



HAL
open science

Computational Capabilities of Analog and Evolving Neural Networks over Infinite Input Streams

Jérémie Cabessa, Olivier Finkel

► **To cite this version:**

Jérémie Cabessa, Olivier Finkel. Computational Capabilities of Analog and Evolving Neural Networks over Infinite Input Streams. Journal of Computer and System Sciences, 2019, 10.1016/j.jcss.2018.11.003 . hal-02318257

HAL Id: hal-02318257

<https://hal.science/hal-02318257>

Submitted on 16 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1 Computational Capabilities of Analog and Evolving
2 Neural Networks over Infinite Input Streams

3 Jérémie Cabessa^a, Olivier Finkel^b

4 ^a*Laboratoire d'économie mathématique et de microéconomie appliquée (LEMMA),*
5 *Université Paris 2 - Panthéon-Assas, 4 Rue Blaise Desgoffe, 75006 Paris, France*
6 *jeremie.cabessa@u-paris2.fr*

7 ^b*Institut de Mathématiques de Jussieu - Paris Rive Gauche, CNRS et Université Paris*
8 *Diderot, UFR de mathématiques case 7012, 75205 Paris Cedex 13, France*
9 *finkel@math.univ-paris-diderot.fr*

10 **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.

11 *Keywords:* recurrent neural networks, analog computation, infinite
12 computation, attractors, Turing machines, Turing machines with oracles,
13 super-Turing, ω -languages, Borel sets, analytic sets,
14 *2008 MSC:* 92B20, 68Q15, 68Q45, 68Q32, 68Q05, 03E15

15 **1. Introduction**

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

20 In this context, the theoretical approach to neural computation consists
21 of studying the computational power of neural network models from the
22 perspective of automata theory. The capabilities of neural networks are
23 known to be related to the kind of activation functions used by the neu-
24 rons, to the nature of their synaptic connections, to the eventual presence
25 of noise in the model, and to the possibility for the neural architecture to
26 evolve over time. The computational capabilities of diverse neural mod-
27 els have been shown to range from the finite automaton level [1, 2, 3, 4],
28 up to the Turing [5, 6, 7, 8, 9, 10, 11] or even to the super-Turing de-
29 gree [12, 13, 14, 15, 16, 17] (for detailed survey, see [18]).

30 More specifically, real-weighted neural networks, also referred to as *analog*
31 *neural nets*, are strictly more powerful than Turing machines. In exponen-
32 tial time of computation, they can decide any possible discrete language.
33 In polynomial time of computation, they are equivalent to Turing machines
34 with polynomially bounded advice, and hence decide the complexity class
35 $P/poly$ [12, 14, 15]. Interestingly, the super-Turing computational capabili-
36 ties of analog networks can be finely characterized in terms of the Kolmogorov
37 complexity of their underlying synaptic real weights. A proper infinite hierar-
38 chy of classes of analog neural nets with real weights of increasing Kolmogorov
39 complexity has been obtained [13]. Besides this, it has been shown that
40 neural networks employing time-dependent synaptic weights, called *evolving*
41 *neural nets*¹, are computationally equivalent to the analog ones. This com-
42 putational equivalence holds irrespectively of whether the synaptic weights
43 of networks are modeled by rational or real numbers and their patterns of
44 evolution restricted to binary updates or expressed by more general form of
45 updating [16, 17].

46 Based on biological and computational considerations, these studies have
47 been extended to alternative paradigms of computation where the networks
48 process infinite rather than finite input streams [16, 23, 24, 25, 26, 17, 27, 28,
49 29, 30, 31, 19, 20]. This approach conciliates two important biological and
50 computer scientist perspectives about neural attractor dynamics on the one
51 hand [32] and non-terminating computational processes on the other [33, 34].

¹Throughout this paper, the expressions *evolving neural networks* refers to neural net-
works with time-dependent synaptic weights, along the lines of [17, 19, 20]. This expression
is not to be understood in the sense of *Evolving Connectionist Systems (ECoS)* [21] nor
in that of *Evolving Neural Networks through Augmenting Topologies (NEAT)* [22].

52 The networks are provided with Boolean input and output cells carrying out
 53 the discrete exchange of information with their environment. When subjected
 54 to some infinite input stream, the outputs of the networks eventually get
 55 trapped into some attractor dynamics. The set of input streams inducing
 56 a meaningful attractor dynamics is the neural ω -language recognized by the
 57 network. The expressive power of the networks is then characterized by the
 58 topological complexity of their underlying neural ω -languages.

59 Within this framework, the Boolean neural networks provided with cer-
 60 tain type specification of their attractors are computationally equivalent to
 61 Büchi or Muller automata [24, 28]. As a consequence, a novel attractor-
 62 based measure of complexity for Boolean neural networks has been obtained.
 63 This complexity measure refers to the ability of the networks to perform
 64 more or less complicated classification tasks of their input streams via the
 65 manifestation of meaningful or spurious attractor dynamics.

66 The sigmoidal neural networks are strictly more powerful than their Bool-
 67 ean counterparts. The static rational-weighted neural networks are compu-
 68 tationally equivalent to Muller Turing machines. In the deterministic and
 69 nondeterministic cases, these networks recognize the (lightface) topological
 70 classes of $BC(\Pi_2^0)$ and Σ_1^1 neural ω -languages, respectively [29, 20]. By con-
 71 trast, the static real-weighted (or analog) neural networks are super-Turing.
 72 In the deterministic and nondeterministic cases, they recognize the (bold-
 73 face) topological classes of $BC(\mathbf{\Pi}_2^0)$ and $\mathbf{\Sigma}_1^1$ neural ω -languages, respectively
 74 [31, 19, 29, 20]. In addition, the evolving neural networks are computationally
 75 equivalent to the static analog ones. This equivalence holds irrespectively
 76 of whether the static and evolving weights of the networks are modeled by
 77 rational or real numbers, and the patterns of evolution restricted to binary
 78 updates or expressed by more general forms of updating.

79 In this paper, we provide an optimal refinement of these results and com-
 80 plete our study undertaken in [35], where only the case of evolving neural
 81 nets is treated in a more succinct way. We fully characterize the expressive
 82 power of analog and evolving networks according to the specific analog and
 83 evolving weights that they employ. Without loss of generality, we focus on
 84 analog or evolving networks using only one analog or one evolving weight,
 85 respectively. For any $\alpha \in 2^\omega$ with corresponding encoding $r_\alpha \in \mathbb{R}$, we show
 86 that deterministic and nondeterministic analog or evolving networks employ-
 87 ing either the single static analog weight r_α or the single evolving weight α
 88 recognize the (lightface) relativized topological classes of $BC(\Pi_2^0)(\alpha)$ and

89 $\Sigma_1^1(\alpha)$ ω -languages, respectively. As a consequence, we show the existence of
 90 two infinite refined hierarchies of classes of analog and evolving neural nets
 91 based on the complexity of their underlying analog and evolving weights.
 92 These hierarchies contain chains of length ω_1 and antichains of uncountable
 93 size.

94 From the point of view of theoretical computer science, these results con-
 95 stitute a generalization of the fundamental hierarchy of classes of analog
 96 networks based on the Kolmogorov complexity of their underlying analog
 97 weights [13]. They provide an optimal refinement of the super-Turing ex-
 98 pressive power of analog and evolving neural networks working on infinite
 99 input streams. They also show that analog and evolving neural networks
 100 represent natural models for oracle-based infinite computation, beyond the
 101 Turing limits. From a biological point of view, these achievements may con-
 102 stitute a theoretical foundation of the primary role played by synaptic plas-
 103 ticity in the computational capabilities of neural networks [36, 37, 38, 39].

104 2. Preliminaries

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

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

121 In the sequel, any space of the form X^ω is assumed to be equipped with
 122 the product topology of the discrete topology on X . Accordingly, the basic
 123 open sets of X^ω are of the form $p \cdot X^\omega$, for some $p \in X^*$. The general open sets

124 are countable unions of basic open sets. In particular, the space of infinite
 125 words of bits (Cantor space) and that of infinite words of N -dimensional
 126 Boolean vectors will be denoted by $2^\omega = \{0, 1\}^\omega$ and $(\mathbb{B}^N)^\omega$, respectively.
 127 They are assumed to be equipped with the above mentioned topology.

128 Let $(\mathcal{X}, \mathcal{T})$ be one of the above topological spaces, or a product of such
 129 spaces. The class of *Borel subsets* of \mathcal{X} , denoted by Δ_1^1 (boldface), is the σ -
 130 algebra generated by \mathcal{T} , i.e., the smallest collection of subsets of \mathcal{X} containing
 131 all open sets and closed under countable union and complementation. For
 132 every non-null countable ordinal $\alpha < \omega_1$, where ω_1 is the first uncountable
 133 ordinal, the Borel classes Σ_α^0 , Π_α^0 and Δ_α^0 of \mathcal{X} are defined as follows:

- 134 • Σ_1^0 is the class of open subsets of \mathcal{X} (namely \mathcal{T})
- 135 • Π_1^0 is the class of closed subsets of \mathcal{X} , i.e., that of complements of open
 136 sets
- 137 • Σ_α^0 is the class of countable unions of subsets of \mathcal{X} in $\bigcup_{\gamma < \alpha} \Pi_\gamma^0$
- 138 • Π_α^0 is the class of countable intersections of subsets of \mathcal{X} in $\bigcup_{\gamma < \alpha} \Sigma_\gamma^0$.
- 139 • $\Delta_\alpha^0 = \Sigma_\alpha^0 \cap \Pi_\alpha^0$

140 The classes Σ_α^0 , Π_α^0 and Δ_α^0 provide a stratification of the class of Borel sets
 141 known as the *Borel hierarchy*. One has $\Delta_1^1 = \bigcup_{\alpha < \omega_1} \Sigma_\alpha^0 = \bigcup_{\alpha < \omega_1} \Pi_\alpha^0$ [40].
 142 The *rank* of a Borel set $A \subseteq \mathcal{X}$ is the smallest ordinal α such that $A \in$
 143 $\Sigma_\alpha^0 \cup \Pi_\alpha^0$. It is commonly considered as a relevant measure of the topological
 144 complexity of Borel sets. The class of sets obtained as finite Boolean combi-
 145 nations (unions, intersections and complementations) of Π_2^0 -sets is denoted
 146 by $BC(\Pi_2^0)$.

147 *Analytic sets* are obtained as projections of either Π_2^0 -sets or general Borel
 148 sets [40]. More precisely, a set $A \subseteq \mathcal{X}$ is *analytic* if there exists some Π_2^0 -
 149 set $B \subseteq \mathcal{X} \times 2^\omega$ such that $A = \{x \in \mathcal{X} : (x, \beta) \in B, \text{ for some } \beta \in 2^\omega\} =$
 150 $\pi_1(B)$ [40]. The class of analytic sets is denoted by Σ_1^1 . It strictly contains
 151 that of Borel sets, i.e., $\Delta_1^1 \subsetneq \Sigma_1^1$ [40].

152 The *effective* (lightface) counterpart of the Borel and analytic classes,
 153 denoted by Σ_n^0 , Π_n^0 , Δ_n^0 as well as Δ_1^1 and Σ_1^1 , are obtained by a similar effec-
 154 tive construction, yet starting from the class Σ_1^0 of effective open sets [41].
 155 The class of finite Boolean combinations of Π_2^0 -sets, denoted by $BC(\Pi_2^0)$
 156 (lightface), and that of effective analytic sets, denoted by Σ_1^1 (lightface),
 157 correspond to the collections of ω -languages recognizable by deterministic
 158 and nondeterministic Muller Turing machines, respectively [42]. One has

159 $BC(\Pi_2^0) \subsetneq BC(\Pi_2^0)$ and $\Sigma_1^1 \subsetneq \Sigma_1^1$.

160 Any topological class Γ of the topological space \mathcal{X} will also be written
 161 as $\Gamma \upharpoonright \mathcal{X}$, whenever the underlying space \mathcal{X} is needed to be specified. In
 162 addition, for any point $x \in \mathcal{X}$, we will use the notation $x \in \Gamma$ to mean that
 163 $\{x\} \in \Gamma$. Besides, any product space $\mathcal{X} \times \mathcal{Y}$ is assumed to be equipped with
 164 the product topology. If $A \subseteq \mathcal{X} \times \mathcal{Y}$ and $y \in \mathcal{Y}$, the *y-section* of A is defined
 165 by $A_y = \{x \in \mathcal{X} : (x, y) \in A\}$. For any class Γ being equal to Σ_1^0 , $BC(\Pi_2^0)$,
 166 Σ_1^1 , or Π_1^1 with underlying product space $\mathcal{X} \times \mathcal{Y}$ and for any $y \in \mathcal{Y}$, the
 167 *relativization of Γ to y* , denoted by $\Gamma(y)$, is the class of all y -sections of sets
 168 in Γ . In other words, $A \in \Gamma(y) \upharpoonright \mathcal{X}$ if and only if there exists $B \in \Gamma \upharpoonright \mathcal{X} \times \mathcal{Y}$
 169 such that $A = B_y$. Moreover, we denote as usual $\Delta_1^1(y) = \Sigma_1^1(y) \cap \Pi_1^1(y)$ [41,
 170 p. 118].

171 For any $\alpha \in 2^\omega$, one can show that the relativized classes $BC(\Pi_2^0)(\alpha)$ and
 172 $\Sigma_1^1(\alpha)$ correspond to the collections of ω -languages recognizable by determin-
 173 istic and nondeterministic Muller Turing machine with oracle α , respectively.
 174 In addition, it can be shown that $x \in \Sigma_1^0(\alpha)$ if and only if the successive let-
 175 ters of x can be produced step by step by some Turing machine with oracle
 176 α . Besides, one has $x \in \Sigma_1^1(\alpha)$ iff $x \in \Delta_1^1(\alpha)$, for any $\alpha \in 2^\omega$ [41].

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

180 3. Recurrent Neural Networks on Infinite Input Streams

181 We consider first-order recurrent neural networks composed of Boolean
 182 input cells, Boolean output cells and sigmoidal internal cells. The sigmoidal
 183 internal neurons introduce the biological source of nonlinearity which is cru-
 184 cial to neural computation. They provide the possibility to surpass the ca-
 185 pabilities of finite state automata, or even of Turing machines. The Boolean
 186 input and output cells carry out the exchange of discrete information be-
 187 tween the network and the environment. When some infinite input stream
 188 is supplied, the output cells eventually enter into some attractor dynamics.
 189 The expressive power of the networks is related to the attractor dynamics of
 190 their Boolean output cells.

191 *3.1. Deterministic case*

192 A *deterministic (first-order) recurrent neural network* (D-RNN) consists
 193 of a synchronous network of neurons related together in a general architec-
 194 ture. It is composed of M Boolean input cells $(u_i)_{i=1}^M$, N sigmoidal internal
 195 neurons $(x_i)_{i=1}^N$, and P Boolean output cells $(y_i)_{i=1}^P$. The dynamics of the
 196 network is computed as follows: given the activation values of the input and
 197 internal neurons $(u_j)_{j=1}^M$ and $(x_j)_{j=1}^N$ at time t , the activation value of each
 198 internal and output neuron x_i and y_i at time $t+1$ is updated by the following
 199 equations, respectively:

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

200

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

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-sigmoid² and Heaviside
 step activation functions defined by

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

201 A synaptic weight or a bias w will be called *static* if it remains constant
 202 over time, i.e., if $w(t) = c$ for all $t \geq 0$. It will be called *bi-valued evolving*
 203 if it varies among two possible values over time, i.e., if $w(t) \in \{0, 1\}$ for all
 204 $t \geq 0$. It will be called *general evolving* otherwise. A D-RNN is illustrated
 205 in Figure 1.

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

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

²The seminal results concerning the computational power of rational- and real-weighted
 neural networks have been obtained in this context of linear-sigmoid functions [12, 8].
 It has then been shown that these results remain valid for any other kind of sigmoidal
 activation function satisfying the properties mentioned in [9, Section 4].

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

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

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

208 The second element of this pair, namely $\vec{y}(t)$, is the *output state* of \mathcal{N} at time
 209 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

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

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

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

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

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

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

212 Note that any D-RNN \mathcal{N} with P Boolean output cells can only have
 213 2^P – i.e., finitely many – possible distinct output states. Consequently, any
 214 Boolean computation bc_s necessarily consists of a finite prefix of output states
 215 followed by an infinite suffix of output states that repeat infinitely often – yet
 216 not necessarily in a periodic manner – denoted by $\text{inf}(bc_s)$. A set of states
 217 of the form $\text{inf}(bc_s) \subseteq \mathbb{B}^P$ will be called an *attractor* of \mathcal{N} [28]. A precise
 218 definition can be given as follows:

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

222 In words, an attractor of \mathcal{N} is a set of output states into which the
 223 Boolean computation of the network could become forever trapped – yet not

224 necessarily in a periodic manner. An attractor of some D-RNN is illustrated
 225 in Figure 1.

226 In this work, we further suppose that the networks' attractors can be of
 227 two distinct types, namely either *accepting* or *rejecting*. The classification of
 228 attractors into meaningful (accepting) or spurious (rejecting) types is an issue
 229 of significant importance in neural network studies [28]; however, it is not
 230 the subject of this work. Here, we rather consider that the type specification
 231 of the networks' attractors has already been established, e.g., according to
 232 some neurophysiological criteria or computational requirements. Hence, from
 233 this point onwards, we always assume that a D-RNN is provided with an
 234 associated classification of all of its attractors into accepting and rejecting
 235 types.

236 This classification of attractors leads to the following Muller-like accep-
 237 tance condition: given some D-RNN \mathcal{N} , an infinite input stream $s \in (\mathbb{B}^M)^\omega$ is
 238 *accepted* \mathcal{N} if $\text{inf}(bc_s)$ is an accepting attractor; it is *rejected* by \mathcal{N} if $\text{inf}(bc_s)$
 239 is a rejecting attractor. The set of all accepted input streams of \mathcal{N} is the

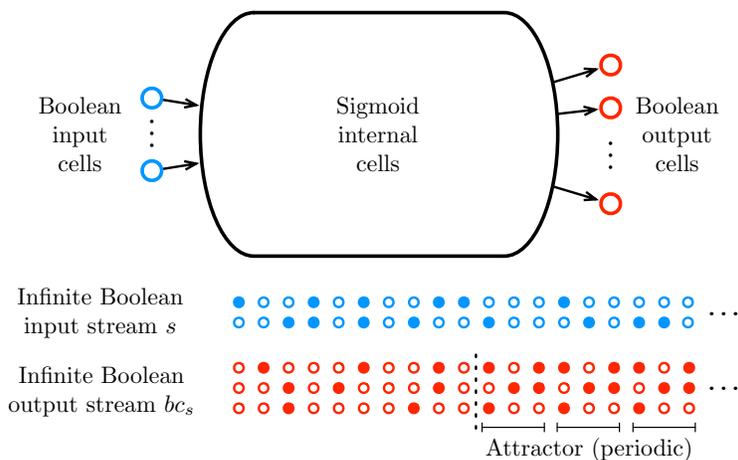


Figure 1: Illustration of the computational process performed by some D-RNN. The infinite Boolean input stream $s = \vec{u}(0)\vec{u}(1)\vec{u}(2)\dots \in (\mathbb{B}^M)^\omega$ induces a corresponding Boolean output stream – or Boolean computation – $bc_s = \vec{y}(0)\vec{y}(1)\vec{y}(2)\dots \in (\mathbb{B}^P)^\omega$. 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 .

240 *neural ω -language recognized by \mathcal{N}* , denoted by $L(\mathcal{N})$. A set $L \subseteq (\mathbb{B}^M)^\omega$ is
 241 said to be *recognizable* by some D-RNN if there exists a network \mathcal{N} such that
 242 $L(\mathcal{N}) = L$.

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

- 245 1. The class of *deterministic static rational neural nets* refers to the D-
 246 RNNs whose all weights are static rational values. It is denoted by
 247 D-St-RNN[\mathbb{Q}]s.
- 248 2. The class of *deterministic static real (or analog) neural nets* refers to
 249 the D-RNNs whose all weights are static real values. It is denoted by
 250 D-St-RNN[\mathbb{R}]s. For the purpose of our study, we stratify this class
 251 into uncountably many subclasses, each one being defined according
 252 to some specific real weights involved in the networks. Formally, for
 253 each $r_1, \dots, r_k \in \mathbb{R}$, the subclass of networks containing r_1, \dots, r_k as
 254 real weights³ and all other ones being rational is denoted by D-St-
 255 RNN[$\mathbb{Q}, r_1, \dots, r_k$]s.
- 256 3. The class of *deterministic bi-valued evolving rational neural nets* refers
 257 to the D-RNNs whose all non-static weights are bi-valued evolving and
 258 all static weight are rational. It is denoted by D-Ev₂-RNN[\mathbb{Q}]s. For
 259 each $\alpha_1, \dots, \alpha_k \in 2^\omega$, the subclass of networks containing $\alpha_1, \dots, \alpha_k$
 260 as sole bi-valued evolving weights, all other ones being static rational, is
 261 denoted by D-Ev₂-RNN[$\mathbb{Q}, \alpha_1, \dots, \alpha_k$]s.
- 262 4. The class of *deterministic (general) evolving rational neural nets* refers
 263 to the D-RNNs whose all static and evolving weights are rational. It is
 264 denoted by D-Ev-RNN[\mathbb{Q}]s.
- 265 5. The class of *deterministic bi-valued evolving real neural nets* refers to
 266 the D-RNNs whose all non-static weights are bi-valued evolving and all
 267 static weight are real. It is denoted by D-Ev₂-RNN[\mathbb{R}]s.
- 268 6. The class of *deterministic (general) evolving real neural nets* refers to
 269 the D-RNNs whose all static and evolving weights are real. It is denoted
 270 by D-Ev-RNN[\mathbb{R}]s.

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

271 3.2. Nondeterministic case

272 We also consider nondeterministic recurrent neural networks, as intro-
 273 duced in [12, 8]. The nondeterminism is expressed by means of an external
 274 binary guess stream processed via some additional Boolean guess cell.

275 Formally, a *nondeterministic (first-order) recurrent neural network* (N-
 276 RNN) consists of a recurrent neural network \mathcal{N} as described in previous
 277 Section 3.1, except that it contains $M + 1$ Boolean input cells $(u_i)_{i=1}^{M+1}$, rather
 278 than M . The cell u_{M+1} , called the *guess cell*, carries the Boolean source of
 279 nondeterminism to be considered [12, 8, 25, 19, 20]. A N-RNN is illustrated
 280 in Figure 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

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

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

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

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

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

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

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

- 295 1. The class of *nondeterministic static rational neural nets* N-St-RNN[Q]s.

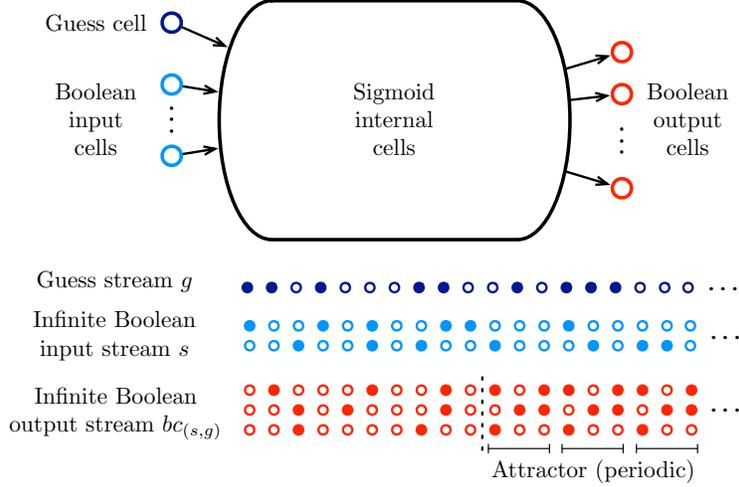


Figure 2: Illustration of the computational process performed by some N-RNN. The infinite guess stream $g = g(0)g(1)g(2) \cdots \in 2^\omega$, represented by the dark blue pattern, together with the infinite Boolean input stream $s = \vec{u}(0)\vec{u}(1)\vec{u}(2) \cdots \in (\mathbb{B}^M)^\omega$ induce 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 Figure 1, the network necessarily enters into some attractor dynamics.

- 296 2. The class of *nondeterministic static real (or analog) neural nets* N-
 297 St-RNN[\mathbb{R}]s. For each $r_1, \dots, r_k \in \mathbb{R}$, we consider the corresponding
 298 subclass N-St-RNN[$\mathbb{Q}, r_1, \dots, r_k$]s.
- 299 3. The class of *nondeterministic bi-valued evolving rational neural nets* N-
 300 Ev₂-RNN[\mathbb{Q}]s. For each $\alpha_1, \dots, \alpha_k \in 2^\omega$, we consider the corresponding
 301 subclass N-Ev₂-RNN[$\mathbb{Q}, \alpha_1, \dots, \alpha_k$]s.
- 302 4. The class of *nondeterministic (general) evolving rational neural nets*
 303 N-Ev-RNN[\mathbb{Q}]s.
- 304 5. The class of *nondeterministic bi-valued evolving real neural nets* N-Ev₂-
 305 RNN[\mathbb{R}]s.
- 306 6. The class of *nondeterministic (general) evolving real neural nets* N-Ev-
 307 RNN[\mathbb{R}]s.

308 **4. Expressive Power of Neural Networks**

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

314 *4.1. Deterministic case*

315 The expressive power of the classes of D-St-RNN[\mathbb{Q}], D-St-RNN[\mathbb{R}], D-
 316 Ev₂-RNN[\mathbb{Q}], D-Ev-RNN[\mathbb{Q}], D-Ev₂-RNN[\mathbb{R}], and D-Ev-RNN[\mathbb{R}] has been
 317 characterized in [20, Theorems 1, 2]. We first recall these results.

318 **Theorem 1.** [20, Theorem 1] *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language. The*
 319 *following conditions are equivalent:*

- 320 (a) $L \in BC(\Pi_2^0)$;
- 321 (b) L is recognizable by some D-St-RNN[\mathbb{Q}];
- 322 (c) L is recognizable by some deterministic Muller Turing machine.

323 **Theorem 2.** [20, Theorem 2] *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language. The*
 324 *following conditions are equivalent:*

- 325 (a) $L \in BC(\Pi_2^0)$;
- 326 (b) L is recognizable by some D-St-RNN[\mathbb{R}];
- 327 (c) L is recognizable by some D-Ev₂-RNN[\mathbb{Q}];
- 328 (d) L is recognizable by some D-Ev-RNN[\mathbb{Q}];
- 329 (e) L is recognizable by some D-Ev₂-RNN[\mathbb{R}];
- 330 (f) L is recognizable by some D-Ev-RNN[\mathbb{R}].

331 Theorem 1 states that D-St-RNN[\mathbb{Q}]s are Turing equivalent. Theorem
 332 2 shows that the classes D-St-RNN[\mathbb{R}]s, D-Ev₂-RNN[\mathbb{Q}]s, D-Ev-RNN[\mathbb{Q}]s,
 333 D-Ev₂-RNN[\mathbb{R}]s and D-Ev-RNN[\mathbb{R}]s are computationally equivalent to each
 334 other and strictly more powerful than deterministic Muller Turing machines,
 335 since $BC(\Pi_2^0) \subsetneq BC(\Pi_2^0)$. In this sense, the deterministic analog and evolu-
 336 ting neural networks are *super-Turing*. Note that the D-Ev₂-RNN[\mathbb{Q}]s achieve
 337 a maximal expressive power by recognizing the whole class of $BC(\Pi_2^0)$ ω -
 338 languages. Indeed, the consideration of either real synaptic weights or more

339 complex evolving patterns in the model does actually not yield to some higher
 340 expressive power.

341 **Remark 1.** The proof of implication “(a) \rightarrow (b)” of Theorem 2, detailed in
 342 [20, Proposition 1], shows that any ω -language $L \in BC(\Pi_2^0)$ can be recog-
 343 nized by some D-St-RNN[\mathbb{R}] employing at most one static irrational weight,
 344 which is in the interval $[0, 1]$ and given in the form of a bias. Similarly,
 345 the proof of implication “(a) \rightarrow (c)” of Theorem 2, also detailed in [20,
 346 Proposition 1], ensures that any ω -language $L \in BC(\Pi_2^0)$ can be recognized
 347 by some D-Ev₂-RNN[\mathbb{Q}] using only one bi-valued evolving weight given as a
 348 bias (cf. [20, Proposition 1] again). By Theorem 2, this means that any D-St-
 349 RNN[\mathbb{R}] is expressively equivalent to some D-St-RNN[\mathbb{Q}, r], where $r \in [0, 1]$,
 350 and any D-Ev₂-RNN[\mathbb{Q}] is expressively equivalent to some D-Ev₂-RNN[\mathbb{Q}, α],
 351 where $\alpha \in 2^\omega$. Hence, from this point onwards, we will focus without loss
 352 of generality on the two specific subclasses of analog or evolving networks
 353 employing only one analog or evolving weight, respectively.

354 We now provide a precise characterization of the expressive power of these
 355 two subclasses of D-St-RNN[\mathbb{Q}, r] and D-Ev₂-RNN[\mathbb{Q}, α], for any $r \in [0, 1]$
 356 and $\alpha \in 2^\omega$, respectively. This result constitutes a significant refinement of
 357 Theorem 2. It is obtained via forthcoming Propositions 1, 2, 3 and 4.

358 **Proposition 1.** *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language and $\alpha \in 2^\omega$. If $L \in$
 359 $BC(\Pi_2^0)(\alpha)$, then L is recognizable by some D-Ev₂-RNN[\mathbb{Q}, α].*

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

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

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

362 Now, consider the D-Ev₂-RNN[\mathbb{Q}, α] \mathcal{N} which consists in a slight modi-
 363 fication of the D-St-RNN[\mathbb{Q}] \mathcal{N}' . More precisely, \mathcal{N} contains the same cells
 364 and synaptic connections as \mathcal{N}' , it admits u_1, \dots, u_M as its input cells, and
 365 the cell u_{M+1} is transformed into an internal cell receiving the bi-valued
 366 evolving weight $\alpha \in 2^\omega$ in the form of a bias. In addition, the attractors of
 367 \mathcal{N} are the same as those of \mathcal{N}' . By construction, for any input $s \in (\mathbb{B}^M)^\omega$,
 368 \mathcal{N} receives the bi-valued evolving weight α as a bias and works precisely

369 like \mathcal{N}' on input $(s, \alpha) \in (\mathbb{B}^{M+1})^\omega$. Consequently, $s \in L(\mathcal{N})$ if and only if
 370 $(s, \alpha) \in L(\mathcal{N}') = L'$. Therefore, $L(\mathcal{N}) = L'_\alpha = L$. This shows that L is
 371 recognized by the D-Ev₂-RNN[\mathbb{Q}, α] \mathcal{N} . \square

372 **Proposition 2.** *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language and $\alpha = \alpha_1\alpha_2\alpha_3\cdots \in$
 373 2^ω . If $L \in BC(\Pi_2^0)(\alpha)$, then L is recognizable by some D-St-RNN[\mathbb{Q}, r_α] \mathcal{N} ,
 374 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

$$\alpha' = 00 \prod_{i=1}^{\infty} (0\alpha_i) = 000\alpha_10\alpha_20\alpha_30\alpha_40\cdots \in 2^\omega.$$

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 $L' \in BC(\Pi_2^0) \upharpoonright (\mathbb{B}^{M+1})^\omega$ such that

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

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

377 Now, consider the real encoding of α given by $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$.
 378 Consider also the D-St-RNN[\mathbb{Q}, r_α] \mathcal{N} obtained by replacing the input cell
 379 u_{M+1} of \mathcal{N}' by the real-weighted neural circuit C with bias r_α depicted in
 380 Figure 3. Circuit C is designed in such a way that it outputs the successive
 381 bits of α' at each successive time step (see Figure 3 for further details of this
 382 decoding procedure). By construction, for any $s \in (\mathbb{B}^M)^\omega$, the behavior of
 383 \mathcal{N} on input s is the same as that of \mathcal{N}' on input (s, α') . In other words,
 384 $s \in L(\mathcal{N})$ if and only if $(s, \alpha') \in L(\mathcal{N}') = L'$. Therefore, $L(\mathcal{N}) = L'_{\alpha'} = L$.
 385 This shows that L is recognized by the D-St-RNN[\mathbb{Q}, r_α] \mathcal{N} . \square

386 **Proposition 3.** *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language and $\alpha \in 2^\omega$. If L is
 387 recognizable by some D-Ev₂-RNN[\mathbb{Q}, α], then $L \in BC(\Pi_2^0)(\alpha)$.*

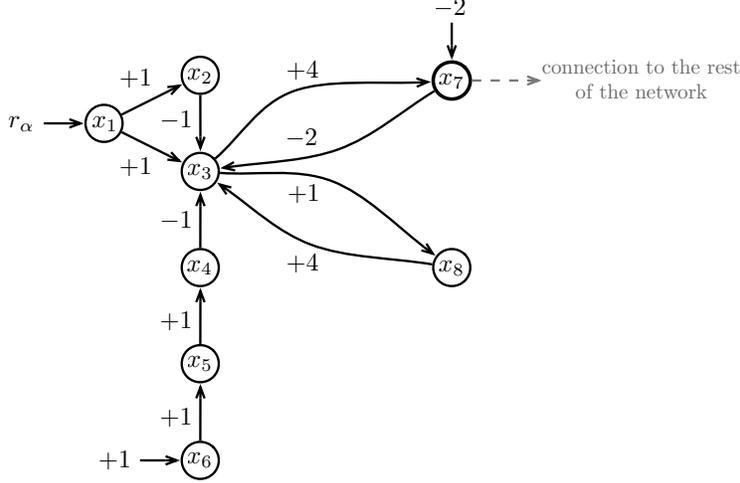


Figure 3: Circuit C : nodes represent sigmoidal neurons and labelled edges are weighted synaptic connections between those. Cell x_1 receives r_α as bias and cell x_7 outputs the successive bits of $\alpha' = 000\alpha_10\alpha_20\alpha_30\dots$. In order to understand this circuit, the following notions need to be recalled [8]. For any $\gamma = \gamma_1\gamma_2\gamma_3\dots \in 2^\omega$, we suppose that γ is a stack whose elements from top to bottom are $\gamma_1, \gamma_2, \gamma_3, \dots$. 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 $\text{top}(\gamma) = \sigma(4r_\gamma - 2)$. In addition, the encoding of the stack $\gamma_2\gamma_3\gamma_4\dots$, which corresponds to the stack γ whose top element has been popped, is given by $\text{pop}(\gamma) = \sigma(4r_\gamma - 2\text{top}(\gamma) - 1)$. 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 $\text{top}(\alpha) = \sigma(4r_\alpha - 2) = \alpha_1$ 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 $\text{pop}(r_\alpha) = \sigma(4r_\alpha - 2\text{top}(\alpha) - 1)$ at time 4. Afterwards, x_7 computes $\text{top}(\text{pop}(r_\alpha)) = \alpha_2$ at time 5, and x_3 computes $\text{pop}(\text{pop}(r_\alpha)) = \text{pop}^2(r_\alpha)$ at time 6. And so on ad infinitum. Hence, x_7 outputs $\text{top}(\text{pop}^i(r_\alpha)) = \alpha_{i+1}$ 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 $\alpha' = 000\alpha_10\alpha_20\alpha_30\dots$ at successive time steps $0, 1, 2, \dots$

388 *Proof.* Let \mathcal{N} be a D-Ev₂-RNN[\mathbb{Q}, α] such that $L(\mathcal{N}) = L$. By Remark 1, we
389 may assume without loss generality that the bi-valued evolving weight α of
390 \mathcal{N} is a bias related to some cell x . Let \mathcal{N}' be the D-St-RNN[\mathbb{Q}] obtained by
391 replacing in \mathcal{N} the cell x and its associated bias by a new input cell u_{M+1} .
392 Network \mathcal{N}' is a D-St-RNN[\mathbb{Q}] with $M + 1$ input cells, and Theorem 1 ensures

393 that $L(\mathcal{N}') \in BC(\Pi_2^0)$. By construction, for any $(s, \alpha) \in (\mathbb{B}^{M+1})^\omega$, the
 394 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
 395 other words, $(s, \alpha) \in L(\mathcal{N}')$ if and only if $s \in L(\mathcal{N})$. Thus $L(\mathcal{N}) = L(\mathcal{N}')_\alpha$.
 396 Since $L(\mathcal{N}') \in BC(\Pi_2^0)$, it follows that $L(\mathcal{N}) \in BC(\Pi_2^0)(\alpha)$. \square

397 **Proposition 4.** *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language and $r \in [0, 1]$. If L
 398 is recognizable by some D-St-RNN[\mathbb{Q}, r], then $L \in BC(\Pi_2^0)(\alpha)$, for some
 399 $\alpha \in 2^\omega$. In particular, if L is recognizable by some D-St-RNN[\mathbb{Q}, r_α], where
 400 $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ and $\alpha_i \in \{0, 1\}$ for each $i \in \mathbb{N}^*$, then $L \in BC(\Pi_2^0)(\alpha)$,
 401 where $\alpha = \alpha_1\alpha_2\alpha_3 \dots$.*

402 *Proof.* If L is recognized by some D-St-RNN[\mathbb{Q}, r], then a fortiori L is rec-
 403 ognized by some D-St-RNN[\mathbb{R}]. By Theorem 2, $L \in BC(\Pi_2^0)$. By Theorem
 404 2 again, L is recognized by some D-Ev₂-RNN[\mathbb{Q}], and by Remark 1, L is
 405 recognized by some D-Ev₂-RNN[\mathbb{Q}, α], for some $\alpha \in 2^\omega$. By Proposition 3,
 406 $L \in BC(\Pi_2^0)(\alpha)$.

Now, suppose that L is recognized by some D-St-RNN[\mathbb{Q}, r_α] \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.,

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

407 For each $n \geq 0$, let $\mathcal{N}|_{K.n}$ be the network \mathcal{N} whose weight r_α has been
 408 replaced by $r_\alpha|_{K.n}$, for some constant $K > 0$ defined in [12, Lemma 4.1]. By
 409 [12, Lemma 4.1], the truncated network $\mathcal{N}|_{K.n}$ computes precisely like \mathcal{N} up
 410 to time step n . Moreover, $\mathcal{N}|_{K.n}$ is a D-St-RNN[\mathbb{Q}], and thus, its behavior
 411 can be simulated by some Turing machine [8].

412 Consider the infinite procedure given by Algorithm 1 below. The pro-
 413 cedure consists in two subroutines performed in parallel. It receives some
 414 input $s \in (\mathbb{B}^M)^\omega$ together with the infinite word $\alpha \in 2^\omega$, and it simulates
 415 the computation of \mathcal{N} working on input s , by using the successive truncated
 416 networks $\mathcal{N}|_{K.n}$. All instructions of Algorithm 1 are recursive, and thus, can
 417 be simulated by some D-St-RNN[\mathbb{Q}] [8]. Hence, the whole Algorithm 1 can
 418 be simulated by some D-Ev₂-RNN[\mathbb{Q}, α] \mathcal{N}' which receives $\alpha = \alpha_1\alpha_2\alpha_3$ as
 419 an evolving bias. Every time \mathcal{N}' enters instruction 11 of Algorithm 1, it
 420 simulates the behavior of the truncated network $\mathcal{N}|_{K.n}$, and thus, by [12,

421 Lemma 4.1], reproduces the output pattern of \mathcal{N} working on input prefix
422 $\vec{u}(0) \cdots \vec{u}(n)$, to finally release the last output state of \mathcal{N} at last time step
423 n . But these successive computational periods of \mathcal{N}' are interspersed with
424 delays due to the simulation of the other instructions of Algorithm 1. In
425 order to deal with these delays, we provide \mathcal{N}' with an additional output
426 cell y_{P+1} which is programmed to be active only when the network simulates
427 the output period of instruction 11. Then, an attractor $A \subseteq \mathbb{B}^{P+1}$ of \mathcal{N}'
428 is defined to be accepting if and only if the $(P+1)$ -th component of each
429 element of A equals 1 (which corresponds to the cell y_{P+1} being active), and
430 the projection of A on \mathbb{B}^P is an accepting attractor of \mathcal{N} .

431 In this way, for any input $s \in (\mathbb{B}^M)^\omega$, the subsequence of the Boolean
432 computation of \mathcal{N}' induced by the active states of y_{P+1} is the same as the
433 Boolean computation of \mathcal{N} , and hence, s is accepting for \mathcal{N}' if and only
434 if s is accepting for \mathcal{N} . Consequently, $L(\mathcal{N}') = L(\mathcal{N})$. Since \mathcal{N}' is a D-
435 Ev₂-RNN[\mathbb{Q}, α], Proposition 3 ensures that $L(\mathcal{N}') \in BC(\Pi_2^0)(\alpha)$. Therefore,
436 $L(\mathcal{N}) \in BC(\Pi_2^0)(\alpha)$ too. \square

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

438 **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*
439 *$r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$. The following conditions are equivalent:*

- 440 (a) $L \in BC(\Pi_2^0)(\alpha)$;
- 441 (b) L is recognizable by some D-Ev₂-RNN[\mathbb{Q}, α];
- 442 (c) L is recognizable by some D-St-RNN[\mathbb{Q}, r_α].

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

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

443 Indeed, $L \in BC(\Pi_2^0)$ if and only if, by Remark 1, L is recognizable by
444 some D-Ev₂-RNN[\mathbb{Q}, α], for some $\alpha \in 2^\omega$, if and only if, by Theorem 3,
445 $L \in BC(\Pi_2^0)(\alpha)$. In words, the relativized classes $BC(\Pi_2^0)(\alpha)$ span the class
446 $BC(\Pi_2^0)$, when α varies over 2^ω .

447 4.2. Nondeterministic case

448 The expressive power of the classes of N-St-RNN[\mathbb{Q}], N-Ev₂-RNN[\mathbb{Q}], N-
449 Ev-RNN[\mathbb{Q}], N-Ev₂-RNN[\mathbb{R}] and N-Ev-RNN[\mathbb{R}] has been established in [19,
450 Theorems 1, 2]. We have the following results:

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 $t = 0, 1, 2, \dots$

2. infinite word $\alpha = \alpha_1\alpha_2\alpha_3\cdots \in 2^\omega$ supplied step by step at successive time steps $t = 0, 1, 2, \dots$

```
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  $n = 0, 1, 2, 3, \dots$  do
9:   wait that  $K \cdot n$  bits of  $\alpha$  have been stored
10:  compute  $r_\alpha|_{K \cdot n}$  // recursive if  $\alpha$  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[ $\mathbb{Q}$ ] [8]
12: end for
13: END SUBROUTINE 2
```

451 **Theorem 4.** [19, Theorems 1] Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language. The
452 following conditions are equivalent:

453 (a) $L \in \Sigma_1^1$;

454 (b) L is recognizable by some N -St-RNN[\mathbb{Q}];

455 (c) L is recognizable by some nondeterministic Muller Turing machine.

456 **Theorem 5.** [19, Theorems 2] Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language. The
457 following conditions are equivalent:

458 (a) $L \in \Sigma_1^1$;

459 (b) L is recognizable by some N -St-RNN[\mathbb{R}];

460 (c) L is recognizable by some N -Ev₂-RNN[\mathbb{Q}];

461 (d) L is recognizable by some N -Ev-RNN[\mathbb{Q}];

462 (e) L is recognizable by some $N\text{-Ev}_2\text{-RNN}[\mathbb{R}]$;

463 (f) L is recognizable by some $N\text{-Ev-RNN}[\mathbb{R}]$.

464 Theorem 4 states that $N\text{-St-RNN}[\mathbb{Q}]$ s are Turing equivalent. Theorem 5
 465 shows that all other classes of $N\text{-St-RNN}[\mathbb{R}]$ s, $N\text{-Ev}_2\text{-RNN}[\mathbb{Q}]$, $N\text{-Ev-RNN}[\mathbb{Q}]$,
 466 $N\text{-Ev}_2\text{-RNN}[\mathbb{R}]$ and $N\text{-Ev}_2\text{-RNN}[\mathbb{R}]$ are strictly more powerful than nonde-
 467 terministic Muller Turing machines, since $\Sigma_1^1 \subsetneq \Sigma_1^1$. In this sense, the non-
 468 deterministic analog and evolving neural networks are also *super-Turing*.

469 **Remark 2.** The nondeterministic counterpart of Remark 1 holds. More
 470 precisely, the proof of Theorem 5 [19, Theorem 2] shows that any ω -language
 471 $L \in \Sigma_1^1$ can be recognized by some $N\text{-St-RNN}[\mathbb{R}]$ employing at most one
 472 static irrational weight which is in the interval $[0, 1]$ and given in the form
 473 of a bias. Similarly, any ω -language $L \in \Sigma_1^1$ can be recognized by some $N\text{-}$
 474 $\text{Ev}_2\text{-RNN}[\mathbb{Q}]$ containing only one bi-valued evolving weight given as a bias.
 475 Consequently, from this point onwards, we will without loss of generality
 476 focus on the subclasses of $N\text{-St-RNN}[\mathbb{Q}, r]$ and $N\text{-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha]$, for any
 477 $r \in [0, 1]$ and $\alpha \in 2^\omega$.

478 We now provide a precise characterization of the expressive power of the
 479 two subclasses of $N\text{-St-RNN}[\mathbb{Q}, r]$ and $N\text{-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha]$, for any $r \in [0, 1]$
 480 and $\alpha \in 2^\omega$, respectively. This result is obtained via forthcoming Propositions
 481 5, 6, 7 and 8, which are direct generalizations of Propositions 1, 2, 3 and 4.

482 **Proposition 5.** *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language and $\alpha \in 2^\omega$. If $L \in$
 483 $\Sigma_1^1(\alpha)$, then L is recognizable by some $N\text{-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha]$.*

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

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

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

487 **Proposition 6.** *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language and $\alpha = \alpha_1\alpha_2\alpha_3 \in 2^\omega$.
 488 If $L \in \Sigma_1^1(\alpha)$, then L is recognizable by some $N\text{-St-RNN}[\mathbb{Q}, r_\alpha]$ \mathcal{N} , where
 489 $r = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$.*

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

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

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

492 Now, consider once again the real encoding of α given by $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in$
493 $[0, 1]$. Consider also the N-St-RNN[\mathbb{Q}, r_α] \mathcal{N} obtained by replacing the input
494 cell u_{M+1} of \mathcal{N}' by the real-weighted neural circuit C with bias r_α depicted
495 in Figure 3. One has $L(\mathcal{N}) = L'_{\alpha'} = L$, which shows that L is recognized by
496 the N-St-RNN[\mathbb{Q}, r_α] \mathcal{N} . \square

497 **Proposition 7.** *Let $L \subseteq (\mathbb{B}^M)^\omega$ be some ω -language and $\alpha \in 2^\omega$. If L is*
498 *recognizable by some N-Ev₂-RNN[\mathbb{Q}, α], then $L \in \Sigma_1^1(\alpha)$.*

499 *Proof.* Let \mathcal{N} be a N-Ev₂-RNN[\mathbb{Q}, α] such that $L(\mathcal{N}) = L$. By Remark 2, we
500 may assume without loss generality that the bi-valued evolving weight α of
501 \mathcal{N} is given as a bias. As in the proof of Proposition 3, we can construct from
502 \mathcal{N} a N-St-RNN[\mathbb{Q}] \mathcal{N}' with $P + 1$ input cells and one guess cell such that, for
503 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
504 shows that $L(\mathcal{N}) = L(\mathcal{N}')_\alpha$. Besides, Theorem 4 ensures that $L(\mathcal{N}') \in \Sigma_1^1$.
505 Therefore, $L(\mathcal{N}) \in \Sigma_1^1(\alpha)$. \square

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

511 *Proof.* If L is recognized by some N-St-RNN[\mathbb{Q}, r], then a fortiori L is recog-
512 nized by some N-St-RNN[\mathbb{R}]. By Theorem 5, $L \in \Sigma_1^1$. By Theorem 5 again,
513 L is recognized by some N-Ev₂-RNN[\mathbb{Q}], and by Remark 2, L is recognized
514 by some N-Ev₂-RNN[\mathbb{Q}, α], for some $\alpha \in 2^\omega$. By Proposition 7, $L \in \Sigma_1^1(\alpha)$.

515 Now, suppose that L is recognized by some N-St-RNN[\mathbb{Q}, r_α] \mathcal{N} , where
516 $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i}$ and $\alpha_i \in \{0, 1\}$ for each $i \in \mathbb{N}^*$. By Remark 2, we may
517 assume without loss of generality that the static weight r_α of \mathcal{N} is given
518 as a bias. Consider the infinite procedure given in previous Algorithm 1,

519 yet slightly modified in such a way that the algorithm receives as input a
520 guess stream $g \in 2^\omega$ provided bit by bit in addition to the input stream
521 $s \in (\mathbb{B}^M)^\omega$ and infinite word $\alpha \in 2^\omega$. This modified version of Algorithm
522 1 can be simulated by some N-Ev₂-RNN[\mathbb{Q}, α] \mathcal{N}' receiving g as a guess
523 stream and $\alpha = \alpha_1\alpha_2\alpha_3$ as an evolving bias. In addition, the accepting and
524 rejecting attractors of \mathcal{N}' are defined in the same way as in Proposition 4. By
525 construction, $L(\mathcal{N}') = L(\mathcal{N})$. Since \mathcal{N}' is a N-Ev₂-RNN[\mathbb{Q}, α], Proposition 7
526 ensures that $L(\mathcal{N}') \in \Sigma_1^1(\alpha)$. Therefore, $L(\mathcal{N}) \in \Sigma_1^1(\alpha)$ too. \square

527 By combining Propositions 5, 6, 7 and 8, the following theorem is ob-
528 tained:

529 **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*
530 $r_\alpha = \sum_{i=1}^{\infty} \frac{2\alpha_i+1}{4^i} \in [0, 1]$. *The following conditions are equivalent:*

- 531 (a) $L \in \Sigma_1^1(\alpha)$;
- 532 (b) L is recognizable by some N-Ev₂-RNN[\mathbb{Q}, α];
- 533 (c) L is recognizable by some N-St-RNN[\mathbb{Q}, r_α].

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

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

534 In other words, the relativized classes $\Sigma_1^1(\alpha)$ span the class Σ_1^1 , when α varies
535 over 2^ω .

536 5. The hierarchy theorem

537 Theorems 3 and 6 provide a precise characterization of the expressive
538 power of the classes of D-St-RNN[\mathbb{Q}, r_α], D-Ev₂-RNN[\mathbb{Q}, α], N-St-RNN[\mathbb{Q}, r_α]
539 and N-Ev₂-RNN[\mathbb{Q}, α], for any $\alpha \in 2^\omega$. We will show that these classes can
540 be stratified into transfinitely many subclasses based on the complexity of
541 the analog and evolving weights employed by the networks.

542 Towards this purpose, we first present some conditions that pairs of infi-
543 nite words necessarily satisfy whenever their corresponding relativized classes
544 are included one into the other.

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

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

$$\Sigma_1^1(\alpha) \subseteq \Sigma_1^1(\beta) \longleftrightarrow \alpha \in \Delta_1^1(\beta) \quad (4)$$

546 *Proof.* We prove both left-to-right implications. Recall that $\alpha \in \Sigma_1^0(\alpha)$. In
 547 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
 548 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 [41].

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

564 We now show the existence of an infinite sequence of infinite words whose
 565 corresponding succession of relativized classes properly stratify the “super-
 566 Turing” classes of $BC(\Pi_2^0)$ and Σ_1^1 neural ω -languages. In addition, the
 567 hierarchy induced by the inclusion relation between the relativized classes
 568 possesses chains of length ω_1 as well as uncountable antichains.

569 **Proposition 10.** *There exists a sequence $(\alpha_i)_{i < \omega_1}$, where $\alpha_i \in 2^\omega$ for all
 570 $i < \omega_1$, such that*

571 (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 $i <$
 572 $j < \omega_1$;

573 (b) $\Sigma_1^1(\alpha_0) = \Sigma_1^1$ and $\Sigma_1^1(\alpha_i) \subsetneq \Sigma_1^1(\alpha_j)$, for all $i < j < \omega_1$.

574 *Moreover, there exists some uncountable set $A \subseteq 2^\omega$ such that the following
 575 relations $BC(\Pi_2^0)(\alpha_i) \not\subseteq BC(\Pi_2^0)(\alpha_j)$ and $\Sigma_1^1(\alpha_i) \not\subseteq \Sigma_1^1(\alpha_j)$ hold, for every
 576 distinct $\alpha_i, \alpha_j \in A$.*

577 *Proof.* Take $\alpha_0 \in \Sigma_1^0$. Suppose that for $\gamma < \omega_1$, the sequence $(\alpha_i)_{i < \gamma}$ has been
578 constructed and satisfies the required property. We build the next element
579 α_γ of that sequence, i.e., the element such that $\Sigma_1^1(\alpha_i) \subsetneq \Sigma_1^1(\alpha_\gamma)$, for all $i < \gamma$.
580 Note that, for each $i < \gamma$, the set $\Delta_1^1(\alpha_i)$ is countable. Since $\gamma < \omega_1$, the union
581 $\bigcup_{i < \gamma} \Delta_1^1(\alpha_i)$ is countable too. Hence, there exists $\alpha \in 2^\omega \setminus \bigcup_{i < \gamma} \Delta_1^1(\alpha_i)$. Now,
582 let $\{\beta_i : i < \omega\}$ be an enumeration of the countable set $\{\alpha\} \cup \{\alpha_i : i < \gamma\}$,
583 and let $\alpha_\gamma \in 2^\omega$ be the encoding of $\{\beta_i : i < \omega\}$ given by $\alpha_\gamma(\langle i, n \rangle) = \beta_i(n)$,
584 where $\langle \cdot, \cdot \rangle : \omega^2 \rightarrow \omega$ is a classical recursive pairing function. Each function
585 $f_i : \alpha_\gamma \mapsto (\alpha_\gamma)_i = \beta_i$ is recursive, and therefore, $\beta_i \in \Sigma_1^0(\alpha_\gamma)$, for each $i < \omega$.

586 We show that $BC(\Pi_2^0)(\alpha_j) \subseteq BC(\Pi_2^0)(\alpha_\gamma)$, for all $j < \gamma$. Let $L \in$
587 $BC(\Pi_2^0)(\alpha_j) = BC(\Pi_2^0)(\beta_i)$, for some $i < \omega$. This means that L is recogniz-
588 able by some deterministic Muller TM \mathcal{M} with oracle β_i . Since $\beta_i \in \Sigma_1^0(\alpha_\gamma)$,
589 L is also recognized by the deterministic Muller TM \mathcal{M}' with oracle α_γ which,
590 in a suitable alternating manner, produces β_i bit by bit from α_γ , and works
591 precisely like \mathcal{M} with oracle β_i . Therefore, $L \in BC(\Pi_2^0)(\alpha_\gamma)$. By replacing in
592 this argument every occurrences of “ $BC(\Pi_2^0)$ ” by “ Σ_1^1 ” and of “deterministic”
593 by “nondeterministic”, one obtains that $\Sigma_1^1(\alpha_j) \subseteq \Sigma_1^1(\alpha_\gamma)$, for all $j < \gamma$.

594 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
595 all $j < \gamma$. Towards a contradiction, suppose that $BC(\Pi_2^0)(\alpha_\gamma) \subseteq BC(\Pi_2^0)(\alpha_j)$
596 or $\Sigma_1^1(\alpha_\gamma) \subseteq \Sigma_1^1(\alpha_j)$, for some $j < \gamma$. Then Relations (3) and (4) ensure that
597 $\alpha_\gamma \in \Delta_1^1(\alpha_j)$. But $\alpha = \beta_k$ for some $k < \omega$, and by the above stated fact,
598 $\alpha = \beta_k \in \Sigma_1^0(\alpha_\gamma)$. The two relations $\alpha \in \Sigma_1^0(\alpha_\gamma)$ and $\alpha_\gamma \in \Delta_1^1(\alpha_j)$ imply that
599 $\alpha \in \Delta_1^1(\alpha_j)$. This contradicts the fact that $\alpha \in 2^\omega \setminus \bigcup_{i < \gamma} \Delta_1^1(\alpha_i)$.

600 We finally prove the existence of an uncountable antichain. There exists
601 an uncountable set $A \subseteq 2^\omega$ such that $\alpha_i \notin \Delta_1^1(\alpha_j)$, for all distinct $\alpha_i, \alpha_j \in$
602 A [43]. By Relations (3) and (4), $BC(\Pi_2^0)(\alpha_i) \not\subseteq BC(\Pi_2^0)(\alpha_j)$ and $\Sigma_1^1(\alpha_i) \not\subseteq$
603 $\Sigma_1^1(\alpha_j)$, for all distinct $\alpha_i, \alpha_j \in A$. \square

604 Let $\mathcal{L}(\text{D-St-RNN}[\mathbb{Q}, r])$, $\mathcal{L}(\text{D-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha])$, $\mathcal{L}(\text{N-St-RNN}[\mathbb{Q}, r])$ and
605 $\mathcal{L}(\text{N-Ev}_2\text{-RNN}[\mathbb{Q}, \alpha])$ denote the classes of neural ω -languages recognized by
606 D-St-RNN $[\mathbb{Q}, r]$, D-Ev $_2$ -RNN $[\mathbb{Q}, \alpha]$, N-St-RNN $[\mathbb{Q}, r]$ and N-Ev $_2$ -RNN $[\mathbb{Q}, \alpha]$,
607 respectively. Theorems 3 and 6 together with Proposition 10 imply the exist-
608 ence of four proper hierarchies of classes of deterministic and nondetermin-
609 istic analog and evolving neural networks of increasing expressive power.

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

612 (a) $\mathcal{L}(\text{D-St-RNN}[\mathbb{Q}, r_i]) \subsetneq \mathcal{L}(\text{D-St-RNN}[\mathbb{Q}, r_j])$, for all $i < j < \omega_1$;

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

Proof. Theorems 3 and 6 ensure that

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

616 where r_α is the encoding of α described in Proposition 4, for any $\alpha \in 2^\omega$. By
617 Proposition 10, there exists some sequence $(\alpha_i)_{i < \omega_1}$ satisfying Points (b) and
618 (d). In addition, by taking $r_i = r_{\alpha_i}$ for all $i < \omega_1$, one obtains a sequence
619 $(r_i)_{i < \omega_1}$ satisfying Points (a) and (c). \square

Finally, let R be the equivalence relation defined by

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

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

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

625 *Proof.* According to Theorem 6 and Relation (4), the relation $R \subseteq 2^\omega \times 2^\omega$
626 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
627 relation “ $\alpha \in \Delta_1^1(\beta)$ ” is a Π_1^1 relation which can be expressed by a Π_1^1 -
628 formula $\phi(\alpha, \beta)$, see [41, 4D.14, p. 171] and [44]. Thus R is a Π_1^1 -relation.
629 Towards a contradiction, assume now that R is Σ_1^1 , and take $\beta \in \Sigma_1^0$. Then
630 $R(\cdot, \beta) = \{\alpha : R(\alpha, \beta)\} = \{\alpha : \alpha \in \Delta_1^1(\beta) \ \& \ \beta \in \Delta_1^1(\alpha)\} = \{\alpha : \alpha \in$
631 $\Delta_1^1(\beta)\} = \{\alpha : \alpha \in \Delta_1^1\}$ should also be in Σ_1^1 . But it is known that the set
632 $\{\alpha : \alpha \in \Delta_1^1\}$ is not Σ_1^1 , see [41, 4D.16, p. 171]. This concludes the proof. \square

633 6. Conclusion

634 The present study concerns the expressive power of sigmoidal recurrent
635 neural networks involved in a computational paradigm based on infinite

636 rather than finite input streams. This approach conciliates two important bi-
637 ological and computer scientist perspectives about neural attractor dynamics
638 and non-terminating computational processes, respectively.

639 In this context, we provided a full characterization of the expressive power
640 of the networks. For any $\alpha \in 2^\omega$ with corresponding encoding $r_\alpha \in \mathbb{R}$, the
641 deterministic and nondeterministic analog or evolving networks employing
642 either the single static analog weight r_α or the single evolving weight α rec-
643 ognize the (lightface) relativized topological classes of $BC(\Pi_2^0)(\alpha)$ and $\Sigma_1^1(\alpha)$
644 ω -languages, respectively. As a consequence, two infinite refined hierarchies
645 of classes of analog and evolving neural nets based on the complexity of their
646 underlying analog and evolving weights are obtained. These hierarchies rep-
647 resent a generalization to the context of ω -computation of the fundamental
648 previous hierarchy of classes of analog networks based on the Kolmogorov
649 complexity of their underlying analog weights [13].

650 From a purely theoretical perspective, these results show that analog and
651 evolving neural networks constitute natural equivalent models for oracle-
652 based infinite computation, beyond the Turing limits. In the analog case,
653 the extra-recursive power of the networks arises from their possibility to
654 have access to more and more precise rational approximations of some given
655 real weights [12]. In the evolving case, the extra capabilities emerge from
656 the non-recursive patterns of evolution of the synapses [17]. Despite their
657 mathematical equivalence, the two neural models are conceptually distinct:
658 while the former remains at a purely conceptual level, the later relies on
659 considerations that could be observable in nature.

660 From a more practical point of view, the two phenomena of attractor dy-
661 namics and synaptic plasticity are of primordial importance to the processing
662 and coding of information in both artificial and biological neural networks.
663 In fact, the concept of an attractor has been shown to carry strong com-
664 putational implications. According to Kauffman: “Because many complex
665 systems harbour attractors to which the system settle down, the attractors
666 literally are most of what the systems do” [45, p.191]. In the neural net-
667 work context, alternative attractors are commonly interpreted as alternative
668 memories, but have also been associated to motor behaviors, perceptions and
669 thoughts [46, 47, 48, 49, 32, 50]. Likewise, synaptic plasticity is known to be
670 crucially related to the storage and encoding of memory traces in the cen-
671 tral nervous system, and provides the basis for most models of learning and
672 memory in neural networks [36, 37, 38, 39]. In view of these considerations,

673 our results may constitute a theoretical foundation of the computational ca-
674 pabilities of neural networks in touch with these two crucial phenomena.

675 More generally, this study strengthen the connectedness between the
676 fields of theoretical computer science, with possible extensions to the more
677 practical domain of machine learning, and theoretical neuroscience. We hope
678 that such comparative studies between neural networks and abstract ma-
679 chines might eventually bring further insight to the understanding of both
680 biological and artificial intelligences. Similarly to the foundational work of
681 Turing, which played a crucial role in the practical realization of modern com-
682 puters, further theoretical considerations about neural- and natural-based
683 models of computation might contribute to the emergence of novel computa-
684 tional technologies, and step by step, open the way to the next computational
685 generation.

686 Acknowledgments

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

689 References

- 690 [1] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in
691 nervous activity, *Bulletin of Mathematical Biophysic* 5 (1943) 115–133.
- 692 [2] S. C. Kleene, Representation of events in nerve nets and finite automata,
693 in: C. Shannon, J. McCarthy (Eds.), *Automata Studies*, Princeton Uni-
694 versity Press, Princeton, NJ, 1956, pp. 3–41.
- 695 [3] M. L. Minsky, *Computation: finite and infinite machines*, Prentice-Hall,
696 Inc., Englewood Cliffs, N. J., 1967.
- 697 [4] H. T. Siegelmann, Recurrent neural networks and finite automata, *Com-
698 putational Intelligence* 12 (1996) 567–574.
- 699 [5] A. M. Turing, *Intelligent machinery*, Technical report, National Physical
700 Laboratory, Teddington, UK (1948).
- 701 [6] J. B. Pollack, *On connectionist models of natural language processing*,
702 Ph.D. thesis, Computing Research Laboratory, New Mexico State Uni-
703 versity, Las Cruces, NM (1987).

- 704 [7] R. Hartley, H. Szu, A comparison of the computational power of neural
705 network models, in: C. Butler (Ed.), Proceedings of the IEEE First
706 International Conference on Neural Networks, IEEE, 1987, pp. 17–22.
- 707 [8] H. T. Siegelmann, E. D. Sontag, On the computational power of neural
708 nets, *J. Comput. Syst. Sci.* 50 (1) (1995) 132–150.
- 709 [9] J. Kilian, H. T. Siegelmann, The dynamic universality of sigmoidal neu-
710 ral networks, *Inf. Comput.* 128 (1) (1996) 48–56.
- 711 [10] H. Hyötyniemi, Turing machines are recurrent neural networks, in:
712 J. Alander, T. Honkela, J. M. (Eds.), STeP '96 - Genes, Nets and
713 Symbols; Finnish Artificial Intelligence Conference, Vaasa 20-23 Aug.
714 1996, University of Vaasa, Finnish Artificial Intelligence Society (FAIS),
715 Vaasa, Finland, 1996, pp. 13–24, sTeP '96 - Genes, Nets and Symbols;
716 Finnish Artificial Intelligence Conference, Vaasa, Finland, 20-23 August
717 1996.
- 718 [11] J. a. P. G. Neto, H. T. Siegelmann, J. F. Costa, C. P. S. Araujo, Turing
719 universality of neural nets (revisited), in: EUROCAST '97: Proceedings
720 of the A Selection of Papers from the 6th International Workshop on
721 Computer Aided Systems Theory, Springer-Verlag, London, UK, 1997,
722 pp. 361–366.
- 723 [12] H. T. Siegelmann, E. D. Sontag, Analog computation via neural net-
724 works, *Theor. Comput. Sci.* 131 (2) (1994) 331–360.
- 725 [13] J. L. Balcázar, R. Gavaldà, H. T. Siegelmann, Computational power of
726 neural networks: a characterization in terms of kolmogorov complexity,
727 *IEEE Transactions on Information Theory* 43 (4) (1997) 1175–1183.
- 728 [14] H. T. Siegelmann, *Neural networks and analog computation: beyond
729 the Turing limit*, Birkhauser Boston Inc., Cambridge, MA, USA, 1999.
- 730 [15] H. T. Siegelmann, Neural and super-Turing computing, *Minds Mach.*
731 13 (1) (2003) 103–114.
- 732 [16] J. Cabessa, H. T. Siegelmann, Evolving recurrent neural networks are
733 super-Turing, in: Proceedings of IJCNN 2011, IEEE, 2011, pp. 3200–
734 3206.

- 735 [17] J. Cabessa, H. T. Siegelmann, The super-Turing computational power
736 of plastic recurrent neural networks, *Int. J. Neural Syst.* 24 (8).
- 737 [18] J. Síma, P. Orponen, General-purpose computation with neural net-
738 works: A survey of complexity theoretic results, *Neural Computation*
739 15 (12) (2003) 2727–2778.
- 740 [19] J. Cabessa, J. Duparc, Expressive power of nondeterministic recurrent
741 neural networks in terms of their attractor dynamics, *IJUC* 12 (1) (2016)
742 25–50.
- 743 [20] J. Cabessa, A. E. P. Villa, Expressive power of first-order recurrent neu-
744 ral networks determined by their attractor dynamics, *Journal of Com-
745 puter and System Sciences* 82 (8) (2016) 1232–1250.
- 746 [21] N. Kasabov, *Evolving connectionist systems - the knowledge engineering
747 approach* (2. ed.), Springer, 2007.
- 748 [22] K. O. Stanley, R. Miikkulainen, Evolving neural network through aug-
749 menting topologies, *Evolutionary Computation* 10 (2) (2002) 99–127.
- 750 [23] J. Cabessa, H. T. Siegelmann, The computational power of interactive
751 recurrent neural networks, *Neural Computation* 24 (4) (2012) 996–1019.
- 752 [24] J. Cabessa, A. E. P. Villa, A hierarchical classification of first-order
753 recurrent neural networks, in: A. H. D. et al. (Ed.), *Proceedings of
754 LATA 2010*, Vol. 6031 of *Lecture Notes in Computer Science*, Springer,
755 2010, pp. 142–153.
- 756 [25] J. Cabessa, A. E. P. Villa, The expressive power of analog recurrent neu-
757 ral networks on infinite input streams, *Theor. Comput. Sci.* 436 (2012)
758 23–34.
- 759 [26] J. Cabessa, A. E. P. Villa, The super-Turing computational power of
760 interactive evolving recurrent neural networks, in: V. M. et al. (Ed.),
761 *Proceedings of ICANN 2013*, Vol. 8131 of *Lecture Notes in Computer
762 Science*, Springer, 2013, pp. 58–65.
- 763 [27] J. Cabessa, A. E. P. Villa, Interactive evolving recurrent neural networks
764 are super-Turing universal, in: S. W. et al. (Ed.), *Proceedings of ICANN
765 2014*, Vol. 8681 of *Lecture Notes in Computer Science*, Springer, 2014,
766 pp. 57–64.

- 767 [28] J. Cabessa, A. E. P. Villa, An attractor-based complexity measurement
768 for boolean recurrent neural networks, PLoS ONE 9 (4) (2014) e94204+.
- 769 [29] J. Cabessa, A. E. P. Villa, Computational capabilities of recurrent neural
770 networks based on their attractor dynamics, in: 2015 International Joint
771 Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July
772 12-17, 2015, IEEE, 2015, pp. 1–8.
- 773 [30] J. Cabessa, A. E. P. Villa, Recurrent neural networks and super-Turing
774 interactive computation, in: P. Koprinkova-Hristova, V. Mladenov,
775 K. N. Kasabov (Eds.), Artificial Neural Networks: Methods and Ap-
776 plications in Bio-/Neuroinformatics, Springer, 2015, pp. 1–29.
- 777 [31] J. Cabessa, J. Duparc, Expressive power of non-deterministic evolving
778 recurrent neural networks in terms of their attractor dynamics, in: C. S.
779 Calude, M. J. Dinneen (Eds.), Unconventional Computation and Nat-
780 ural Computation - 14th International Conference, UCNC 2015, Auck-
781 land, New Zealand, August 30 - September 3, 2015, Proceedings, Vol.
782 9252 of Lecture Notes in Computer Science, Springer, 2015, pp. 144–156.
- 783 [32] D. J. Amit, Modeling brain function: The world of attractor neural
784 networks, Cambridge University Press, 1992.
- 785 [33] W. Thomas, Automata on infinite objects, in: J. van Leeuwen (Ed.),
786 Handbook of Theoretical Computer Science, Volume B: Formal Models
787 and Semantics, Elsevier and MIT Press, 1990, pp. 133–192.
- 788 [34] D. Perrin, J.-E. Pin, Infinite Words – Automata, Semigroups, Logic and
789 Games, Vol. 141 of Pure and Applied Mathematics, Elsevier, 2004.
- 790 [35] J. Cabessa, O. Finkel, [Expressive power of evolving neural networks
791 working on infinite input streams](#), in: R. Klasing, M. Zeitoun (Eds.),
792 Fundamentals of Computation Theory - 21st International Symposium,
793 FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings, Vol.
794 10472 of Lecture Notes in Computer Science, Springer, 2017, pp. 150–
795 163. doi:10.1007/978-3-662-55751-8_13.
796 URL https://doi.org/10.1007/978-3-662-55751-8_13
- 797 [36] L. F. Abbott, S. B. Nelson, Synaptic plasticity: taming the beast, Nat.
798 Neurosci. 3 Suppl. (2000) 1178–1183.

- 799 [37] S. J. Martin, P. D. Grimwood, R. G. M. Morris, Synaptic plasticity and
800 memory: An evaluation of the hypothesis, *Annu. Rev. Neurosci.* 23 (1)
801 (2000) 649–711.
- 802 [38] P. D. Roberts, C. C. Bell, Spike timing dependent synaptic plasticity in
803 biological systems, *Biol. Cybern.* 87 (2002) 392–403.
- 804 [39] N. Caporale, Y. Dan, Spike timing-dependent plasticity: a Hebbian
805 learning rule, *Annu. Rev. Neurosci.* 31 (2008) 25–46.
- 806 [40] A. S. Kechris, Classical descriptive set theory, Vol. 156 of Graduate
807 Texts in Mathematics, Springer-Verlag, New York, 1995.
- 808 [41] Y. N. Moschovakis, Descriptive Set Theory, 2nd Edition, Mathematical
809 surveys and monographs, American Mathematical Society, 2009.
- 810 [42] L. Staiger, ω -languages, in: Handbook of formal languages, vol. 3: be-
811 yond words, Springer-Verlag New York, Inc., New York, NY, USA, 1997,
812 pp. 339–387.
- 813 [43] K. R. Apt, ω -models in analytical hierarchy, *Bulletin de l’académie*
814 *polonaise des sciences* XX (11) (1972) 901–904.
- 815 [44] O. Finkel, Ambiguity of omega-languages of Turing machines, *Logical*
816 *Methods in Computer Science* 10 (3).
- 817 [45] S. A. Kauffman, The origins of order: Self-organization and selection in
818 evolution, Oxford University Press, New York, 1993.
- 819 [46] W. A. Little, The existence of persistent states in the brain, *Mathemat-*
820 *ical biosciences* 19 (1974) 101–120.
- 821 [47] W. A. Little, G. L. Shaw, Analytical study of the memory storage ca-
822 pacity of a neural network, *Mathematical biosciences* 39 (1978) 281–290.
- 823 [48] J. J. Hopfield, Neural networks and physical systems with emergent
824 collective computational abilities, *Proceedings of the National Academy*
825 *of Sciences* 79 (1982) 2554–2558.
- 826 [49] J. J. Hopfield, Neurons with graded response have collective compu-
827 tational properties like those of two-state neurons, *Proceedings of the*
828 *National Academy of Sciences* 81 (10) (1984) 3088–3092.

- 829 [50] C. Eliasmith, A unified approach to building and controlling spiking
830 attractor networks, *Neural Comput.* 17 (6) (2005) 1276–1314.