1}^{M+1}$, rather than M. The cell u_{M+1}, called the *guess cell*, carries the Boolean source of nondeterminism to be considered [44,45,12,6,18]. A N-RNN is illustrated in Fig. 2.

Given some N-RNN $\mathcal{N}$, any sequence $g = g(0)g(1)g(2)\cdots \in 2^\omega$ submitted to guess cell u_{M+1} is a *guess stream* for $\mathcal{N}$. Assuming the initial state of the network to be $\langle \vec{x}(0), \vec{y}(0)\rangle = \langle \vec{0}, \vec{0}\rangle$, any infinite input and guess streams

$$s = \left(\vec{u}(t)\right)_{t\in\mathbb{N}} \in \left(\mathbb{B}^M\right)^\omega \text{ and } g = (g(t))_{t\in\mathbb{N}} \in 2^\omega$$

[3] In this definition, the real weights $r_1, \ldots, r_k$ are not a priori required to be irrational; they could be rational weights which we wish to specify.

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

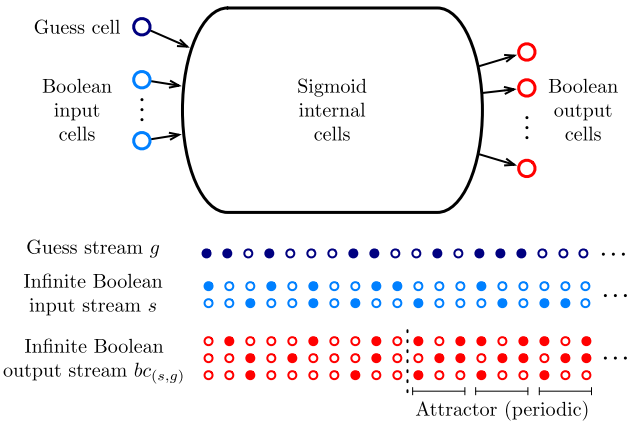

Fig. 2. Illustration of the computational process performed by some N-RNN. The infinite guess stream $g = g(0)g(1)g(2)\cdots \in 2^\omega$ together with the infinite Boolean input stream $s = \vec{u}(0)\vec{u}(1)\vec{u}(2)\cdots \in (\mathbb{B}^M)^\omega$ induce a corresponding Boolean output stream – or Boolean computation – $bc_{(s,g)} = \vec{y}(0)\vec{y}(1)\vec{y}(2)\cdots \in (\mathbb{B}^P)^\omega$. The filled and empty circles represent active and quiet Boolean cells, respectively. As in Fig. 1, the network necessarily enters into some attractor dynamics.

induce via Equations (1) and (2) two infinite sequences of states and output states

$$c_{(s,g)} = \left(\langle \vec{x}(t), \vec{y}(t)\rangle\right)_{t\in\mathbb{N}} \in \left([0,1]^N \times \mathbb{B}^P\right)^\omega$$

$$bc_{(s,g)} = \left(\vec{y}(t)\right)_{t\in\mathbb{N}} \in \left(\mathbb{B}^P\right)^\omega$$

called the *computation* and *Boolean computation* of $\mathcal{N}$ induced by (s, g), respectively. Furthermore, Definition 1 of an *attractor* remains unchanged in this case. The computation of an N-RNN is illustrated in Fig. 2.

We also assume that any N-RNN $\mathcal{N}$ is equipped with a corresponding classification of all of its attractors into accepting and rejecting types. An infinite input stream $s \in (\mathbb{B}^M)^\omega$ is *accepted* by $\mathcal{N}$ if there exists some guess stream $g \in 2^\omega$ such that $\inf(bc_{(s,g)})$ is an accepting attractor. It is *rejected* by $\mathcal{N}$ otherwise, i.e., if for all guess streams $g \in 2^\omega$, the set $\inf(bc_{(s,g)})$ is a rejecting attractor. The set of all accepted input streams is the *neural ω-language recognized by $\mathcal{N}$*, denoted by $L(\mathcal{N})$. A set $L \subseteq (\mathbb{B}^M)^\omega$ is said to be *recognizable* by some nondeterministic recurrent neural network if there exists a N-RNN $\mathcal{N}$ such that $L(\mathcal{N}) = L$.

As for the deterministic case, we consider the following classes and subclasses of N-RNNs according to the nature of their synaptic weights:

1. The class of *nondeterministic static rational neural nets* N-St-RNN[$\mathbb{Q}$]s.
2. The class of *nondeterministic static real (or analog) neural nets* N-St-RNN[$\mathbb{R}$]s. For each $r_1, \ldots, r_k \in \mathbb{R}$, we consider the corresponding subclass N-St-RNN[$\mathbb{Q}, r_1, \ldots, r_k$]s.
3. The class of *nondeterministic bi-valued evolving rational neural nets* N-Ev$_2$-RNN[$\mathbb{Q}$]s. For each $\alpha_1, \ldots, \alpha_k \in 2^\omega$, we consider the corresponding subclass N-Ev$_2$-RNN[$\mathbb{Q}, \alpha_1, \ldots, \alpha_k$]s.
4. The class of *nondeterministic (general) evolving rational neural nets* N-Ev-RNN[$\mathbb{Q}$]s.
5. The class of *nondeterministic bi-valued evolving real neural nets* N-Ev$_2$-RNN[$\mathbb{R}$]s.
6. The class of *nondeterministic (general) evolving real neural nets* N-Ev-RNN[$\mathbb{R}$]s.

4. Expressive power of neural networks

We provide a precise characterization of the expressive power of analog and evolving neural networks based on the specific analog and evolving weights that these networks employ, respectively. As a consequence, two proper hierarchies of classes of analog and evolving networks based on the complexity of their underlying weights can be obtained in Section 5.

4.1. Deterministic case

The expressive power of the classes of D-St-RNN[$\mathbb{Q}$], D-St-RNN[$\mathbb{R}$], D-Ev$_2$-RNN[$\mathbb{Q}$], D-Ev-RNN[$\mathbb{Q}$], D-Ev$_2$-RNN[$\mathbb{R}$], and D-Ev-RNN[$\mathbb{R}$] has been characterized in [18, Theorems 1, 2]. We first recall these results.

Theorem 1. *[18, Theorem 1] Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language. The following conditions are equivalent:*

(a) $L \in BC(\Pi_2^0)$;

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

(b) L is recognizable by some D-St-RNN[$\mathbb{Q}$];
(c) L is recognizable by some deterministic Muller Turing machine.

Theorem 2. *[18, Theorem 2] Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language. The following conditions are equivalent:*

(a) $L \in BC(\mathbf{\Pi_2^0})$;
(b) L is recognizable by some D-St-RNN[$\mathbb{R}$];
(c) L is recognizable by some D-Ev$_2$-RNN[$\mathbb{Q}$];
(d) L is recognizable by some D-Ev-RNN[$\mathbb{Q}$];
(e) L is recognizable by some D-Ev$_2$-RNN[$\mathbb{R}$];
(f) L is recognizable by some D-Ev-RNN[$\mathbb{R}$].

Theorem 1 states that D-St-RNN[$\mathbb{Q}$]s are Turing equivalent. Theorem 2 shows that the classes D-St-RNN[$\mathbb{R}$]s, D-Ev$_2$-RNN[$\mathbb{Q}$]s, D-Ev-RNN[$\mathbb{Q}$]s, D-Ev$_2$-RNN[$\mathbb{R}$]s and D-Ev-RNN[$\mathbb{R}$]s are computationally equivalent to each other and strictly more powerful than deterministic Muller Turing machines, since $BC(\Pi_2^0) \subsetneq BC(\mathbf{\Pi_2^0})$. In this sense, the deterministic analog and evolving neural networks are *super-Turing*. Note that the D-Ev$_2$-RNN[$\mathbb{Q}$]s achieve a maximal expressive power by recognizing the whole class of $BC(\mathbf{\Pi_2^0})$ ω-languages. Indeed, the consideration of either real synaptic weights or more complex evolving patterns in the model does actually not yield to some higher expressive power.

Remark 1. The proof of implication "$(a) \to (b)$" of Theorem 2, detailed in [18, Proposition 1], shows that any ω-language $L \in BC(\mathbf{\Pi_2^0})$ can be recognized by some D-St-RNN[$\mathbb{R}$] employing at most one static irrational weight, which is in the interval $[0, 1]$ and given in the form of a bias. Similarly, the proof of implication "$(a) \to (c)$" of Theorem 2, also detailed in [18, Proposition 1], ensures that any ω-language $L \in BC(\mathbf{\Pi_2^0})$ can be recognized by some D-Ev$_2$-RNN[$\mathbb{Q}$] using only one bi-valued evolving weight given as a bias (cf. [18, Proposition 1] again). By Theorem 2, this means that any D-St-RNN[$\mathbb{R}$] is expressively equivalent to some D-St-RNN[$\mathbb{Q}, r$], where $r \in [0, 1]$, and any D-Ev$_2$-RNN[$\mathbb{Q}$] is expressively equivalent to some D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], where $\alpha \in 2^\omega$. Hence, from this point onwards, we will focus without loss of generality on the two specific subclasses of analog or evolving networks employing only one analog or evolving weight, respectively.

We now provide a precise characterization of the expressive power of these two subclasses of D-St-RNN[$\mathbb{Q}, r$] and D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], for any $r \in [0, 1]$ and $\alpha \in 2^\omega$, respectively. This result constitutes a significant refinement of Theorem 2. It is obtained via forthcoming Propositions 1, 2, 3 and 4.

Proposition 1. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $\alpha \in 2^\omega$. If $L \in BC(\Pi_2^0)(\alpha)$, then L is recognizable by some D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$].*

Proof. If $L \in BC(\Pi_2^0)(\alpha) \restriction (\mathbb{B}^M)^\omega$, then by definition, there exists $L' \in BC(\Pi_2^0) \restriction (\mathbb{B}^{M+1})^\omega$ such that

$$L = L'_\alpha = \left\{ s \in (\mathbb{B}^M)^\omega : (s, \alpha) \in L' \right\}.$$

Hence, Theorem 1 ensures that there exists a D-St-RNN[$\mathbb{Q}$] $\mathcal{N}'$ with $M + 1$ input cells $u_1, \ldots, u_{M+1}$ such that $L(\mathcal{N}') = L'$.

Now, consider the D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$] $\mathcal{N}$ which consists in a slight modification of the D-St-RNN[$\mathbb{Q}$] $\mathcal{N}'$. More precisely, $\mathcal{N}$ contains the same cells and synaptic connections as $\mathcal{N}'$, it admits $u_1, \ldots, u_M$ as its input cells, and the cell u_{M+1} is transformed into an internal cell receiving the bi-valued evolving weight $\alpha \in 2^\omega$ in the form of a bias. In addition, the attractors of $\mathcal{N}$ are the same as those of $\mathcal{N}'$. By construction, for any input $s \in (\mathbb{B}^M)^\omega$, $\mathcal{N}$ receives the bi-valued evolving weight α as a bias and works precisely like $\mathcal{N}'$ on input $(s, \alpha) \in (\mathbb{B}^{M+1})^\omega$. Consequently, $s \in L(\mathcal{N})$ if and only if $(s, \alpha) \in L(\mathcal{N}') = L'$. Therefore, $L(\mathcal{N}) = L'_\alpha = L$. This shows that L is recognized by the D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$] $\mathcal{N}$. □

Proposition 2. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $\alpha = \alpha_1\alpha_2\alpha_3 \cdots \in 2^\omega$. If $L \in BC(\Pi_2^0)(\alpha)$, then L is recognizable by some D-St-RNN[$\mathbb{Q}, r_\alpha$] $\mathcal{N}$, where $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$.*

Proof. Suppose that $L \in BC(\Pi_2^0)(\alpha)$. Then L is recognized by some deterministic Muller Turing machine $\mathcal{M}$ with oracle α. Let

$$\alpha' = 00 \prod_{i=1}^{\infty} (0\alpha_i) = 000\alpha_1 0\alpha_2 0\alpha_3 0\alpha_4 0 \cdots \in 2^\omega.$$

Clearly, the successive letters α_i's of α can be produced by some Turing machine with oracle α', i.e., $\alpha \in \Sigma_0^1(\alpha')$. Consequently, L is also recognized by the deterministic Muller Turing machine with oracle α' which retrieves step by step the successive letters of α from its oracle α', and concomitantly, simulates the behavior of $\mathcal{M}$ with oracle α. This means that $L \in BC(\Pi_2^0)(\alpha')$. Hence, there exists $L' \in BC(\Pi_2^0) \restriction (\mathbb{B}^{M+1})^\omega$ such that

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

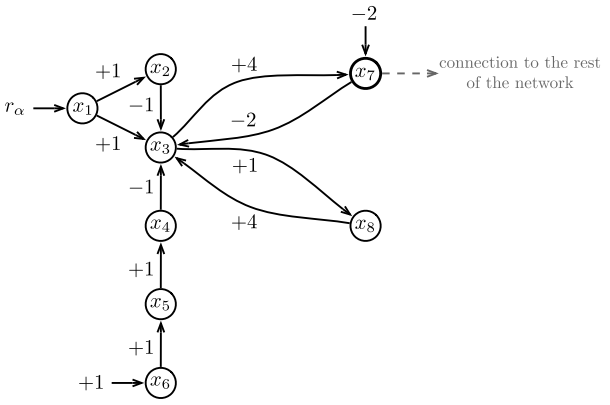

Fig. 3. Circuit C: nodes represent sigmoidal neurons and labelled edges are weighted synaptic connections between those. Cell x_1 receives r_α as bias and cell x_7 outputs the successive bits of $\alpha' = 000\alpha_1 0\alpha_2 0\alpha_3 0\cdots$. In order to understand this circuit, the following notions need to be recalled [45]. For any $\gamma = \gamma_1\gamma_2\gamma_3\cdots \in 2^\omega$, we suppose that γ is a stack whose elements from top to bottom are $\gamma_1, \gamma_2, \gamma_3, \ldots$. We further assume that γ is encoded by the real number $r_\gamma = \sum_{i=1}^{\infty} \frac{2\gamma_i+1}{4^i} \in [0,1]$. By definition of r_γ, the top element γ_1 of γ is given by $top(\gamma) = \sigma(4r_\gamma - 2)$. In addition, the encoding of the stack $\gamma_2\gamma_3\gamma_4\cdots$, which corresponds to the stack γ whose top element has been popped, is given by $pop(\gamma) = \sigma(4r_\gamma - 2top(\gamma) - 1)$. The design of circuit C is based on these considerations. Cell x_3 receives from x_1 a permanent activity of intensity $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ from time 2 onwards. But this activity is neutralized from time 3 onwards, due to the activity coming from x_2. Hence, x_3 holds activation value $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ at time 2 only. Next, x_7 computes $top(r_\alpha) = \sigma(4r_\alpha - 2) = \alpha_1$ at time 3, and thanks to the chain of cells x_6, x_5 and x_4 which brings an activity of intensity -1 to x_3, the later cell computes $pop(r_\alpha) = \sigma(4r_\alpha - 2top(\alpha) - 1)$ at time 4. Afterwards, x_7 computes $top(pop(r_\alpha)) = \alpha_2$ at time 5, and x_3 computes $pop(pop(r_\alpha)) = pop^2(r_\alpha)$ at time 6. And so on ad infinitum. Hence, x_7 outputs $top(pop^i(r_\alpha)) = \alpha_{i+1}$ at successive time steps $2i+3$, for all $i \in \mathbb{N}$, and it outputs 0 at any other time step. In other words, x_7 outputs the successive bits of $\alpha' = 000\alpha_1 0\alpha_2 0\alpha_3 0\cdots$ at successive time steps $0, 1, 2, \ldots$.

$$L = L'_{\alpha'} = \left\{ s \in (\mathbb{B}^M)^\omega : (s, \alpha') \in L' \right\}.$$

By Theorem 1, there exists a D-St-RNN[$\mathbb{Q}$] $\mathcal{N}'$ with $M+1$ input cells $u_1, \ldots, u_{M+1}$ such that $L(\mathcal{N}') = L'$.

Now, consider the real encoding of α given by $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0,1]$. Consider also the D-St-RNN[$\mathbb{Q}, r_\alpha$] $\mathcal{N}$ obtained by replacing the input cell u_{M+1} of $\mathcal{N}'$ by the real-weighted neural circuit C with bias r_α depicted in Fig. 3. Circuit C is designed in such a way that it outputs the successive bits of α' at each successive time step (see Fig. 3 for further details of this decoding procedure). By construction, for any $s \in (\mathbb{B}^M)^\omega$, the behavior of $\mathcal{N}$ on input s is the same as that of $\mathcal{N}'$ on input (s, α'). In other words, $s \in L(\mathcal{N})$ if and only if $(s, \alpha') \in L(\mathcal{N}') = L'$. Therefore, $L(\mathcal{N}) = L'_{\alpha'} = L$. This shows that L is recognized by the D-St-RNN[$\mathbb{Q}, r_\alpha$] $\mathcal{N}$. □

Proposition 3. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $\alpha \in 2^\omega$. If L is recognizable by some D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], then $L \in BC(\Pi_2^0)(\alpha)$.*

Proof. Let $\mathcal{N}$ be a D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$] such that $L(\mathcal{N}) = L$. By Remark 1, we may assume without loss generality that the bi-valued evolving weight α of $\mathcal{N}$ is a bias related to some cell x. Let $\mathcal{N}'$ be the D-St-RNN[$\mathbb{Q}$] obtained by replacing in $\mathcal{N}$ the cell x and its associated bias by a new input cell u_{M+1}. Network $\mathcal{N}'$ is a D-St-RNN[$\mathbb{Q}$] with $M+1$ input cells, and Theorem 1 ensures that $L(\mathcal{N}') \in BC(\Pi_2^0)$. By construction, for any $(s, \alpha) \in (\mathbb{B}^{M+1})^\omega$, the behavior of $\mathcal{N}'$ on input (s, α) is the same as that of $\mathcal{N}$ on input $s \in (\mathbb{B}^M)^\omega$. In other words, $(s, \alpha) \in L(\mathcal{N}')$ if and only if $s \in L(\mathcal{N})$. Thus $L(\mathcal{N}) = L(\mathcal{N}')_\alpha$. Since $L(\mathcal{N}') \in BC(\Pi_2^0)$, it follows that $L(\mathcal{N}) \in BC(\Pi_2^0)(\alpha)$. □

Proposition 4. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $r \in [0,1]$. If L is recognizable by some D-St-RNN[$\mathbb{Q}, r$], then $L \in BC(\Pi_2^0)(\alpha)$, for some $\alpha \in 2^\omega$. In particular, if L is recognizable by some D-St-RNN[$\mathbb{Q}, r_\alpha$], where $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ and $\alpha_i \in \{0,1\}$ for each $i \in \mathbb{N}^*$, then $L \in BC(\Pi_2^0)(\alpha)$, where $\alpha = \alpha_1\alpha_2\alpha_3\cdots$.*

Proof. If L is recognized by some D-St-RNN[$\mathbb{Q}, r$], then a fortiori L is recognized by some D-St-RNN[$\mathbb{R}$]. By Theorem 2, $L \in BC(\mathbf{\Pi_2^0})$. By Theorem 2 again, L is recognized by some D-Ev$_2$-RNN[$\mathbb{Q}$], and by Remark 1, L is recognized by some D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], for some $\alpha \in 2^\omega$. By Proposition 3, $L \in BC(\Pi_2^0)(\alpha)$.

Now, suppose that L is recognized by some D-St-RNN[$\mathbb{Q}, r_\alpha$] $\mathcal{N}$, where $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ and $\alpha_i \in \{0,1\}$, for each $i \in \mathbb{N}^*$. By Remark 1, we may assume without loss of generality that the static weight r_α of $\mathcal{N}$ is a bias. Let $r_\alpha|_K$ denote the truncation of r_α after K bits, i.e.,

$$r_\alpha|_K = \sum_{i=1}^{K} \frac{2\alpha_i + 1}{4^i}.$$

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

Algorithm 1 Infinite procedure.

Require: 1. input $s = \vec{u}(0)\vec{u}(1)\vec{u}(2)\cdots \in (\mathbb{B}^M)^\omega$ supplied step by step at successive time steps $t = 0, 1, 2, \ldots$
2. infinite word $\alpha = \alpha_1\alpha_2\alpha_3\cdots \in 2^\omega$ supplied step by step at successive time steps $t = 0, 1, 2, \ldots$

1: **SUBROUTINE 1**
2: **for all** time step $t \geq 0$ **do**
3: store the incoming Boolean vector $\vec{u}(t) \in \mathbb{B}^M$
4: store the incoming bit $\alpha_{t+1} \in \{0, 1\}$
5: **end for**
6: **END SUBROUTINE 1**

7: **SUBROUTINE 2**
8: **for** $n = 0, 1, 2, 3, \ldots$ **do**
9: wait that $K \cdot n$ bits of α have been stored
10: compute $r_\alpha|_{K \cdot n}$ `// recursive if α given bit by bit`
11: simulate the computation of the truncated network $\mathcal{N}|_{K \cdot n}$ working on input prefix $\vec{u}(0)\cdots\vec{u}(n)$, but output the result of that computation only for the last time step n `// recursive, since` $\mathcal{N}|_{K \cdot n}$ `is a D-St-RNN[ℚ]` [45]
12: **end for**
13: **END SUBROUTINE 2**

For each $n \geq 0$, let $\mathcal{N}|_{K \cdot n}$ be the network $\mathcal{N}$ whose weight r_α has been replaced by $r_\alpha|_{K \cdot n}$, for some constant $K > 0$ defined in [44, Lemma 4.1]. By [44, Lemma 4.1], the truncated network $\mathcal{N}|_{K \cdot n}$ computes precisely like $\mathcal{N}$ up to time step n. Moreover, $\mathcal{N}|_{K \cdot n}$ is a D-St-RNN[$\mathbb{Q}$], and thus, its behavior can be simulated by some Turing machine [45].

Consider the infinite procedure given by Algorithm 1. The procedure consists in two subroutines performed in parallel. It receives some input $s \in (\mathbb{B}^M)^\omega$ together with the infinite word $\alpha \in 2^\omega$, and it simulates the computation of $\mathcal{N}$ working on input s, by using the successive truncated networks $\mathcal{N}|_{K \cdot n}$. All instructions of Algorithm 1 are recursive, and thus, can be simulated by some D-St-RNN[$\mathbb{Q}$] [45]. Hence, the whole Algorithm 1 can be simulated by some D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$] $\mathcal{N}'$ which receives $\alpha = \alpha_1\alpha_2\alpha_3$ as an evolving bias. Every time $\mathcal{N}'$ enters instruction 11 of Algorithm 1, it simulates the behavior of the truncated network $\mathcal{N}|_{K \cdot n}$, and thus, by [44, Lemma 4.1], reproduces the output pattern of $\mathcal{N}$ working on input prefix $\vec{u}(0)\cdots\vec{u}(n)$, to finally release the last output state of $\mathcal{N}$ at last time step n. But these successive computational periods of $\mathcal{N}'$ are interspersed with delays due to the simulation of the other instructions of Algorithm 1. In order to deal with these delays, we provide $\mathcal{N}'$ with an additional output cell y_{P+1} which is programmed to be active only when the network simulates the output period of instruction 11. Then, an attractor $A \subseteq \mathbb{B}^{P+1}$ of $\mathcal{N}'$ is defined to be accepting if and only if the $(P+1)$-th component of each element of A equals 1 (which corresponds to the cell y_{P+1} being active), and the projection of A on $\mathbb{B}^P$ is an accepting attractor of $\mathcal{N}$.

In this way, for any input $s \in (\mathbb{B}^M)^\omega$, the subsequence of the Boolean computation of $\mathcal{N}'$ induced by the active states of y_{P+1} is the same as the Boolean computation of $\mathcal{N}$, and hence, s is accepting for $\mathcal{N}'$ if and only if s is accepting for $\mathcal{N}$. Consequently, $L(\mathcal{N}') = L(\mathcal{N})$. Since $\mathcal{N}'$ is a D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], Proposition 3 ensures that $L(\mathcal{N}') \in BC(\Pi_2^0)(\alpha)$. Therefore, $L(\mathcal{N}) \in BC(\Pi_2^0)(\alpha)$ too. □

Propositions 1, 2, 3 and 4 lead to the following theorem:

Theorem 3. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language, $\alpha = \alpha_1\alpha_2\alpha_3\cdots \in 2^\omega$ and $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$. The following conditions are equivalent:*

(a) $L \in BC(\Pi_2^0)(\alpha)$;
(b) L is recognizable by some D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$];
(c) L is recognizable by some D-St-RNN[$\mathbb{Q}, r_\alpha$].

From Theorem 3 and Remark 1, the following set-theoretical result can be retrieved:

$$BC(\mathbf{\Pi_2^0}) = \bigcup_{\alpha \in 2^\omega} BC(\Pi_2^0)(\alpha).$$

Indeed, $L \in BC(\mathbf{\Pi_2^0})$ if and only if, by Remark 1, L is recognizable by some D-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], for some $\alpha \in 2^\omega$, if and only if, by Theorem 3, $L \in BC(\Pi_2^0)(\alpha)$. In words, the relativized classes $BC(\Pi_2^0)(\alpha)$ span the class $BC(\mathbf{\Pi_2^0})$, when α varies over 2^ω.

4.2. Nondeterministic case

The expressive power of the classes of N-St-RNN[$\mathbb{Q}$], N-Ev$_2$-RNN[$\mathbb{Q}$], N-Ev-RNN[$\mathbb{Q}$], N-Ev$_2$-RNN[$\mathbb{R}$] and N-Ev-RNN[$\mathbb{R}$] has been established in [6, Theorems 1, 2]. We have the following results:

Theorem 4. *[6, Theorems 1] Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language. The following conditions are equivalent:*

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

(a) $L \in \Sigma_1^1$;
(b) *L is recognizable by some N-St-RNN*$[\mathbb{Q}]$;
(c) *L is recognizable by some nondeterministic Muller Turing machine.*

Theorem 5. *[6, Theorems 2] Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language. The following conditions are equivalent:*

(a) $L \in \mathbf{\Sigma_1^1}$;
(b) *L is recognizable by some N-St-RNN*$[\mathbb{R}]$;
(c) *L is recognizable by some N-Ev*$_2$*-RNN*$[\mathbb{Q}]$;
(d) *L is recognizable by some N-Ev-RNN*$[\mathbb{Q}]$;
(e) *L is recognizable by some N-Ev*$_2$*-RNN*$[\mathbb{R}]$;
(f) *L is recognizable by some N-Ev-RNN*$[\mathbb{R}]$.

Theorem 4 states that N-St-RNN[$\mathbb{Q}$]s are Turing equivalent. Theorem 5 shows that all other classes of N-St-RNN[$\mathbb{R}$]s, N-Ev$_2$-RNN[$\mathbb{Q}$], N-Ev-RNN[$\mathbb{Q}$], N-Ev$_2$-RNN[$\mathbb{R}$] and N-Ev$_2$-RNN[$\mathbb{R}$] are strictly more powerful than nondeterministic Muller Turing machines, since $\Sigma_1^1 \subsetneq \mathbf{\Sigma_1^1}$. In this sense, the nondeterministic analog and evolving neural networks are also *super-Turing*.

Remark 2. The nondeterministic counterpart of Remark 1 holds. More precisely, the proof of Theorem 5 [6, Theorem 2] shows that any ω-language $L \in \mathbf{\Sigma_1^1}$ can be recognized by some N-St-RNN[$\mathbb{R}$] employing at most one static irrational weight which is in the interval $[0, 1]$ and given in the form of a bias. Similarly, any ω-language $L \in \mathbf{\Sigma_1^1}$ can be recognized by some N-Ev$_2$-RNN[$\mathbb{Q}$] containing only one bi-valued evolving weight given as a bias. Consequently, from this point onwards, we will without loss of generality focus on the subclasses of N-St-RNN[$\mathbb{Q}, r$] and N-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], for any $r \in [0, 1]$ and $\alpha \in 2^\omega$.

We now provide a precise characterization of the expressive power of the two subclasses of N-St-RNN[$\mathbb{Q}, r$] and N-Ev$_2$-RNN[$\mathbb{Q}, \alpha$], for any $r \in [0, 1]$ and $\alpha \in 2^\omega$, respectively. This result is obtained via forthcoming Propositions 5, 6, 7 and 8, which are direct generalizations of Propositions 1, 2, 3 and 4.

Proposition 5. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $\alpha \in 2^\omega$. If $L \in \Sigma_1^1(\alpha)$, then L is recognizable by some N-Ev*$_2$*-RNN*$[\mathbb{Q}, \alpha]$.

Proof. If $L \in \Sigma_1^1(\alpha) \upharpoonright (\mathbb{B}^M)^\omega$, then by definition, there exists $L' \in \Sigma_1^1 \upharpoonright (\mathbb{B}^{M+1})^\omega$ such that

$$L = L'_\alpha = \left\{ s \in (\mathbb{B}^M)^\omega : (s, \alpha) \in L' \right\}.$$

Theorem 4 ensures that there exists a N-St-RNN[$\mathbb{Q}$] $\mathcal{N}'$ with $M + 1$ input cells such that $L(\mathcal{N}') = L'$. As in the proof of Proposition 1, one can modify network $\mathcal{N}'$ to obtain a N-Ev$_2$-RNN[$\mathbb{Q}, \alpha$] $\mathcal{N}$ such that $L(\mathcal{N}) = L'_\alpha = L$. □

Proposition 6. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $\alpha = \alpha_1\alpha_2\alpha_3 \in 2^\omega$. If $L \in \Sigma_1^1(\alpha)$, then L is recognizable by some N-St-RNN*$[\mathbb{Q}, r_\alpha]$ $\mathcal{N}$, *where $r = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$.*

Proof. Suppose that $L \in \Sigma_1^1(\alpha)$. Let $\alpha' = 00 \prod_{i=1}^{\infty}(0\alpha_i) = 00\alpha_1 0\alpha_2 0\alpha_3 0\alpha_4 0 \cdots \in 2^\omega$. One has $\alpha \in \Sigma_0^1(\alpha')$. The relations $L \in \Sigma_1^1(\alpha)$ and $\alpha \in \Sigma_0^1(\alpha')$ imply $L \in \Sigma_1^1(\alpha')$. Consequently, there exists $L' \in \Sigma_1^1 \upharpoonright (\mathbb{B}^{M+1})^\omega$ such that

$$L = L'_{\alpha'} = \left\{ s \in (\mathbb{B}^M)^\omega : (s, \alpha') \in L' \right\}.$$

By Theorem 1, there exists a N-St-RNN[$\mathbb{Q}$] $\mathcal{N}'$ with $M + 1$ input cells $u_1, \ldots, u_{M+1}$ and one guess cell u_{M+2} such that $L(\mathcal{N}') = L'$.

Now, consider once again the real encoding of α given by $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$. Consider also the N-St-RNN[$\mathbb{Q}, r_\alpha$] $\mathcal{N}$ obtained by replacing the input cell u_{M+1} of $\mathcal{N}'$ by the real-weighted neural circuit C with bias r_α depicted in Fig. 3. One has $L(\mathcal{N}) = L'_{\alpha'} = L$, which shows that L is recognized by the N-St-RNN[$\mathbb{Q}, r_\alpha$] $\mathcal{N}$. □

Proposition 7. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $\alpha \in 2^\omega$. If L is recognizable by some N-Ev*$_2$*-RNN*$[\mathbb{Q}, \alpha]$, *then $L \in \Sigma_1^1(\alpha)$.*

Proof. Let $\mathcal{N}$ be a N-Ev$_2$-RNN[$\mathbb{Q}, \alpha$] such that $L(\mathcal{N}) = L$. By Remark 2, we may assume without loss generality that the bi-valued evolving weight α of $\mathcal{N}$ is given as a bias. As in the proof of Proposition 3, we can construct from $\mathcal{N}$ a N-St-RNN[$\mathbb{Q}$] $\mathcal{N}'$ with $P + 1$ input cells and one guess cell such that, for any $(s, \alpha) \in (\mathbb{B}^{M+1})^\omega$, one has $(s, \alpha) \in L(\mathcal{N}')$ if and only if $s \in L(\mathcal{N})$. This shows that $L(\mathcal{N}) = L(\mathcal{N}')_\alpha$. Besides, Theorem 4 ensures that $L(\mathcal{N}') \in \Sigma_1^1$. Therefore, $L(\mathcal{N}) \in \Sigma_1^1(\alpha)$. □

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

Proposition 8. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language and $r \in [0,1]$. If L is recognizable by some N-St-RNN$[\mathbb{Q}, r]$, then $L \in \Sigma_1^1(\alpha)$, for some $\alpha \in 2^\omega$. In particular, if L is recognizable by some N-St-RNN$[\mathbb{Q}, r_\alpha]$, where $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ and $\alpha_i \in \{0,1\}$ for each $i \in \mathbb{N}^*$, then $L \in \Sigma_1^1(\alpha)$, where $\alpha = \alpha_1\alpha_2\alpha_3\cdots$.*

Proof. If L is recognized by some N-St-RNN$[\mathbb{Q}, r]$, then a fortiori L is recognized by some N-St-RNN$[\mathbb{R}]$. By Theorem 5, $L \in \mathbf{\Sigma_1^1}$. By Theorem 5 again, L is recognized by some N-Ev$_2$-RNN$[\mathbb{Q}]$, and by Remark 2, L is recognized by some N-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$, for some $\alpha \in 2^\omega$. By Proposition 7, $L \in \Sigma_1^1(\alpha)$.

Now, suppose that L is recognized by some N-St-RNN$[\mathbb{Q}, r_\alpha]$ $\mathcal{N}$, where $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ and $\alpha_i \in \{0,1\}$ for each $i \in \mathbb{N}^*$. By Remark 2, we may assume without loss of generality that the static weight r_α of $\mathcal{N}$ is given as a bias. Consider the infinite procedure given in previous Algorithm 1, yet slightly modified in such a way that the algorithm receives as input a guess stream $g \in 2^\omega$ provided bit by bit in addition to the input stream $s \in (\mathbb{B}^M)^\omega$ and infinite word $\alpha \in 2^\omega$. This modified version of Algorithm 1 can be simulated by some N-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$ $\mathcal{N}'$ receiving g as a guess stream and $\alpha = \alpha_1\alpha_2\alpha_3$ as an evolving bias. In addition, the accepting and rejecting attractors of $\mathcal{N}'$ are defined in the same way as in Proposition 4. By construction, $L(\mathcal{N}') = L(\mathcal{N})$. Since $\mathcal{N}'$ is a N-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$, Proposition 7 ensures that $L(\mathcal{N}') \in \Sigma_1^1(\alpha)$. Therefore, $L(\mathcal{N}) \in \Sigma_1^1(\alpha)$ too. □

By combining Propositions 5, 6, 7 and 8, the following theorem is obtained:

Theorem 6. *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω-language, $\alpha = \alpha_1\alpha_2\alpha_3\cdots \in 2^\omega$ and $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0,1]$. The following conditions are equivalent:*

(a) $L \in \Sigma_1^1(\alpha)$;
(b) L is recognizable by some N-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$;
(c) L is recognizable by some N-St-RNN$[\mathbb{Q}, r_\alpha]$.

From Theorem 6 and Remark 2, the following set-theoretical result can be retrieved:

$$\mathbf{\Sigma_1^1} = \bigcup_{\alpha \in 2^\omega} \Sigma_1^1(\alpha).$$

In other words, the relativized classes $\Sigma_1^1(\alpha)$ span the class $\mathbf{\Sigma_1^1}$, when α varies over 2^ω.

5. The hierarchy theorem

Theorems 3 and 6 provide a precise characterization of the expressive power of the classes of D-St-RNN$[\mathbb{Q}, r_\alpha]$, D-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$, N-St-RNN$[\mathbb{Q}, r_\alpha]$ and N-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$, for any $\alpha \in 2^\omega$. We will show that these classes can be stratified into transfinitely many subclasses based on the complexity of the analog and evolving weights employed by the networks.

Towards this purpose, we first present some conditions that pairs of infinite words necessarily satisfy whenever their corresponding relativized classes are included one into the other.

Proposition 9. *Let $\alpha, \beta \in 2^\omega$. The following relations hold:*

$$BC(\Pi_2^0)(\alpha) \subseteq BC(\Pi_2^0)(\beta) \longrightarrow \alpha \in \Delta_1^1(\beta) \tag{3}$$

$$\Sigma_1^1(\alpha) \subseteq \Sigma_1^1(\beta) \longleftrightarrow \alpha \in \Delta_1^1(\beta) \tag{4}$$

Proof. We prove both left-to-right implications. Recall that $\alpha \in \Sigma_1^0(\alpha)$. In the first case, one has $\alpha \in \Sigma_1^0(\alpha) \subseteq BC(\Pi_2^0)(\alpha) \subseteq BC(\Pi_2^0)(\beta) \subseteq \Delta_1^1(\beta)$. In the second case, $\alpha \in \Sigma_1^0(\alpha) \subseteq \Sigma_1^1(\alpha) \subseteq \Sigma_1^1(\beta)$, and thus $\alpha \in \Delta_1^1(\beta)$, by [36].

For the converse implication of relation (4), suppose that $\alpha \in \Delta_1^1(\beta)$. Then $\alpha \in \Sigma_1^1(\beta)$, which means that the ω-language $\{\alpha\}$ is recognized by some nondeterministic Muller TM $\mathcal{M}_1$ with oracle β. Now, let $L \in \Sigma_1^1(\alpha)$. Then L is recognized by a nondeterministic Muller TM $\mathcal{M}_2$ with oracle α. Consider the nondeterministic Muller TM $\mathcal{M}$ with oracle β which works as follows: if x is written on its input tape, then $\mathcal{M}$ nondeterministically writes some $y \in 2^\omega$ bit by bit on one of its work tape, and concomitantly, it simulates in parallel the behaviors of $\mathcal{M}_1$ on y as well as that of $\mathcal{M}_2$ with oracle y on x. The TM $\mathcal{M}$ is suitably programmed in order to always have enough bits of y being written on its work tape so that the next simulations steps of $\mathcal{M}_1$ with oracle y can be performed without fail. In addition, the machine $\mathcal{M}$ accepts input x iff both simulation processes of $\mathcal{M}_1$ and $\mathcal{M}_2$ are accepting, i.e., iff $y = \alpha$ and the simulation of $\mathcal{M}_2$ with oracle $y = \alpha$ accepts x, which is to say that $x \in L(\mathcal{M}_2) = L$. Hence, $\mathcal{M}$ recognizes L also, and thus $L \in \Sigma_1^1(\beta)$. This shows that $\Sigma_1^1(\alpha) \subseteq \Sigma_1^1(\beta)$. □

We now show the existence of an infinite sequence of infinite words whose corresponding succession of relativized classes properly stratify the "super-Turing" classes of $BC(\mathbf{\Pi_2^0})$ and $\mathbf{\Sigma_1^1}$ neural ω-languages. In addition, the hierarchy induced by the inclusion relation between the relativized classes possesses chains of length ω_1 as well as uncountable antichains.

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

Proposition 10. *There exists a sequence $(\alpha_i)_{i<\omega_1}$, where $\alpha_i \in 2^\omega$ for all $i < \omega_1$, such that*

(a) $BC(\Pi_2^0)(\alpha_0) = BC(\Pi_2^0)$ *and* $BC(\Pi_2^0)(\alpha_i) \subsetneq BC(\Pi_2^0)(\alpha_j)$, *for all* $i < j < \omega_1$;
(b) $\Sigma_1^1(\alpha_0) = \Sigma_1^1$ *and* $\Sigma_1^1(\alpha_i) \subsetneq \Sigma_1^1(\alpha_j)$, *for all* $i < j < \omega_1$.

Moreover, there exists some uncountable set $A \subseteq 2^\omega$ such that the following relations $BC(\Pi_2^0)(\alpha_i) \nsubseteq BC(\Pi_2^0)(\alpha_j)$ and $\Sigma_1^1(\alpha_i) \nsubseteq \Sigma_1^1(\alpha_j)$ hold, for every distinct $\alpha_i, \alpha_j \in A$.

Proof. Take $\alpha_0 \in \Sigma_1^0$. Suppose that for $\gamma < \omega_1$, the sequence $(\alpha_i)_{i<\gamma}$ has been constructed and satisfies the required property. We build the next element α_γ of that sequence, i.e., the element such that $\Sigma_1^1(\alpha_i) \subsetneq \Sigma_1^1(\alpha_\gamma)$, for all $i < \gamma$. Note that, for each $i < \gamma$, the set $\Delta_1^1(\alpha_i)$ is countable. Since $\gamma < \omega_1$, the union $\bigcup_{i<\gamma} \Delta_1^1(\alpha_i)$ is countable too. Hence, there exists $\alpha \in 2^\omega \setminus \bigcup_{i<\gamma} \Delta_1^1(\alpha_i)$. Now, let $\{\beta_i : i < \omega\}$ be an enumeration of the countable set $\{\alpha\} \cup \{\alpha_i : i < \gamma\}$, and let $\alpha_\gamma \in 2^\omega$ be the encoding of $\{\beta_i : i < \omega\}$ given by $\alpha_\gamma(\langle i, n\rangle) = \beta_i(n)$, where $\langle ., .\rangle : \omega^2 \to \omega$ is a classical recursive pairing function. Each function $f_i : \alpha_\gamma \mapsto (\alpha_\gamma)_i = \beta_i$ is recursive, and therefore, $\beta_i \in \Sigma_1^0(\alpha_\gamma)$, for each $i < \omega$.

We show that $BC(\Pi_2^0)(\alpha_j) \subseteq BC(\Pi_2^0)(\alpha_\gamma)$, for all $j < \gamma$. Let $L \in BC(\Pi_2^0)(\alpha_j) = BC(\Pi_2^0)(\beta_i)$, for some $i < \omega$. This means that L is recognizable by some deterministic Muller TM $\mathcal{M}$ with oracle β_i. Since $\beta_i \in \Sigma_1^0(\alpha_\gamma)$, L is also recognized by the deterministic Muller TM $\mathcal{M}'$ with oracle α_γ which, in a suitable alternating manner, produces β_i bit by bit from α_γ, and works precisely like $\mathcal{M}$ with oracle β_i. Therefore, $L \in BC(\Pi_2^0)(\alpha_\gamma)$. By replacing in this argument every occurrences of "$BC(\Pi_2^0)$" by "Σ_1^1" and of "deterministic" by "nondeterministic", one obtains that $\Sigma_1^1(\alpha_j) \subseteq \Sigma_1^1(\alpha_\gamma)$, for all $j < \gamma$.

We now show that $BC(\Pi_2^0)(\alpha_j) \subsetneq BC(\Pi_2^0)(\alpha_\gamma)$ and $\Sigma_1^1(\alpha_j) \subsetneq \Sigma_1^1(\alpha_\gamma)$, for all $j < \gamma$. Towards a contradiction, suppose that $BC(\Pi_2^0)(\alpha_\gamma) \subseteq BC(\Pi_2^0)(\alpha_j)$ or $\Sigma_1^1(\alpha_\gamma) \subseteq \Sigma_1^1(\alpha_j)$, for some $j < \gamma$. Then Relations (3) and (4) ensure that $\alpha_\gamma \in \Delta_1^1(\alpha_j)$. But $\alpha = \beta_k$ for some $k < \omega$, and by the above stated fact, $\alpha = \beta_k \in \Sigma_1^0(\alpha_\gamma)$. The two relations $\alpha \in \Sigma_1^0(\alpha_\gamma)$ and $\alpha_\gamma \in \Delta_1^1(\alpha_j)$ imply that $\alpha \in \Delta_1^1(\alpha_j)$. This contradicts the fact that $\alpha \in 2^\omega \setminus \bigcup_{i<\gamma} \Delta_1^1(\alpha_i)$.

We finally prove the existence of an uncountable antichain. There exists an uncountable set $A \subseteq 2^\omega$ such that $\alpha_i \notin \Delta_1^1(\alpha_j)$, for all distinct $\alpha_i, \alpha_j \in A$ [3]. By Relations (3) and (4), $BC(\Pi_2^0)(\alpha_i) \nsubseteq BC(\Pi_2^0)(\alpha_j)$ and $\Sigma_1^1(\alpha_i) \nsubseteq \Sigma_1^1(\alpha_j)$, for all distinct $\alpha_i, \alpha_j \in A$. □

Let $\mathcal{L}$(D-St-RNN$[\mathbb{Q}, r]$), $\mathcal{L}$(D-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$), $\mathcal{L}$(N-St-RNN$[\mathbb{Q}, r]$) and $\mathcal{L}$(N-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$) denote the classes of neural ω-languages recognized by D-St-RNN$[\mathbb{Q}, r]$, D-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$, N-St-RNN$[\mathbb{Q}, r]$ and N-Ev$_2$-RNN$[\mathbb{Q}, \alpha]$, respectively. Theorems 3 and 6 together with Proposition 10 imply the existence of four proper hierarchies of classes of deterministic and nondeterministic analog and evolving neural networks of increasing expressive power.

Theorem 7. *There exists a sequence of real numbers $(r_i)_{i<\omega_1}$ and a sequence of infinite words $(\alpha_i)_{i<\omega_1}$ such that*

(a) $\mathcal{L}$(D-St-RNN$[\mathbb{Q}, r_i]) \subsetneq \mathcal{L}$(D-St-RNN$[\mathbb{Q}, r_j])$, *for all* $i < j < \omega_1$;
(b) $\mathcal{L}$(D-Ev$_2$-RNN$[\mathbb{Q}, \alpha_i]) \subsetneq \mathcal{L}$(D-Ev$_2$-RNN$[\mathbb{Q}, \alpha_j])$, *for all* $i < j < \omega_1$;
(c) $\mathcal{L}$(N-St-RNN$[\mathbb{Q}, r_i]) \subsetneq \mathcal{L}$(N-St-RNN$[\mathbb{Q}, r_j])$, *for all* $i < j < \omega_1$;
(d) $\mathcal{L}$(N-Ev$_2$-RNN$[\mathbb{Q}, \alpha_i]) \subsetneq \mathcal{L}$(N-Ev$_2$-RNN$[\mathbb{Q}, \alpha_j])$, *for all* $i < j < \omega_1$.

Proof. Theorems 3 and 6 ensure that

$$\mathcal{L}(\text{D-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha]) = \mathcal{L}(\text{D-St-RNN}[\mathbb{Q}, r_\alpha]) = BC(\Pi_2^0)(\alpha)$$
$$\mathcal{L}(\text{N-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha]) = \mathcal{L}(\text{N-St-RNN}[\mathbb{Q}, r_\alpha]) = \Sigma_1^1(\alpha)$$

where r_α is the encoding of α described in Proposition 4, for any $\alpha \in 2^\omega$. By Proposition 10, there exists some sequence $(\alpha_i)_{i<\omega_1}$ satisfying Points (b) and (d). In addition, by taking $r_i = r_{\alpha_i}$ for all $i < \omega_1$, one obtains a sequence $(r_i)_{i<\omega_1}$ satisfying Points (a) and (c). □

Finally, let R be the equivalence relation defined by

$$R(\alpha, \beta) \text{ iff } \mathcal{L}(\text{N-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha]) = \mathcal{L}(\text{N-Ev}_2\text{-RNN}[\mathbb{Q}, \beta])$$

This relation represents the decision problem of whether two classes of nondeterministic evolving networks (determined by the evolving weights α and β) have the same expressive power. We show that this relation is undecidable and of complexity $\Pi_1^1 \setminus \Sigma_1^1$.

Proposition 11. *The equivalence relation R is in the class $\Pi_1^1 \setminus \Sigma_1^1$.*

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

Proof. According to Theorem 6 and Relation (4), the relation $R \subseteq 2^\omega \times 2^\omega$ satisfies $R(\alpha_1, \alpha_2)$ iff $\alpha_1 \in \Delta^1_1(\alpha_2)$ and $\alpha_2 \in \Delta^1_1(\alpha_1)$. It is known that the relation "$\alpha \in \Delta^1_1(\beta)$" is a Π^1_1 relation which can be expressed by a Π^1_1-formula $\phi(\alpha, \beta)$, see [36, 4D.14, p. 171] and [21]. Thus R is a Π^1_1-relation. Towards a contradiction, assume now that R is Σ^1_1, and take $\beta \in \Sigma^0_1$. Then $R(., \beta) = \{\alpha : R(\alpha, \beta)\} = \{\alpha : \alpha \in \Delta^1_1(\beta)\ \&\ \beta \in \Delta^1_1(\alpha)\} = \{\alpha : \alpha \in \Delta^1_1(\beta)\} = \{\alpha : \alpha \in \Delta^1_1\}$ should also be in Σ^1_1. But it is known that the set $\{\alpha : \alpha \in \Delta^1_1\}$ is not Σ^1_1, see [36, 4D.16, p. 171]. This concludes the proof. □

6. Conclusion

The present study concerns the expressive power of sigmoidal recurrent neural networks involved in a computational paradigm based on infinite rather than finite input streams. This approach conciliates two important biological and computer scientist perspectives about neural attractor dynamics and non-terminating computational processes, respectively.

In this context, we provided a full characterization of the expressive power of the networks. For any $\alpha \in 2^\omega$ with corresponding encoding $r_\alpha \in \mathbb{R}$, the deterministic and nondeterministic analog or evolving networks employing either the single static analog weight r_α or the single evolving weight α recognize the (lightface) relativized topological classes of $BC(\Pi^0_2)(\alpha)$ and $\Sigma^1_1(\alpha)$ ω-languages, respectively. As a consequence, two infinite refined hierarchies of classes of analog and evolving neural nets based on the complexity of their underlying analog and evolving weights are obtained. These hierarchies represent a generalization to the context of ω-computation of the fundamental previous hierarchy of classes of analog networks based on the Kolmogorov complexity of their underlying analog weights [4].

From a purely theoretical perspective, these results show that analog and evolving neural networks constitute natural equivalent models for oracle-based infinite computation, beyond the Turing limits. In the analog case, the extra-recursive power of the networks arises from their possibility to have access to more and more precise rational approximations of some given real weights [44]. In the evolving case, the extra capabilities emerge from the non-recursive patterns of evolution of the synapses [10]. Despite their mathematical equivalence, the two neural models are conceptually distinct: while the former remains at a purely conceptual level, the later relies on considerations that could be observable in nature.

From a more practical point of view, the two phenomena of attractor dynamics and synaptic plasticity are of primordial importance to the processing and coding of information in both artificial and biological neural networks. In fact, the concept of an attractor has been shown to carry strong computational implications. According to Kauffman: "Because many complex systems harbour attractors to which the system settle down, the attractors literally are most of what the systems do" [27, p.191]. In the neural network context, alternative attractors are commonly interpreted as alternative memories, but have also been associated to motor behaviors, perceptions and thoughts [31,32,23,24,2,20]. Likewise, synaptic plasticity is known to be crucially related to the storage and encoding of memory traces in the central nervous system, and provides the basis for most models of learning and memory in neural networks [1,33,40,19]. In view of these considerations, our results may constitute a theoretical foundation of the computational capabilities of neural networks in touch with these two crucial phenomena.

More generally, this study strengthen the connectedness between the fields of theoretical computer science, with possible extensions to the more practical domain of machine learning, and theoretical neuroscience. We hope that such comparative studies between neural networks and abstract machines might eventually bring further insight to the understanding of both biological and artificial intelligences. Similarly to the foundational work of Turing, which played a crucial role in the practical realization of modern computers, further theoretical considerations about neural- and natural-based models of computation might contribute to the emergence of novel computational technologies, and step by step, open the way to the next computational generation.

Acknowledgments

Partial support from DARPA project no. HR001117S0016-L2M-FP-015 is gratefully acknowledged.

References

[1] L.F. Abbott, S.B. Nelson, Synaptic plasticity: taming the beast, Nat. Neurosci. 3 Suppl. (2000) 1178–1183.
[2] D.J. Amit, Modelling Brain Function: The World of Attractor Neural Networks, 1st edition, Cambridge University Press, New York, NY, USA, 1992.
[3] K.R. Apt, ω-models in analytical hierarchy, Bull. Acad. Pol. Sci. XX (1972) 901–904.
[4] J.L. Balcázar, R. Gavaldà, H.T. Siegelmann, Computational power of neural networks: a characterization in terms of Kolmogorov complexity, IEEE Trans. Inf. Theory 43 (1997) 1175–1183.
[5] J. Cabessa, J. Duparc, Expressive power of non-deterministic evolving recurrent neural networks in terms of their attractor dynamics, in: C.S. Calude, M.J. Dinneen (Eds.), Proceedings of UCNC 2015, in: Lecture Notes in Computer Science, vol. 9252, Springer, 2015, pp. 144–156.
[6] J. Cabessa, J. Duparc, Expressive power of nondeterministic recurrent neural networks in terms of their attractor dynamics, Int. J. Unconv. Comput. 12 (2016) 25–50.
[7] J. Cabessa, O. Finkel, Expressive power of evolving neural networks working on infinite input streams, in: R. Klasing, M. Zeitoun (Eds.), Proceedings of FCT 2017, in: Lecture Notes in Computer Science, vol. 10472, Springer, 2017, pp. 150–163.
[8] J. Cabessa, H.T. Siegelmann, Evolving recurrent neural networks are super-Turing, in: Proceedings of IJCNN 2011, IEEE, 2011, pp. 3200–3206.
[9] J. Cabessa, H.T. Siegelmann, The computational power of interactive recurrent neural networks, Neural Comput. 24 (2012) 996–1019.
[10] J. Cabessa, H.T. Siegelmann, The super-Turing computational power of plastic recurrent neural networks, Int. J. Neural Syst. 24 (2014).

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003

[11] J. Cabessa, A.E.P. Villa, A hierarchical classification of first-Order recurrent neural networks, in: A.H. Dediu, et al. (Eds.), Proceedings of LATA 2010, in: Lecture Notes in Computer Science, vol. 6031, Springer, 2010, pp. 142–153.
[12] J. Cabessa, A.E.P. Villa, The expressive power of analog recurrent neural networks on infinite input streams, Theor. Comput. Science 436 (2012) 23–34.
[13] J. Cabessa, A.E.P. Villa, The super-Turing computational power of interactive evolving recurrent neural networks, in: V. Mladenov, et al. (Eds.), Proceedings of ICANN 2013, in: Lecture Notes in Computer Science, vol. 8131, Springer, 2013, pp. 58–65.
[14] J. Cabessa, A.E.P. Villa, An attractor-based complexity measurement for boolean recurrent neural networks, PLoS ONE 9 (2014) e94204+.
[15] J. Cabessa, A.E.P. Villa, Interactive evolving recurrent neural networks are super-Turing Universal, in: S. Wermter, et al. (Eds.), Proceedings of ICANN 2014, in: Lecture Notes in Computer Science, vol. 8681, Springer, 2014, pp. 57–64.
[16] J. Cabessa, A.E.P. Villa, Computational capabilities of recurrent neural networks based on their attractor dynamics, in: Proceedings of IJCNN 2015, IEEE, 2015, pp. 1–8.
[17] J. Cabessa, A.E.P. Villa, Recurrent neural networks and super-Turing interactive computation, in: P. Koprinkova-Hristova, V. Mladenov, K.N. Kasabov (Eds.), Artificial Neural Networks: Methods and Applications in Bio-/Neuroinformatics, Springer, 2015, pp. 1–29.
[18] J. Cabessa, A.E.P. Villa, Expressive power of first-order recurrent neural networks determined by their attractor dynamics, J. Comput. Syst. Sci. 82 (2016) 1232–1250.
[19] N. Caporale, Y. Dan, Spike timing-dependent plasticity: a Hebbian learning rule, Annu. Rev. Neurosci. 31 (2008) 25–46.
[20] C. Eliasmith, A unified approach to building and controlling spiking attractor networks, Neural Comput. 17 (2005) 1276–1314.
[21] O. Finkel, Ambiguity of omega-languages of Turing Machines, Log. Methods Comput. Sci. 10 (2014).
[22] R. Hartley, H. Szu, A comparison of the computational power of neural network models, in: C. Butler (Ed.), Proceedings of the IEEE First International Conference on Neural Networks, IEEE, 1987, pp. 17–22.
[23] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. 79 (1982) 2554–2558.
[24] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, Proc. Natl. Acad. Sci. 81 (1984) 3088–3092.
[25] H. Hyötyniemi, Turing machines are recurrent neural networks, in: J. Alander, T. Honkela, M. Jakobsson (Eds.), Proceedings of STeP 1996, University of Vaasa, Finnish Artificial Intelligence Society (FAIS), 1996, pp. 13–24.
[26] N. Kasabov, Evolving Connectionist Systems – The Knowledge Engineering Approach, 2nd ed., Springer, 2007.
[27] S.A. Kauffman, The Origins of Order: Self-Organization and Selection in Evolution, Oxford University Press, New York, 1993.
[28] A.S. Kechris, Classical Descriptive Set Theory, Graduate Texts in Mathematics, vol. 156, Springer, New York, NY, USA, 1995.
[29] J. Kilian, H.T. Siegelmann, The dynamic universality of sigmoidal neural networks, Inf. Comput. 128 (1996) 48–56.
[30] S.C. Kleene, Representation of events in nerve nets and finite automata, in: C. Shannon, J. McCarthy (Eds.), Automata Studies, Princeton University Press, Princeton, NJ, 1956, pp. 3–41.
[31] W.A. Little, The existence of persistent states in the brain, Math. Biosci. 19 (1974) 101–120.
[32] W.A. Little, G.L. Shaw, Analytical study of the memory storage capacity of a neural network, Math. Biosci. 39 (1978) 281–290.
[33] S.J. Martin, P.D. Grimwood, R.G.M. Morris, Synaptic plasticity and memory: an evaluation of the hypothesis, Annu. Rev. Neurosci. 23 (2000) 649–711.
[34] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.
[35] M.L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967.
[36] Y.N. Moschovakis, Descriptive Set Theory, Mathematical Surveys and Monographs, second edition, American Mathematical Society, 2009.
[37] J.a.P.G. Neto, H.T. Siegelmann, J.F. Costa, C.P.S. Araujo, Turing universality of neural nets (revisited), in: Proceedings of EUROCAST '97: Computer Aided Systems Theory, in: Lecture Notes in Computer Science, vol. 1333, Springer, London, UK, 1997, pp. 361–366.
[38] D. Perrin, J.E. Pin, Infinite Words – Automata, Semigroups, Logic and Games, Pure and Applied Mathematics, vol. 141, Elsevier, 2004.
[39] J.B. Pollack, On Connectionist Models of Natural Language Processing, Ph.D. thesis, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, 1987.
[40] P.D. Roberts, C.C. Bell, Spike timing dependent synaptic plasticity in biological systems, Biol. Cybern. 87 (2002) 392–403.
[41] H.T. Siegelmann, Recurrent neural networks and finite automata, Comput. Intell. 12 (1996) 567–574.
[42] H.T. Siegelmann, Neural Networks and Analog Computation: Beyond the Turing Limit, Birkhauser Boston Inc., Cambridge, MA, USA, 1999.
[43] H.T. Siegelmann, Neural and super-Turing computing, Minds Mach. 13 (2003) 103–114.
[44] H.T. Siegelmann, E.D. Sontag, Analog computation via neural networks, Theor. Comput. Sci. 131 (1994) 331–360.
[45] H.T. Siegelmann, E.D. Sontag, On the computational power of neural nets, J. Comput. Syst. Sci. 50 (1995) 132–150.
[46] J. Síma, P. Orponen, General-purpose computation with neural networks: a survey of complexity theoretic results, Neural Comput. 15 (2003) 2727–2778.
[47] L. Staiger, ω-languages, in: Handbook of Formal Languages, Vol. 3: Beyond Words, Springer, New York, NY, USA, 1997, pp. 339–387.
[48] K.O. Stanley, R. Miikkulainen, Evolving neural network through augmenting topologies, Evol. Comput. 10 (2002) 99–127.
[49] W. Thomas, Automata on infinite objects, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, Elsevier and MIT Press, 1990, pp. 133–192.
[50] A.M. Turing, Intelligent Machinery, Technical Report, National Physical Laboratory, Teddington, UK, 1948.

Please cite this article in press as: J. Cabessa, O. Finkel, Computational capabilities of analog and evolving neural networks over infinite input streams, J. Comput. Syst. Sci. (2018), https://doi.org/10.1016/j.jcss.2018.11.003