Counting with Analog Neurons

Jiří Šíma*

Institute of Computer Science, Czech Academy of Sciences,
P. O. Box 5, 18207 Prague 8, Czech Republic, sima@cs.cas.cz

Abstract. We refine the analysis of binary-state neural networks with α extra analog neurons (αANNs). For rational weights, it has been known that online 1ANNs accept context-sensitive languages including examples of non-context-free languages, while offline 3ANNs are Turing complete. We now prove that the deterministic (context-free) language containing the words of n zeros followed by n ones, cannot be recognized offline by any 1ANN with real weights. Hence, the offline 1ANNs are not Turing complete. On the other hand, we show that any deterministic language can be accepted by a 2ANN with rational weights. Thus, two extra analog units can count to any number which is not the case of one analog neuron.

Keywords: Neural computing · Analog state · Deterministic pushdown automaton · Deterministic context-free language · Chomsky hierarchy.

1 Introduction

The computational power of (recurrent) neural networks with the saturated-linear activation function[1] depends on the descriptive complexity of their weight parameters [10, 18]. Neural nets with *integer* weights, corresponding to binary-state networks, coincide with finite automata [1, 3, 4, 7, 14, 20]. *Rational* weights make the analog-state networks computationally equivalent to Turing machines [4, 12], and thus (by a real-time simulation [12]) polynomial-time computations of such networks are characterized by the fundamental complexity class P. Moreover, neural nets with arbitrary *real* weights can even derive "super-Turing" computational capabilities [10]. In particular, their polynomial-time computations correspond to the nonuniform complexity class P/poly while any input/output mapping (including undecidable problems) can be computed within exponential time [11]. In addition, a proper hierarchy of nonuniform complexity classes between P and P/poly has been established for polynomial-time computations of neural nets with increasing Kolmogorov complexity of real weights [2].

As can be seen, our understanding of the computational power of recursive (Turing complete) neural networks is satisfactorily fine-grained when changing from rational to arbitrary real weights. In contrast, there is still a gap between

* Research was done with institutional support RVO: 67985807 and partially supported by the grant of the Czech Science Foundation No. 19-05704S.

[1] Some of the results are valid for more general classes of activation functions [6, 13, 21] including the logistic function [5].

integer and rational weights which results in a jump from regular to recursively enumerable languages in the Chomsky hierarchy. In the effort of refining the analysis of subrecursive neural nets we have introduced a model of binary-state networks extended with α extra analog-state neurons (αANNs) [15], as already *three* additional analog units allow for Turing universality [16]. Although this model of αANNs has been inspired by theoretical issues, neural networks with different types of units/layers are widely used in practical applications, e.g. in deep learning [9], and they thus require a detailed mathematical analysis.

In our previous work, we have characterized syntactically the class of languages accepted online by the 1ANNs with one extra analog neuron [17] in terms of so-called cut languages [19]. The *online* (real-time) input/output protocol means that a (potentially infinite) input word $\mathbf{x}$ is sequentially read symbol after symbol, each being processed with a *constant-time overhead*, while a neural network simultaneously signals via its output neuron whether the prefix of $\mathbf{x}$ that has been read so far, belongs to the respective language [3, 20]. By using the underlying syntactic characterization we have shown that the languages recognized online by the 1ANNs with rational weights are context-sensitive, and we have presented explicit examples of such languages that are not context-free. Furthermore, we have formulated a sufficient condition when a 1ANN accepts only a regular language in terms of quasi-periodicity of its real weight parameters. For example, 1ANNs with weights from the smallest field extension $\mathbb{Q}(\beta)$ over rational numbers including a Pisot number $\beta = 1/w$ where w is the self-loop weight of the analog unit, have only a power of finite automata. These results [17] refine the classification of subrecursive neural networks with the weights between integer and rational weights, within the Chomsky hierarchy.

In addition, we have shown [16] that any language accepted by a Turing machine in time $T(n)$ can be accepted offline by a binary-state neural network with *three* extra analog units (3ANNs) having rational weights in time $O(T(n))$. The *offline* input/output protocol assumes that an input word $\mathbf{x}$ is read by a neural network either sequentially, each symbol on request with no time bounds for its processing (possibly with the recognition of each input prefix), or $\mathbf{x}$ is already encoded in a real initial state of the analog neuron. The neural network then carries out its computation until it possibly halts and decides whether $\mathbf{x}$ belongs to the underlying language, which is indicated by its output neurons [12]. Thus, for rational weights, the languages accepted online by 1ANNs or offline by 3ANNs are context-sensitive or recursively enumerable, respectively.

In this paper, we further refine the analysis of αANNs by showing that the deterministic (context-free) language $L = \{0^n 1^n \mid n \geq 1\}$, containing the words of n zeros followed by n ones, cannot be recognized even offline by any 1ANN with one extra analog neuron having arbitrary real weights. Hence, the offline 1ANNs are not Turing complete. The proof is based on an asymptotic analysis of computations by 1ANNs whose dynamics is quite restricted for recalling a stored number. It follows that 1ANNs cannot count to an arbitrarily large number, although we know 1ANNs with rational weights accept some context-sensitive cut languages that are not context-free [17].

On the other hand, we prove that the deterministic languages (including L) which are accepted by the deterministic pushdown automata (DPDA) can be recognized by 2ANNs with *two* extra analog neurons and rational weights. This means that two extra analog units can count to any number which is not the case of one analog neuron. The proof exploits the classical technique of implementing the stack of a DPDA by analog units, including the encoding of stack contents based on a Cantor-like set [12]. The synchronization of a fully parallel computation by 2ANNs is implemented by alternating the storage of stack contents between two analog neurons so that the first neuron realizes the `top` and `push` operations while the second one carries out the `pop` operation.

The paper is organized as follows. In Section 2, we introduce a formal model of binary-state neural networks αANNs with α extra analog units. Section 3 shows an example of a deterministic language that cannot be recognized by any 1ANN with real weights, while Section 4 presents a simulation of any DPDA by a 2ANN with rational weights. We present some open problems in Section 5.

2 Neural Networks with a Few Extra Analog Units

For a small integer constant $\alpha \geq 0$ (e.g. $\alpha \leq 3$), we specify a computational model of a *binary-state neural network αANN with α extra analog units*, $\mathcal{N}$, which will be used as a formal language acceptor. The network $\mathcal{N}$ consists of $s \geq \alpha$ *units (neurons)*, indexed as $V = \{1, \ldots, s\}$. The units in $\mathcal{N}$ are assumed to be binary-state (shortly *binary*) neurons (i.e. *perceptrons, threshold gates)* except for the first α neurons $1, \ldots, \alpha \in V$ which are *analog* units. The neurons are connected into a directed graph representing an *architecture* of $\mathcal{N}$, in which each edge $(i, j) \in V^2$ leading from unit i to j is labeled with a real *weight* $w(i, j) = w_{ji} \in \mathbb{R}$. The absence of a connection within the architecture corresponds to a zero weight between the respective neurons, and vice versa.

The *computational dynamics* of $\mathcal{N}$ determines for each unit $j \in V$ its *state (output)* $y_j^{(t)}$ at discrete time instants $t = 0, 1, 2, \ldots$. The outputs $y_1^{(t)}, \ldots, y_\alpha^{(t)}$ from analog units $1, \ldots, \alpha \in V$ are real numbers from the unit interval $\mathbb{I} = [0, 1]$, whereas the states $y_j^{(t)}$ of the remaining $s - \alpha$ neurons $j \in V' = V \setminus \{1, \ldots, \alpha\}$ are binary values from $\{0, 1\}$. This establishes the *network state* $\mathbf{y}^{(t)} = \left(y_1^{(t)}, \ldots, y_\alpha^{(t)}, y_{\alpha+1}^{(t)}, \ldots, y_s^{(t)}\right) \in \mathbb{I}^\alpha \times \{0, 1\}^{s-\alpha}$ at each discrete time instant $t \geq 0$. For notational simplicity, we assume a synchronous fully parallel mode without loss of efficiency [8]. At the beginning of a computation, the neural network $\mathcal{N}$ is placed in an *initial state* $\mathbf{y}^{(0)} \in \{0, 1\}^s$. At discrete time instant $t \geq 0$, an *excitation* of any neuron $j \in V$ is defined as $\xi_j^{(t)} = \sum_{i=0}^{s} w_{ji} y_i^{(t)}$, including a real *bias* value $w_{j0} \in \mathbb{R}$ which can be viewed as the weight $w(0, j)$ from a formal constant unit input $y_0^{(t)} \equiv 1$ for every $t \geq 0$ (i.e. $0 \in V'$). At the next instant $t+1$, all the neurons $j \in V$ compute their new outputs $y_j^{(t+1)}$ in parallel by applying an *activation function* $\sigma_j : \mathbb{R} \longrightarrow \mathbb{I}$ to $\xi_j^{(t)}$, that is, $y_j^{(t+1)} = \sigma_j\left(\xi_j^{(t)}\right)$ for $j \in V$. The analog units $j \in \{1, \ldots, \alpha\}$ employ the *saturated-linear* function $\sigma_j(\xi) = \sigma(\xi)$

where $\sigma(\xi) = \xi$ for $0 \leq \xi \leq 1$, while $\sigma(\xi) = 1$ for $\xi > 1$, and $\sigma(\xi) = 0$ for $\xi < 0$. For neurons $j \in V'$ with binary states $y_j \in \{0,1\}$, the *Heaviside* activation function $\sigma_j(\xi) = H(\xi)$ is used where $H(\xi) = 1$ for $\xi \geq 0$ and $H(\xi) = 0$ for $\xi < 0$. This determines the new network state $\mathbf{y}^{(t+1)} \in \mathbb{I}^\alpha \times \{0,1\}^{s-\alpha}$ at time $t+1$.

The computational power of neural networks has been studied analogously to the traditional models of computations so that the networks are exploited as acceptors of formal languages $L \subseteq \Sigma^*$ [18]. For simplicity, we assume the binary alphabet $\Sigma = \{0,1\}$ and for a finite αANN $\mathcal{N}$, we use the following offline input/output protocol employing its special neurons $\mathrm{nxt}, \mathrm{inp}, \mathrm{out} \in V'$. An input word (string) $\mathbf{x} = x_1 \dots x_n \in \{0,1\}^n$ of arbitrary length $n \geq 0$, is sequentially presented to $\mathcal{N}$, bit after bit, via the so-called *input neuron* $\mathrm{inp} \in V'$, at the time instants $0 < \tau_1 < \tau_2 < \cdots < \tau_n$ when queried by $\mathcal{N}$. Thus, once the prefix $x_1, \dots, x_{k-1}$ of $\mathbf{x}$ for $1 \leq k \leq n$, has been read, the next input bit $x_k \in \{0,1\}$ is presented to $\mathcal{N}$ one computational step after $\mathcal{N}$ activates the neuron $\mathrm{nxt} \in V'$, that is, $y_{\mathrm{inp}}^{(t)} = x_k$ and $y_{\mathrm{nxt}}^{(t-1)} = 1$ if $t = \tau_k$, and $y_{\mathrm{inp}}^{(t)} = y_{\mathrm{nxt}}^{(t-1)} = 0$, otherwise, for $k = 1, \dots, n$. At the same time, $\mathcal{N}$ carries its computation deciding about each prefix of the input word $\mathbf{x}$ whether it belongs to L, which is indicated by the output neuron $\mathrm{out} \in V'$ when the neuron nxt is active, i.e. $y_{\mathrm{out}}^{(\tau_{k+1}-1)} = 1$ if $x_1 \dots x_k \in L$, and $y_{\mathrm{out}}^{(\tau_{k+1}-1)} = 0$ if $x_1 \dots x_k \notin L$, where $\tau_{n+1} > \tau_n$ is the time instant when the input word $\mathbf{x}$ is decided. We say that a language $L \subseteq \{0,1\}^*$ is *accepted (recognized)* by αANN $\mathcal{N}$, which is denoted by $L = \mathcal{L}(\mathcal{N})$, if for any input word $\mathbf{x} \in \{0,1\}^*$, $\mathbf{x} \in L$ iff $\mathcal{N}$ halts and accepts $\mathbf{x}$.

3 One Analog Neuron Cannot Count

In this section, we prove that the deterministic language L, containing the words of n zeros followed by n ones, which imitates counting, cannot be accepted by any 1ANN $\mathcal{N}$ with one extra analog unit. The main idea of the proof is based on the fact that $\mathcal{N}$ must keep the count of the initial segment of zeros in an input word because this must later be compared to the number of subsequent ones in order to decide whether the input is accepted. However, this count is unbounded while $\mathcal{N}$ has only finitely many possible binary states. Thus, this number can just be stored by using a real state of the analog neuron. On the contrary, suppose $L = \mathcal{L}(\mathcal{N})$ is accepted by $\mathcal{N}$. By presenting a series of zeros as an input to $\mathcal{N}$, we obtain an infinite bounded sequence of real analog-state values which has a monotone convergent subsequence according to the Bolzano-Weierstrass theorem. This subsequence is further pruned so that it remains infinite while the following condition is satisfied. Starting with any analog value from this pruned convergent subsequence, the binary states enter the same cycle in a while after a subsequent series of ones is presented to $\mathcal{N}$, which induces a periodic behavior in the limit. This periodicity provides only a finite number of thresholds for separating an infinite number of analog values from each other, which represent the counts of zeros. This means that $\mathcal{N}$ would accept two input words composed of different number of zeros followed by the same number of ones, which is a contradiction. The technical details are presented in the following proof sketch.

Theorem 1. *The deterministic context-free language* $L = \{0^n 1^n \,|\, n \geq 1\}$ *cannot be recognized by a neural network 1ANN with one extra analog unit having real weights.*

Proof. (Sketch.) On the contrary, assume that $\mathcal{N}$ is a neural network 1ANN with one extra analog unit such that $L = \mathcal{L}(\mathcal{N})$. Let $y_j^{(t)}(\mathbf{x})$ and $\xi_j^{(t)}(\mathbf{x})$ be the state and the excitation of neuron $j \in V$ at time instant $t \geq 0$, respectively, when an input word $\mathbf{x} \in \{0,1\}^n$ of length n is presented to $\mathcal{N}$, which satisfies $t < \tau_{n+1}$ by the input protocol (formally, we also allow infinite input strings $\mathbf{x} \in \{0,1\}^\omega$). Denote by $\mathbf{y}^{(t)}(\mathbf{x}) = \left(y_1^{(t)}(\mathbf{x}), \ldots, y_s^{(t)}(\mathbf{x})\right) \in \mathbb{I} \times \{0,1\}^{s-1}$ and $\tilde{\mathbf{y}}^{(t)}(\mathbf{x}) = \left(y_2^{(t)}(\mathbf{x}), \ldots, y_s^{(t)}(\mathbf{x})\right) \in \{0,1\}^{s-1}$ the corresponding network state, respectively, restricted to binary neurons. For the infinite input string 0^ω, there exists $t_0 \geq 0$ such that the state of analog unit meets $y_1^{(t_0)}(0^\omega) \in \{0,1\}$ (we know $y_1^{(0)}(0^\omega) \in \{0,1\}$ by definition) and $0 < y_1^{(t)}(0^\omega) < 1$ for every $t > t_0$, since otherwise there would be infinitely many time instants t with the same network state $\mathbf{y}^{(t)}(0^\omega)$ due to $\{0,1\}^s$ is finite, which provides $n_1 < n_2$ such that $\mathcal{N}$ would accept incorrectly the input word $0^{n_2}1^{n_1} \notin L$. For the same reason, the self-loop weight meets $w_{11} \neq 0$ since for $w_{11} = 0$, the analog unit could produce only a finite number of output values $y_1^{(t)} \in \left\{ \sum_{i \in V'} w_{1i} y_i \,\middle|\, (y_2, \ldots, y_s) \in \{0,1\}^{s-1} \right\}$ for $t > t_0$. Define the *base* $\beta = 1/w_{11}$ and the set of *digits*, $A = \left\{ \beta \sum_{i \in V'} w_{1i} y_i \,\middle|\, (y_2, \ldots, y_s) \in \{0,1\}^{s-1} \right\} \cup \{0, \beta\}$. We introduce an infinite sequence of digits, $a_1 a_2 a_3 \ldots \in A^\omega$ as $a_1 = \beta\, y_1^{(t_0)}(0^\omega) \in \{0, \beta\} \subseteq A$ and $a_k = \beta \sum_{i \in V'} w_{1i} y_i^{(t_0+k-2)}(0^\omega) \in A$ for $k \geq 2$. For every $t \geq t_0$, we have $y_1^{(t+1)}(0^\omega) = \sum_{i=0}^{s} w_{1i} y_i^{(t)}(0^\omega) = \beta^{-1} \left(a_{t-t_0+2} + y_1^{(t)}(0^\omega) \right)$, which implies $y_1^{(t)}(0^\omega) = \sum_{k=1}^{t-t_0+1} a_{t-t_0-k+2}\, \beta^{-k}$. It follows that $|\beta| > 1$ because $0 < y_1^{(t)}(0^\omega) < 1$ for every $t > t_0$.

Consider an infinite sequence of time instants $0 < t_1 < t_2 < t_3 < \cdots$ such that for each n, $t_n = \tau_{n+1} - 1$ is the last time instant before the next $(n+1)$th bit is presented to $\mathcal{N}$ after the input 0^n has been read, that is, $y_{\text{nxt}}^{(t_n)}(0^n) = 1$. Since the infinite sequence of real numbers $y_1^{(t_n)}(0^n) \in \mathbb{I}$ for $n \geq 1$, is bounded, there exists its monotone convergent subsequence $y_1^{(t_{n_p})}(0^{n_p}) \in (0\,,1)$ for $p \geq 1$, where $t_{n_1} > t_0$, $n_1 < n_2 < n_3 < \cdots$, and $c_0 = \lim_{p \to \infty} y_1^{(t_{n_p})}(0^{n_p})$, according to Bolzano-Weierstrass theorem. We assume that this subsequence is nondecreasing, that is, $y_1^{(t_{n_p})}(0^{n_p}) \leq y_1^{(t_{n_{p+1}})}(0^{n_{p+1}})$ for every $p \geq 1$, while the argument for a nonincreasing subsequence is analogous. In the following considerations, we will repeatedly remove some elements from the sequence (n_p) given by Bolzano-Weierstrass theorem, so that infinitely many elements remain, which satisfy additional conditions. For simplicity, we will keep the original notation (n_p) for these pruned sequences without loss of generality.

There are only finitely many possible states of binary neurons taken from $\{0,1\}^{s-1}$, and hence, there exists $\tilde{\mathbf{u}} \in \{0,1\}^{s-1}$ which occurs infinitely many times in the corresponding subsequence $\tilde{\mathbf{y}}^{(t_{n_p})}(0^{n_p})$ for $p \geq 1$. By skipping the remaining elements, we can assume without loss of generality that $\tilde{\mathbf{y}}^{(t_{n_p})}(0^{n_p}) = \tilde{\mathbf{u}}$

for every $p \geq 1$. It follows that the subsequence $y_1^{(t_{n_p})}(0^{n_p})$ for $p \geq 1$, is increasing since for $y_1^{(t_{n_p})}(0^{n_p}) = y_1^{(t_{n_{p+1}})}(0^{n_{p+1}})$, we have $\mathbf{y}^{(t_{n_p})}(0^{n_p}) = \mathbf{y}^{(t_{n_p})}(0^{n_{p+1}})$, and hence, the input $0^{n_{p+1}}1^{n_p} \notin L$ would be incorrectly accepted by $\mathcal{N}$.

We will inductively construct an increasing infinite sequence (m_p) of natural numbers $m_p \geq 0$ such that for each $p \geq 1$ and for every $q > p$,

$$\tilde{\mathbf{y}}^{(t_{n_p}+k)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{y}}^{(t_{n_q}+k)}(0^{n_q}1^{n_p}) \quad \text{for every } k = 0, \ldots, m_p \tag{1}$$

$$\tilde{\mathbf{y}}^{(t_{n_p}+m_p+1)}(0^{n_p}1^{n_p}) \neq \tilde{\mathbf{y}}^{(t_{n_q}+m_p+1)}(0^{n_q}1^{n_p})\,, \tag{2}$$

while pruning the corresponding sequence (n_p) so that the number of elements in (n_p) remains infinite. Observe that by definition, $m_p \leq m_{p+1}$, and condition (1) holds at least for $k = 0$, whereas condition (2) is met before the next input bit is presented to $\mathcal{N}$ after the input $0^{n_p}1^{n_p} \in L$ has been read, due to $0^{n_q}1^{n_p} \notin L$ for $q > p$. Suppose $m_1 < m_2 < \cdots < m_{p-1}$ have been constructed, satisfying (1) and (2). For the next index $p \geq 1$, let $\tilde{m}_p \geq 0$ be the maximal natural number that meets (1) with m_p replaced by $\tilde{m}_p$, which means $\tilde{m}_p \geq m_{p-1}$. On the contrary assume that $\tilde{m}_p = m_{p-1}$. There exists $\tilde{\mathbf{u}}' \in \{0,1\}^{s-1}$ such that the set $Q = \left\{ q \geq p \;\middle|\; \tilde{\mathbf{y}}^{(t_{n_q}+\tilde{m}_p+1)}(0^{n_q}1^{n_p}) = \tilde{\mathbf{u}}' \right\}$ is infinite since there are only 2^{s-1} possible states of binary neurons. We omit all the elements n_q in (n_p) such that $p \leq q \notin Q$, while the pruned sequence (n_p), including the indices from infinite Q, remains infinite, and $p = \min Q$ is the new succeeding index in the pruned (n_p). In addition, the new maximal value of $\tilde{m}_p$ satisfying (1) for this index p, increases by at least 1, and hence, we have $\tilde{m}_p > m_{p-1}$. Moreover, we can assume without loss of generality that there are infinitely many indices q that meet (2) with m_p replaced by $\tilde{m}_p$, since otherwise we could skip them in (n_p), while increasing $\tilde{m}_p$. Thus, the constructed sequence $m_1, \ldots, m_{p-1}$ is extended with $m_p = \tilde{m}_p > m_{p-1}$ and the sequence (n_p) is further pruned by removing those indices $q > p$ for which (2) is not satisfied. This completes the inductive construction which ensures the sequence (m_p) which corresponds to (n_p) and satisfies (1) and (2), is increasing, and hence unbounded. Hereafter, we assume there are infinitely many even numbers in (m_p) while the proof for the opposite case when there are infinitely many odd numbers in (m_p), is analogous. Thus, by pruning the sequence (n_p) we can assume without loss of generality that m_p is even for every $p \geq 1$.

In addition, for each $p \geq 1$, define m'_p to be the maximum number such that $0 \leq m'_p \leq m_p$ and $0 \leq \xi_1^{(t_{n_p}+k)}(0^{n_p}1^{n_p}) \leq 1$ for every $k = 0, \ldots, m'_p$, which holds at least for $k = 0$ because $\xi_1^{(t_{n_p})}(0^{n_p}1^{n_p}) = y_1^{(t_{n_p}+1)}(0^{n_{p+1}}) \in (0\,,1)$. We introduce $b_k = \beta \sum_{i \in V'} w_{1i} y_i^{(t_{n_p}+k-1)}(0^{n_p}1^{n_p}) \in A$ for $k = 1, \ldots, m_p + 1$, which is a consistent definition for every $p \geq 1$, due to (1). We have $y_1^{(t_{n_p}+k)}(0^{n_p}1^{n_p}) = \sum_{i=0}^{s} w_{1i} y_i^{(t_{n_p}+k-1)}(0^{n_p}1^{n_p}) = \beta^{-1}\left(b_k + y_1^{(t_{n_p}+k-1)}(0^{n_p}1^{n_p})\right)$ for $1 \leq k \leq m'_p+1$. Hence, $\xi_1^{(t)}(0^{n_p}1^{n_p}) = \beta^{-(t-t_{n_p}+1)} y_1^{(t_{n_p})}(0^{n_p}) + \sum_{k=1}^{t-t_{n_p}+1} b_{t-t_{n_p}-k+2}\,\beta^{-k}$ for each $p \geq 1$ and $t_{n_p} \leq t \leq t_{n_p} + \min(m'_p+1, m_p)$. One can prove that $m'_p = m_p$ for every $p \geq 1$.

Since the sequence $y_1^{(t_{n_p})}(0^{n_p})$ is increasing, we have for every $p \geq 1$,

$$y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \beta^{-m_p}\, y_1^{(t_{n_p})}(0^{n_p}) + \sum_{k=1}^{m_p} b_{m_p-k+1}\, \beta^{-k}$$

$$< \quad \beta^{-m_p}\, y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) + \sum_{k=1}^{m_p} b_{m_p-k+1}\, \beta^{-k} = y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) \qquad (3)$$

due to $y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \xi_1^{(t_{n_p}+m_p-1)}(0^{n_p}1^{n_p})$ and m_p is even. There exists $\tilde{\mathbf{v}} \in \{0,1\}^{s-1}$ such that $\tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{v}}$ for infinitely many $p \geq 1$, since there are only 2^{s-1} states of binary neurons, and by pruning the sequence (n_p), we can assume without loss of generality that $\tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{v}}$ for every $p \geq 1$. Similarly, assume there exists a binary neuron $j_0 \in \{2,\ldots,s\}$ such that $y_{j_0}^{(t_{n_p}+m_p+1)}(0^{n_p}1^{n_p}) \neq y_{j_0}^{(t_{n_{p+1}}+m_p+1)}(0^{n_{p+1}}1^{n_p})$ for every $p \geq 1$, according to (2), since there are only $s-1$ binary neurons. It follows that $w_{j_0,1} \neq 0$ because $\tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p})$ by (1), and we define $c = -\frac{1}{w_{j_0,1}} \sum_{i \in V'} w_{j_0,i}\, y_i^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p})$, which is a consistent definition due to $\tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{v}}$ for every $p \geq 1$. Assume $w_{j_0,1} > 0$, while the argument for $w_{j_0,1} < 0$ is analogous. We have $y_{j_0}^{(t_{n_p}+m_p+1)}(0^{n_p}1^{n_p}) = 1$ iff $\xi_{j_0}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) \geq 0$ iff $y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) \geq c$, and similarly, $y_{j_0}^{(t_{n_{p+1}}+m_p+1)}(0^{n_{p+1}}1^{n_p}) = 1$ iff $y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) \geq c$, which implies

$$y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) < c \leq y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) \qquad (4)$$

by (3). We obtain $y_1^{(t_{n_p})}(0^{n_p}) < \beta^{m_p}\left(c - \sum_{k=1}^{m_p} b_{m_p-k+1}\, \beta^{-k}\right) \leq y_1^{(t_{n_{p+1}})}(0^{n_{p+1}})$ for every $p \geq 1$, which implies $\lim_{p\to\infty} \sum_{k=1}^{m_p} b_{m_p-k+1}\, \beta^{-k} = c$.

For $p \geq 1$, $c_p = \lim_{q\to\infty} y_1^{(t_{n_q}+m_p)}(0^{n_q}1^{m_p}) = \beta^{-m_p}c_0 + \sum_{k=1}^{m_p} b_{m_p-k+1}\beta^{-k}$ according to (3), which implies $\lim_{p\to\infty} c_p = c$. We introduce the intervals, $I_{p,r} = \left[\beta^{-r}c + \sum_{k=1}^{r} b_{m_p+r-k+1}\beta^{-k}\,,\ \beta^{-r}c_p + \sum_{k=1}^{r} b_{m_p+r-k+1}\beta^{-k}\right)$ for every $p \geq 1$ and $r = 0,\ldots,\ell_p - 1$, where $\ell_p = m_{p+1} - m_p$ is even. Note that for notational simplicity, we assume $\beta > 1$ while for $\beta < -1$ when the interval lower and upper bounds are swapped for odd r, the argument is similar. It follows from (4) that $y_1^{(t_{n_q}+m_p+r)}(0^{n_q}1^{m_p}) \in I_{p,r}$ for every $q > p$. For all sufficiently large $p \geq p_0$, we will prove by induction on $r = 0,\ldots,\ell_p - 1$ that $I_{p+1,r} \subset I_{p,r}$ and

$$\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) = \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})\,. \qquad (5)$$

For the base case $r = 0$, the length of $I_{p,0} = [c, c_p)$ is β^{ℓ_p} times greater than that of the interval $I_{p+1} = \left[\beta^{-\ell_p}c + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k}\,,\ c_{p+1}\right)$ because $c_{p+1} = \beta^{-\ell_p}c_p + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k}$. According to (4), $\beta^{-\ell_p}c + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k} \leq \beta^{-\ell_p} y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k} = y_1^{(t_{n_{p+1}}+m_{p+1})}(0^{n_{p+1}}1^{n_{p+1}})$

$< c$, which means $I_{p+1,0} = [c, c_{p+1}) \subset I_{p+1}$. Hence, $c_{p+1} < c_p$ and $I_{p+1,0} \subset I_{p,0}$. In addition, $\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_{p+1}}) = \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1})}(0^{n_{p+2}}1^{n_{p+2}}) = \tilde{\mathbf{v}}$ by (1).

For the induction step, assume $I_{p+1,k} \subset I_{p,k}$ and $\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+k)}(0^{n_{p+1}}1^{n_{p+1}})$ $= \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+k)}(0^{n_{p+2}}1^{n_{p+2}})$ for $k = 0, \ldots, r-1$. By definition of b_k, we know $b_{m_p+k} = b_{m_{p+1}+k}$ for $k = 1, \ldots, r$. Hence, the intervals $I_{p,r}$ and $I_{p+1,r}$ have the same lower bound by definition, which ensures $I_{p+1,r} \subset I_{p,r}$ due to their upper bounds satisfy $c_{p+1} < c_p$. On the contrary assume $\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) \neq \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})$, which means there is $j_1 \in \{2, \ldots, s\}$ such that $y_{j_1}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) \neq y_{j_1}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})$. It follows that $w_{j_1,1}$ $\neq 0$ because $\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}}) = \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r-1)}(0^{n_{p+2}}1^{n_{p+2}})$. We define $c' = -\frac{1}{w_{j_1,1}} \sum_{i \in V'} w_{j_1,i}\, y_i^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}})$. For example, consider the case when $w_{j_1,1} > 0$ and $y_{j_1}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) = 1$, while the argument for the remaining cases is similar. By the analogy to c, we have $y_1^{(t_{n_{p+2}}+m_{p+1}+r-1)}(0^{n_{p+2}}1^{n_{p+2}}) < c' \leq y_1^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}})$. If $c' \in I_{p+1,r-1} \subset I_{p,r-1}$, then $y_1^{(t_{n_q}+m_{p+1}+r-1)}(0^{n_q}1^{n_{p+2}}) \geq c'$ for sufficiently large $q > p+2$, which implies $y_{j_1}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}}) \neq y_{j_1}^{(t_{n_q}+m_{p+1}+r)}(0^{n_q}1^{n_{p+2}})$, contradicting (1). If $c' \notin I_{p+1,r-1}$, then $c' \notin I_{q,r-1} \subseteq I_{p+1,r-1}$ for every $q > p$, which gives $y_{j_1}^{(t_{n_{q+1}}+m_q+r)}(0^{n_{q+1}}1^{n_{q+1}}) = y_{j_1}^{(t_{n_{q+2}}+m_{q+1}+r)}(0^{n_{q+2}}1^{n_{q+2}})$ for every $q > p$. Thus, for all sufficiently large $p \geq p_0$, we have $\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r)}$ $(0^{n_{p+1}}1^{n_{p+1}})$ $= \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})$ since there are only 2^{s-1} possible values of c', which completes the induction step.

We conclude that for all sufficiently large $p \geq p_0$, $b'_r = b_{m_p+r} = b_{m_{p+1}+r}$ for $r = 1, \ldots, \ell_p = \ell$, according to (5), which implies $c = B \sum_{q=1}^{\infty} \beta^{-\ell(q-1)} = \frac{B}{1-\beta^{-\ell}}$ where $B = \sum_{r=1}^{\ell} b'_{\ell-r+1}\beta^{-r}$. Hence, one can show that the expression $\beta^{m_p}\left(c - \sum_{k=1}^{m_p} b_{m_p-k+1}\,\beta^{-k}\right) = \beta^{m_{p_0}}\left(c - \sum_{k=1}^{m_{p_0}} b_{m_{p_0}-k+1}\beta^{-k}\right) = C$ is constant for every $p \geq p_0$. Thus, $y_1^{(t_{n_p})}(0^{n_p}) < C \leq y_1^{(t_{n_{p+1}})}(0^{n_{p+1}})$ for every $p \geq p_0$, which is a contradiction. This completes the proof of the theorem. □

4 Two Analog Neurons Accept Deterministic Languages

In this section, we show that a deterministic pushdown automaton can be simulated by a 2ANN with two extra analog unit.

Theorem 2. *For any deterministic context-free language $L \subseteq \{0,1\}^*$, there is a neural network 2ANN with two extra analog units having rational weights, $\mathcal{N}$, which accepts $L = \mathcal{L}(\mathcal{N})$.*

Proof. Let $L = \mathcal{L}(\mathcal{M})$ be accepted by a DPDA $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$ where $Q \neq \emptyset$ is a finite set of states, Σ and Γ are finite sets of input and stack symbols, respectively, which are assumed for simplicity to be the binary alphabet $\Sigma = \Gamma = \{0,1\}$. In addition, $q_0 \in Q$ is the start state, $Z_0 \in \Gamma$ is the starting stack symbol,

and $F \subseteq Q$ is the set of accepting states. Moreover, $\delta : (Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \longrightarrow \mathcal{P}(Q \times \Gamma^*)$ is a transition function that given a current state $q \in Q$ of $\mathcal{M}$, a next symbol $x \in \Sigma \cup \{\varepsilon\}$ of an input word which is read from left to right (including the empty string ε, which means no symbol is read), and a symbol $Z \in \Gamma$ on the top of the stack, produces either the empty set $\delta(q, x, Z) = \emptyset$ (i.e. $\mathcal{M}$ halts), or a one-element set $\delta(q, x, Z) = \{(q', \gamma)\}$ with a new state $q' \in Q$ and a string $\gamma \in \Gamma^*$ that replaces Z on the top of the stack where the first symbol of γ becomes the top element. In order to ensure that $\mathcal{M}$ is truly deterministic, it is assumed that for any $q \in Q$, $Z \in \Gamma$, if $\delta(q, \varepsilon, Z) \neq \emptyset$, then $\delta(q, x, Z) = \emptyset$ for every $x \in \Sigma$. An input word $\mathbf{x} \in \Sigma^*$ is accepted by $\mathcal{M}$ if there is a (unique) sequence of transitions of $\mathcal{M}$ defined by δ, from the start state q_0 with the starting symbol Z_0 on the stack, which, while reading $\mathbf{x}$, terminates in an accepting state $q_f \in F$. We assume without loss of generality that if $\delta(q, x, Z) = \{(q', \gamma)\}$, then the length $|\gamma|$ of string γ is at most 2, whereas $\gamma = Z'Z$ for some $Z' \in \Gamma$, if $|\gamma| = 2$.

We will construct a neural network 2ANN with two extra analog units, $\mathcal{N}$, which accepts the same language $L = \mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{M})$ by simulating the deterministic pushdown automaton $\mathcal{M}$. The stack of $\mathcal{M}$ is realized by the two analog neurons $1, 2 \in V$ of $\mathcal{N}$, where the first unit implements the `top` and `push` operations while the `pop` operation is performed by the second analog neuron. The current contents of the stack, $Z_1 \dots Z_p \in \Gamma^p$ are encoded by the state of an analog neuron,

$$y_k^{\text{cur}} = \sum_{i=1}^{p} \frac{2Z_i + 1}{4^i} \in \mathbb{I} \qquad \text{for } k \in \{1, 2\}\,, \tag{6}$$

using a Cantor-like set which allows an efficient neural implementation of the stack operations [12], producing the new state of analog neurons:

$$\texttt{top} = H(2y_1 - 1) \tag{7}$$

$$\texttt{push}(Z): \quad y_1^{\text{new}} = \sigma\left(\frac{1}{4}y_1^{\text{cur}} + \frac{1}{2}Z + \frac{1}{4}\right) \tag{8}$$

$$\texttt{pop}: \quad y_2^{\text{new}} = \sigma(4y_2^{\text{cur}} - 2\texttt{top} - 1)\,. \tag{9}$$

The finite control of $\mathcal{M}$ which is defined by the transition function δ, is implemented by binary neurons. We will describe its functionality while the omitted technical details are ensured using known techniques of implementing finite automata by neural networks with only integer weights [1, 3, 4, 7, 14, 20]. At the beginning of the simulation of $\mathcal{M}$ by $\mathcal{N}$, the stack is initialized by the starting stack symbol Z_0. This is implemented by a special binary neuron init $\in V'$ which is only initially active, that is, $y_{\text{init}}^{(t)} = 1$ iff $t = 0$. Thus, init is connected to the first analog neuron $1 \in V$ via the weight $w(\text{init}, 1) = (2Z_0 + 1)/4$, which encodes the stack contents Z_0 by analog state $y_1^{(1)} = (2Z_0 + 1)/4$, according to (6). Then, each transition of $\mathcal{M}$ is realized by one so-called *macrostep* $\tau \geq 1$ which is composed of 12 computational steps of $\mathcal{N}$, starting at the discrete time instant $t = 12(\tau - 1) + 2$ (including the first two steps $t = 0, 1$ for the stack initialization). Hereafter, the computational time $t = 0, 1, 2, \dots, 12$ of $\mathcal{N}$ is for simplicity related

Table 1. The macrostep of 2ANN $\mathcal{N}$ simulating one transition of DPDA $\mathcal{M}$ (including the `pop` and `push` operations)

t	$y_1^{(t)}$	$y_2^{(t)}$	$y_{\rm ctrl}^{(t)}$	$y_{z_0}^{(t)}$	$y_{z_1}^{(t)}$	$y_{z'_0}^{(t)}$	$y_{z'_1}^{(t)}$	$y_{\tt top}^{(t)}$	$y_{\rm nxt}^{(t)}$	$y_{\rm out}^{(t)}$	$y_{inp}^{(t)}$
0	z	0	1	0	0	0	0		0	0	0
1	0	z	0	0	0	0	0	Z	0	0	0
2	z	0	1	0	0	0	0		0	0	0
3	0	z	0	0	0	0	0		1	$q \in F$	0
4	z	0	1	0	0	0	0		0	0	x
5	0	z	0	0	0	0	0		0	0	0
6	z	0	1	0	0	0	0		0	0	0
7	0	z	1	0	0	0	0		0	0	0
8	0	$2z$	1	$Z=0$	$Z=1$	0	0		0	0	0
9	0	$z'=4z-2Z-1$	0	0	0	0	0		0	0	0
10	z'	0	0	0	0	0	0		0	0	0
11	$\frac{z'}{2}$	0	0	0	0	$Z'=0$	$Z'=1$		0	0	0
$12 \equiv 0$	$z''=\frac{z'}{4}+\frac{Z'}{2}+\frac{1}{4}$	0	1	0	0	0	0		0	0	0

to the macrostep. The state evolution of selected neurons during the macrostep is presented in Table 1.

At the beginning of the macrostep when $t = 0$, the state of the first analog neuron $1 \in V$ encodes the current contents of the stack, that is, $y_1^{(0)} = z \in \mathbb{I}$ by (6). The storage of the stack contents alternates between the two analog neurons which are connected by the weights $w(1,2) = w(2,1) = 1$. These unit weights copy the state from the first analog neuron to the second one and back, under the control of binary neuron $\mathrm{ctrl} \in V'$. During the macrostep, the output of ctrl produces a sequence of binary states given by the regular expression $1(01)^3(110+010)(001+101)$ starting with $y_{\rm ctrl}^{(0)} = 1$, where the strings 110 and 001 deviating from the regular signal $(01)^*$ correspond to the `pop` and `push` operations, respectively, if they occur as described below. For this purpose, the weights $w(\mathrm{ctrl}, 1) = -W$, $w(\mathrm{ctrl}, 2) = W$, and $w(0,2) = -W$ are introduced, where $W > 0$ is a sufficiently large positive parameter excluding the influence from other neurons. It follows that $z = y_1^{(0)} = y_2^{(1)} = y_1^{(2)} = y_2^{(3)} = y_1^{(4)} = \cdots$ and $0 = y_2^{(0)} = y_1^{(1)} = y_2^{(2)} = y_1^{(3)} = y_2^{(4)} = \cdots$, as shown in Table 1.

At time instant $t = 1$ of the macrostep, the binary neuron $\mathrm{top} \in V'$ reads the top element $Z \in \Gamma$ from the stack, that is, $y_{\tt top}^{(1)} = Z \in \{0,1\}$, which is implemented by the weight $w(1, \mathrm{top}) = 2$ and the bias $w(0, \mathrm{top}) = -1$, according to (7). If $\delta(q, x, Z) \neq \emptyset$ for some $x \in \Sigma$, where $q \in Q$ is a current state of $\mathcal{M}$ encoded by binary neurons of $\mathcal{N}$, which is tested at time instant $t = 2$, then $y_{\rm nxt}^{(3)} = 1$, $y_{\rm out}^{(3)} = 1$ iff $q \in F$ is a final state, and $y_{\rm inp}^{(4)} = x \in \{0,1\}$ is the next input symbol, by the input/output protocol. Anyway, the next two steps $t = 5, 6$ of the macrostep are exploited for evaluating the transition function $\delta(q, x, Z)$ where $x = \varepsilon$ is the empty word if $\delta(q, \varepsilon, Z) \neq \emptyset$. If $\delta(q, x, Z) = \emptyset$, then the simulation by $\mathcal{N}$ terminates since the computation of $\mathcal{M}$ halts.

Thus, assume $\delta(q, x, Z) = \{(q', \gamma)\}$ where $q' \in Q$ is the new state of $\mathcal{M}$, which substitutes the old one encoded by binary neurons of $\mathcal{N}$, and $\gamma \in \Gamma^*$ should replace the top symbol on the stack. If $|\gamma| \leq 1$, then the top symbol Z is popped from the stack during time instant $t = 7, 8, 9$ of the macrostep. At time instant $t = 7$, the current contents of the stack are stored by the second analog neuron as $y_2^{(7)} = z$. The `pop` operation is implemented by the weights $w(2, 2) = 2$, $w(z_0, 2) = -1$, and $w(z_1, 2) = -3$ from the binary neurons $z_0, z_1 \in V'$ whose outputs are activated at time instant $t = 8$ of the macrostep so that $y_{z_b}^{(8)} = 1$ iff $Z = b \in \{0, 1\}$. Moreover, we know $y_{\text{ctrl}}^{(7)} = y_{\text{ctrl}}^{(8)} = 1$ and $y_{\text{ctrl}}^{(9)} = 0$ when the `pop` operation applies (otherwise, $y_{\text{ctrl}}^{(7)} = 0$). Hence, $y_2^{(8)} = 2z$ by $w(2, 2) = 2$, and $y_2^{(9)} = 4z - 2\texttt{top} - 1 = z' \in \mathbb{I}$ due to $w(z_0, 2) = -1$ and $w(z_1, 2) = -3$, which pops the top symbol $Z = \texttt{top}$ from the stack according to (9).

If $|\gamma| \geq 1$, then either $\gamma = Z'$ or $\gamma = Z'Z$, where $Z' \in \Gamma$ is the new top symbol which is pushed to the stack during time instant $t = 10, 11, 12$ of the macrostep. At time instant $t = 10$, the current contents of the stack are stored by the first analog neuron as $y_1^{(10)} = z'$. The $\texttt{push}(Z')$ operation is implemented by the weights $w(1, 1) = \frac{1}{2}$, $w(z'_0, 1) = \frac{1}{4}$, and $w(z'_1, 1) = \frac{3}{4}$ from the binary neurons $z'_0, z'_1 \in V'$ whose outputs are activated at time instant $t = 11$ of the macrostep so that $y_{z'_b}^{(11)} = 1$ iff $Z' = b \in \{0, 1\}$. Moreover, we know $y_{\text{ctrl}}^{(10)} = y_{\text{ctrl}}^{(11)} = 0$ and $y_{\text{ctrl}}^{(12)} = 1$ when the `push` operation applies (otherwise, $y_{\text{ctrl}}^{(10)} = 1$). Hence, $y_1^{(11)} = \frac{z'}{2}$ by $w(1, 1) = \frac{1}{2}$, and $y_2^{(12)} = \frac{z'}{4} + \frac{Z'}{2} + \frac{1}{4} = z'' \in \mathbb{I}$ due to $w(z'_0, 1) = \frac{1}{4}$ and $w(z'_1, 1) = \frac{3}{4}$, which pushes the symbol Z' to the stack according to (8). At time instant $t = 12$, the macrostep of $\mathcal{N}$ simulating one transition of $\mathcal{M}$ using rational weights is finished while the new contents z'' of the stack are stored by the first analog neuron as required for the next macrostep. This completes the simulation and the proof of the theorem. □

5 Conclusion

In this paper, we have refined the analysis of the computational power of binary-state neural networks αANNs extended with α analog-state neurons. We have proven that the deterministic (context-free) language $L = \{0^n 1^n \mid n \geq 1\}$ which imitates counting to any number n, cannot be recognized offline by any 1ANN with real weights. It is an open question whether a 1ANN can recognize any non-regular context-free language and whether there is a non-context-sensitive language that can be accepted offline by a 1ANN. In addition, we have shown that any deterministic language can be accepted by a 2ANN with rational weights. It is an open problem whether two extra rational-weight analog units suffice for simulating any Turing machine. Another challenge for further research is to prove a proper "natural" hierarchy of neural networks between integer and rational weights similarly as it is known between rational and real weights [2] and possibly, map it to known hierarchies of regular/context-free languages. This problem is related to a more general issue of finding suitable complexity measures of sub-recursive neural networks establishing the complexity hierarchies, which could be

employed in practical neurocomputing, e.g. the precision of weight parameters, energy complexity [14], temporal coding etc.

References

1. Alon, N., Dewdney, A.K., Ott, T.J.: Efficient simulation of finite automata by neural nets. Journal of the ACM **38**(2), 495–514 (1991)
2. Balcázar, J.L., Gavaldà, R., Siegelmann, H.T.: Computational power of neural networks: A characterization in terms of Kolmogorov complexity. IEEE Transactions on Information Theory **43**(4), 1175–1183 (1997)
3. Horne, B.G., Hush, D.R.: Bounds on the complexity of recurrent neural network implementations of finite state machines. Neural Networks **9**(2), 243–252 (1996)
4. Indyk, P.: Optimal simulation of automata by neural nets. In: Proceedings of the STACS 1995 Twelfth Annual Symposium on Theoretical Aspects of Computer Science. LNCS, vol. 900, pp. 337–348 (1995)
5. Kilian, J., Siegelmann, H.T.: The dynamic universality of sigmoidal neural networks. Information and Computation **128**(1), 48–56 (1996)
6. Koiran, P.: A family of universal recurrent networks. Theoretical Computer Science **168**(2), 473–480 (1996)
7. Minsky, M.: Computations: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs (1967)
8. Orponen, P.: Computing with truly asynchronous threshold logic networks. Theoretical Computer Science **174**(1-2), 123–136 (1997)
9. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks **61**, 85–117 (2015)
10. Siegelmann, H.T.: Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhäuser, Boston (1999)
11. Siegelmann, H.T., Sontag, E.D.: Analog computation via neural networks. Theoretical Computer Science **131**(2), 331–360 (1994)
12. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. Journal of Computer System Science **50**(1), 132–150 (1995)
13. Šíma, J.: Analog stable simulation of discrete neural networks. Neural Network World **7**(6), 679–686 (1997)
14. Šíma, J.: Energy complexity of recurrent neural networks. Neural Computation **26**(5), 953–973 (2014)
15. Šíma, J.: The power of extra analog neuron. In: Proceedings of the TPNC 2014 Third International Conference on Theory and Practice of Natural Computing. LNCS, vol. 8890, pp. 243–254 (2014)
16. Šíma, J.: Three analog units are Turing universal. In: Proceedings of the TPNC 2018 Seventh International Conference on Theory and Practice of Natural Computing. LNCS, vol. 11324, pp. 460–472 (2018)
17. Šíma, J.: Subrecursive neural networks. Neural Networks **116**, 208–223 (2019)
18. Šíma, J., Orponen, P.: General-purpose computation with neural networks: A survey of complexity theoretic results. Neural Computation **15**(12), 2727–2778 (2003)
19. Šíma, J., Savický, P.: Quasi-periodic β-expansions and cut languages. Theoretical Computer Science **720**, 1–23 (2018)
20. Šíma, J., Wiedermann, J.: Theory of neuromata. Journal of the ACM **45**(1), 155–178 (1998)
21. Šorel, M., Šíma, J.: Robust RBF finite automata. Neurocomputing **62**, 93–110 (2004)