A Low-Energy Implementation of Finite Automata by Optimal-Size Neural Nets

Jiří Šíma*

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

Abstract. Recently, a new so-called energy complexity measure has been introduced and studied for feedforward perceptron networks. This measure is inspired by the fact that biological neurons require more energy to transmit a spike than not to fire and the activity of neurons in the brain is quite sparse, with only about 1% of neurons firing. We investigate the energy complexity for recurrent networks which bounds the number of active neurons at any time instant of a computation. We prove that any deterministic finite automaton with m states can be simulated by a neural network of optimal size $s = \Theta(\sqrt{m})$ with time overhead $O(s/e)$ per one input bit, using the energy $O(e)$, for any $e = \Omega(\log s)$ and $e = O(s)$, which shows the time-energy tradeoff in recurrent networks.

1 Introduction

In biological neural networks the energy cost of a firing neuron is relatively high while energy supplied to the brain is limited and hence the activity of neurons in the brain is quite sparse, with only about 1% of neurons firing [4]. This is in contrast to artificial neural networks in which on average every second unit fires during a computation. This fact has recently motivated the definition of a new complexity measure for *feedforward* perceptron networks (threshold circuits), the so-called energy complexity [11] which is the maximum number of units in the network which output 1, taken over all the inputs to the circuit. The energy has been shown to be closely related by tradeoff results to other complexity measures such as the network size (i.e., the number of neurons) [13, 15], the circuit depth (i.e., parallel computational time) [12, 13], and the fan-in (i.e., the maximum number of inputs to a single unit) [10] etc. In addition, energy complexity has found its use in circuit complexity, e.g. as a tool for proving the lower bounds [14] etc.

In this paper, we investigate for the first time energy complexity for *recurrent* neural networks which we define to be the maximum number of neurons outputting 1 at any time instant, taken over all possible computations. It has been known for a long time that the computational power of binary-state recurrent networks corresponds to that of finite automata since the network of size s units can reach only a finite number (at most 2^s) different states [8]. A simple way of simulating a given deterministic finite automaton A with m states by a neural

* Research was supported by the projects GA ČR P202/10/1333 and RVO: 67985807.

network N of size $O(m)$ is to implement each of the $2m$ transitions of A (having 0 and 1 transitions for each state) by a single unit in N which checks whether the input bit agrees the respective type of transition [6]. Clearly, this simple linear-size implementation of finite automata requires only a constant energy.

Much effort was given to reducing the size of neural automata (e.g. [1–3, 9]), and indeed, neural networks of size $\Theta(\sqrt{m})$ implementing a given deterministic finite automaton with m states were proposed and proven to be size-optimal [2, 3]. A natural question arises: what is the energy consumption when simulating finite automata by optimal-size neural networks? We answer this question by proving the tradeoff between the energy and the time overhead of the simulation. In particular, we prove that an optimal-size neural network of $s = \Theta(\sqrt{m})$ units can be constructed to simulate a deterministic finite automaton with m states using the energy $O(e)$ for any $e = \Omega(\log s)$ and $e = O(s)$, while the time overhead for processing one input bit is $O(s/e)$. For this purpose, we adapt the asymptotically optimal method of threshold circuit synthesis due to Lupanov [5].

This paper is organized as follows. In Section 2, the main result is formulated after a brief review of the basic definitions. The subsequent two sections are devoted to the technical proof: Section 3 deals with a decomposition of the transition function and Section 4 describes the construction of low-energy neural automata. Section 5 concludes with some remarks on lower bounds on the energy complexity of neural network automata.

2 Neural Networks as Finite Automata

We will first specify the model of a recurrent *neural network* N. The network consists of s *units (neurons)*, indexed as $V = \{1, \ldots, s\}$, where s is called the network *size*. The units are connected into an oriented graph representing the *architecture* of N, in which each edge (i, j) leading from unit i to j is labeled with an integer *weight* $w(i, j)$. The absence of a connection within the architecture corresponds to a zero weight between the respective neurons, and vice versa.

The *computational dynamics* of N determines for each unit $j \in V$ its binary *state (output)* $y_j^{(t)} \in \{0, 1\}$ at discrete time instants $t = 0, 1, 2, \ldots$. We say that neuron j is *active (fires)* at time t if $y_j^{(t)} = 1$, while j is *passive* for $y_j^{(t)} = 0$. This establishes the *network state* $\mathbf{y}^{(t)} = (y_1^{(t)}, \ldots, y_s^{(t)}) \in \{0, 1\}^s$ at each discrete time instant $t \geq 0$. At the beginning of a computation, N is placed in an *initial state* $\mathbf{y}^{(0)}$. At discrete time instant $t \geq 0$, an excitation of any neuron $j \in V$ is defined as $\xi_j^{(t)} = \sum_{i=1}^{s} w(i, j) y_i^{(t)} - h(j)$ including an integer *threshold* $h(j)$ local to unit j. At the next instant $t + 1$, the neurons $j \in \alpha_{t+1}$ from a selected subset $\alpha_{t+1} \subseteq V$ update their states $y_j^{(t+1)} = H(\xi_j^{(t)})$ in parallel by applying the *Heaviside function* $H(\xi)$ which is defined to be 1 for $\xi \geq 0$ and 0 for $\xi < 0$. The remaining units $j \in V \setminus \alpha_{t+1}$ do not change their outputs, that is $y_j^{(t+1)} = y_j^{(t)}$ for $j \notin \alpha_{t+1}$. In this way, the new network state $\mathbf{y}^{(t+1)}$ at time $t + 1$ is determined. We define the *energy complexity* of N to be the maximum number of active units $\sum_{j=1}^{s} y_j^{(t)}$ at any time instant $t \geq 0$, taken over all computations of N.

The computational power of recurrent neural networks has been studied analogously to the traditional models of computations so that the networks are exploited as acceptors of formal languages $L \subseteq \{0,1\}^*$ over the binary alphabet. For the finite networks that are to recognize regular languages, the following input/output protocol has been used [1–3, 7–9]. A binary input word (string) $\mathbf{x} = x_1 \dots x_n \in \{0,1\}^n$ of arbitrary length $n \geq 0$ is sequentially presented to the network bit by bit via an *input neuron* $\mathrm{in} \in V$. The state of this unit is externally set (and clamped) to the respective input bits at prescribed time instants, regardless of any influence from the remaining neurons in the network, that is, $y_{\mathrm{in}}^{(\tau(i-1))} = x_i$ for $i = 1, \dots, n$ where an integer parameter $\tau \geq 1$ is the *period* or *time overhead* for processing a single input bit. Then, an *output neuron* $\mathrm{out} \in V$ signals at time τn whether the input word belongs to underlying language L, that is, $y_{\mathrm{out}}^{(\tau n)} = 1$ for $\mathbf{x} \in L$, whereas $y_{\mathrm{out}}^{(\tau n)} = 0$ for $\mathbf{x} \notin L$.

Now, we can formulate our main result concerning a low-energy implementation of finite automata by optimal-size neural nets:

Theorem 1. *A given deterministic finite automaton A with m states can be simulated by a neural network N of optimal size $s = \Theta(\sqrt{m})$ neurons with time overhead $O(s/e)$ per one input bit, using the energy $O(e)$, where e is any function satisfying $e = \Omega(\log s)$ and $e = O(s)$.*

Proof. A set Q of m states of a given deterministic finite automaton A can be arbitrarily enumerated so that each $q \in Q$ is binary encoded using $p = \lceil \log m \rceil + 1$ bits including one additional bit which indicates the final states. Then, the respective transition function $\delta : Q \times \{0,1\} \longrightarrow Q$ of A, producing its new state $q_{\mathrm{new}} = \delta(q_{\mathrm{old}}, x) \in Q$ from the old state $q_{\mathrm{old}} \in Q$ and input bit $x \in \{0,1\}$, can be viewed as a vector Boolean function $\mathbf{f} : \{0,1\}^{p+1} \longrightarrow \{0,1\}^p$ in terms of binary encoding of states. In the following two sections we will adapt the asymptotically optimal method of threshold circuit synthesis due to Lupanov [5] to implement $\mathbf{f}$ by a low-energy recurrent neural network.

3 The Transition Function Decomposition

The $p+1$ arguments of vector function $\mathbf{f}(\mathbf{u}, \mathbf{v}, \mathbf{z})$ are split into three groups $\mathbf{u} = (u_1, \dots, u_{p_1})$, $\mathbf{v} = (v_1, \dots, v_{p_2})$, and $\mathbf{z} = (z_1, \dots, z_{p_3})$, respectively, where $p_1 = \lfloor (p + 1 - \log p - \log(p + 1 - \log p))/2 \rfloor$, $p_3 = \lfloor \log(p + 1 - \log p) - 2 \rfloor$, and $p_2 = p + 1 - p_3 - p_1$. Then, each function element $f_k : \{0,1\}^{p+1} \longrightarrow \{0,1\}$ $(1 \leq k \leq p)$ of vector function $\mathbf{f} = (f_1, \dots, f_p)$ is decomposed to

$$f_k(\mathbf{u}, \mathbf{v}, \mathbf{z}) = \bigvee_{\mathbf{c} \in \{0,1\}^{p_3}} \left(f_k(\mathbf{u}, \mathbf{v}, \mathbf{c}) \wedge \bigwedge_{j=1}^{p_3} \ell_{c_j}(z_j) \right) , \tag{1}$$

where the respective literals are defined as $\ell_c(z) = z$ for $c = 1$ and $\ell_c(z) = \neg z$ for $c = 0$. Furthermore, we define vector functions $\mathbf{g}_k : \{0,1\}^{p_1+p_2} \longrightarrow \{0,1\}^{p_1}$ for $k = 1, \dots, p$ as

$$\mathbf{g}_k(\mathbf{u}, \mathbf{v}) = (f_k(\mathbf{u}, \mathbf{v}, [0]^{p_3}), f_k(\mathbf{u}, \mathbf{v}, [1]^{p_3}), \dots, f_k(\mathbf{u}, \mathbf{v}, [2^{p_3} - 1]^{p_3}), 0, \dots, 0) \tag{2}$$

where $[j]^n = \mathbf{c} = (c_1, \ldots, c_n) \in \{0,1\}^n$ denotes an n-bit binary representation of integer $j \geq 0$, that is, $j = \langle \mathbf{c} \rangle = \sum_{i=1}^{n} 2^{i-1} c_i$. The vector produced by $\mathbf{g}_k$ in (2) has p_1 elements out of which the first 2^{p_3} items are defined using f_k for all possible values of argument $\mathbf{z} \in \{0,1\}^{p_3}$, while the remaining ones are 0s, which is a correct definition since $2^{p_3} < p_1$ for sufficiently large p.

Denote $r = p_1 - 1$. For each $\mathbf{g}_k$ $(1 \leq k \leq p)$, we will construct four vector functions $\mathbf{g}_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0,1\}^{p_1}$ and $\mathbf{h}_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0,1\}^{p_1}$ for $a \in \{0,1\}$ such that

$$\mathbf{g}_k(a, \mathbf{u}', \mathbf{v}) = \mathbf{g}_k^a(\mathbf{u}', \mathbf{v}) \oplus \mathbf{h}_k^a(\mathbf{u}', \mathbf{v}) \tag{3}$$

for any $a \in \{0,1\}$, $\mathbf{u}' \in \{0,1\}^r$, and $\mathbf{v} \in \{0,1\}^{p_2}$, where $\oplus$ denotes a bitwise parity (i.e., $\mathbf{z} = \mathbf{x} \oplus \mathbf{y} \in \{0,1\}^n$ is defined for vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, $\mathbf{y} = (y_1, \ldots, y_n) \in \{0,1\}^n$, and $\mathbf{z} = (z_1, \ldots, z_n) \in \{0,1\}^n$ as $z_i = 1$ iff $x_i \neq y_i$ for every $i = 1, \ldots, n$) which is an associative operation. In addition, the construction will guarantee that for any $a \in \{0,1\}$, $\mathbf{v} \in \{0,1\}^{p_2}$, and $\mathbf{u}'_1, \mathbf{u}'_2 \in \{0,1\}^r$,

$$\text{if } \mathbf{u}'_1 \neq \mathbf{u}'_2\,, \text{ then } \mathbf{g}_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \mathbf{g}_k^a(\mathbf{u}'_2, \mathbf{v}) \text{ and } \mathbf{h}_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \mathbf{h}_k^a(\mathbf{u}'_2, \mathbf{v})\,. \tag{4}$$

For any $\mathbf{v} \in \{0,1\}^{p_2}$, the function values of $\mathbf{g}_k^a$ are defined inductively as $\mathbf{g}_k^a([i]^r, \mathbf{v}) \in \{0,1\}^{p_1} \setminus G_k^a(i, \mathbf{v})$ is chosen arbitrarily for $i = 0, \ldots, 2^r - 1$ where

$$\begin{aligned} G_k^a(i, \mathbf{v}) = \{&\mathbf{g}_k^a([j]^r, \mathbf{v})\,, \\ &\mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k(a, [j]^r, \mathbf{v}) \oplus \mathbf{g}_k^a([j]^r, \mathbf{v}) \mid j = 0, \ldots, i-1\}\,, \end{aligned} \tag{5}$$

and functions $\mathbf{h}_k^a$ are defined so that equation (3) is met:

$$\mathbf{h}_k^a(\mathbf{u}', \mathbf{v}) = \mathbf{g}_k(a, \mathbf{u}', \mathbf{v}) \oplus \mathbf{g}_k^a(\mathbf{u}', \mathbf{v})\,. \tag{6}$$

Note that $\emptyset = G_k^a(0, \mathbf{v}) \subseteq G_k^a(1, \mathbf{v}) \subseteq \cdots \subseteq G_k^a(2^r - 1, \mathbf{v})$ and $|G_k^a(i, \mathbf{v})| \leq 2i$ according to (5), which implies $|G_k^a(i, \mathbf{v})| \leq |G_k^a(2^r - 1, \mathbf{v})| \leq 2(2^r - 1)$. Hence, $|\{0,1\}^{p_1} \setminus G_k^a(i, \mathbf{v})| \geq 2^{p_1} - 2(2^r - 1) = 2$, which ensures that $\mathbf{g}_k^a(\mathbf{u}', \mathbf{v})$ is correctly defined for all arguments $\mathbf{u}' \in \{0,1\}^r$. Moreover, condition (4) is satisfied because for any $i, j \in \{0, \ldots, 2^r - 1\}$ such that $i > j$, definition (5) secures $\mathbf{g}_k^a([i]^r, \mathbf{v}) \neq \mathbf{g}_k^a([j]^r, \mathbf{v})$ and $\mathbf{h}_k^a([i]^r, \mathbf{v}) = \mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k^a([i]^r, \mathbf{v}) \neq \mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k(a, [i]^r, \mathbf{v}) \oplus \mathbf{g}_k(a, [j]^r, \mathbf{v}) \oplus \mathbf{g}_k^a([j]^r, \mathbf{v}) = \mathbf{h}_k^a([j]^r, \mathbf{v})$ by using (6) and the fact that $\mathbf{x} \oplus \mathbf{x} \oplus \mathbf{y} = \mathbf{y}$.

We further decompose $\mathbf{g}_k^a$ and $\mathbf{h}_k^a$ by using the functions $\varphi_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0, \ldots, 2^{p_1} - 1\}$ and $\psi_k^a : \{0,1\}^{r+p_2} \longrightarrow \{0, \ldots, 2^{p_1} - 1\}$ as

$$\mathbf{g}_k^a(\mathbf{u}', \mathbf{v}) = [\varphi_k^a(\mathbf{u}', \mathbf{v})]^{p_1} \text{ and } \mathbf{h}_k^a(\mathbf{u}', \mathbf{v}) = [\psi_k^a(\mathbf{u}', \mathbf{v})]^{p_1}\,, \tag{7}$$

respectively, which satisfy for any $a \in \{0,1\}$, $\mathbf{v} \in \{0,1\}^{p_2}$, and $\mathbf{u}'_1, \mathbf{u}'_2 \in \{0,1\}^r$,

$$\text{if } \mathbf{u}'_1 \neq \mathbf{u}'_2\,, \text{ then } \varphi_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \varphi_k^a(\mathbf{u}'_2, \mathbf{v}) \text{ and } \psi_k^a(\mathbf{u}'_1, \mathbf{v}) \neq \psi_k^a(\mathbf{u}'_2, \mathbf{v}) \tag{8}$$

according to (4). Now, we can plug (2), (3), and (7) into (1) which results in

$$f_k(a, \mathbf{u}', \mathbf{v}, \mathbf{z}) = \bigvee_{\mathbf{c} \in \{0,1\}^{p_3}} \left((\mathbf{g}_k(a, \mathbf{u}', \mathbf{v}))_{\langle \mathbf{c} \rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i) \right)$$

$$= \bigvee_{\mathbf{c}\in\{0,1\}^{p_3}} \left(\left(\left([\varphi_k^a(\mathbf{u}',\mathbf{v})]^{p_1} \right)_{\langle \mathbf{c} \rangle} \wedge \neg \left([\psi_k^a(\mathbf{u}',\mathbf{v})]^{p_1} \right)_{\langle \mathbf{c} \rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i) \right) \right.$$
$$\left. \vee \left(\neg \left([\varphi_k^a(\mathbf{u}',\mathbf{v})]^{p_1} \right)_{\langle \mathbf{c} \rangle} \wedge \left([\psi_k^a(\mathbf{u}',\mathbf{v})]^{p_1} \right)_{\langle \mathbf{c} \rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i) \right) \right) , \quad (9)$$

where $(\mathbf{x})_i$ denotes the ith element of vector $\mathbf{x}$.

4 The Finite Automaton Implementation

In this section, we will describe the construction of low-energy recurrent neural network N simulating a given finite automaton A. In particular, set of neurons V is composed of four disjoint layers $V = \nu_0 \cup \nu_1 \cup \nu_2 \cup \nu_3$. A current state of A and an input bit are stored using $p+1$ neurons which constitute layer ν_0. Thus, set ν_0 includes the input neuron in $\in \nu_0$ and the output neuron out $\in \nu_0$ which saves the bit (in the state encoding) that indicates the final states. We will implement formula (9) in N for evaluating the transition function $\mathbf{f}$ in terms of binary encoding of states in order to compute the new state of A. Layer $\nu_0 = \{\text{in}\} \cup \nu_{01} \cup \nu_{02} \cup \nu_{03}$ is disjointly split into four parts corresponding to the partition of arguments of $\mathbf{f}(a, \mathbf{u}', \mathbf{v}, \mathbf{z})$, respectively, that is, $\nu_{01} = \{u_1, \ldots, u_r\}$, $\nu_{02} = \{v_1, \ldots, v_{p_2}\}$, and $\nu_{03} = \{z_1, \ldots, z_{p_3}\}$.

The next layer $\nu_1 = \nu_{11} \cup \nu_{12}$ consists of 2^{p_2} neurons in $\nu_{11} = \{\mu_{\langle \mathbf{b} \rangle} \mid \mathbf{b} \in \{0,1\}^{p_2}\}$ for computing all possible monomials $\bigwedge_{i=1}^{p_2} \ell_{b_i}(v_i)$ over input variables $\mathbf{v}$, and two control units in $\nu_{12} = \{\kappa_0^0, \kappa_0^1\}$ which indicate the input bit value. This is implemented by weights $w(v_i, \mu_{\langle \mathbf{b} \rangle}) = 2b_i - 1$ for $i = 1, \ldots, p_2$, and threshold $h(\mu_{\langle \mathbf{b} \rangle}) = \sum_{i=1}^{p_2} b_i$, for any $\mathbf{b} = (b_1, \ldots, b_{p_2}) \in \{0,1\}^{p_2}$ so that $\mu_{\langle \mathbf{b} \rangle}$ fires iff $\mathbf{b} = \mathbf{v}$. In addition, we define $w(\text{in}, \kappa_0^1) = -w(\text{in}, \kappa_0^0) = 1$ and $h(\kappa_0^1) = 1$, $h(\kappa_0^0) = 0$, which ensures that $y_{in} = 1$ iff κ_0^1 fires iff κ_0^0 is passive.

Furthermore, layer $\nu_2 = \nu_{21} \cup \nu_{22}$ where $\nu_{21} = \{\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a}, \gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a} \mid 1 \leq k \leq p\,,\, a \in \{0,1\}\,, j = 0, \ldots, 2^{p_1} - 1\}$, and $\nu_{22} = \{\kappa_i^a \mid a \in \{0,1\}\,,\, i = 1, \ldots, d+1\}$ with $d = \lceil 2p2^{p_1}/e \rceil$, serves for a low-energy computation of functions $\varphi_k^a(\mathbf{u}', \mathbf{v})$ and $\psi_k^a(\mathbf{u}', \mathbf{v})$. We will first show how to implement functions $\varphi_k^a(\mathbf{u}', \mathbf{v})$ for any $1 \leq k \leq p$ and $a \in \{0,1\}$ with no constraints on energy by using the outputs of neurons from ν_{01} and ν_{11}. In particular, 2^{p_1} pairs of neurons $\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a} \in \nu_{21}$ for $j = 0, \ldots, 2^{p_1} - 1$ are employed having zero thresholds for now (their thresholds will be defined below for the low-energy implementation) and weights $w(u_i, \gamma_{kj}^{\varphi a}) = -w(u_i, \lambda_{kj}^{\varphi a}) = 2^{i-1}$ for $i = 1, \ldots, r$ and $w(\mu_{\langle \mathbf{b} \rangle}, \lambda_{kj}^{\varphi a}) = -w(\mu_{\langle \mathbf{b} \rangle}, \gamma_{kj}^{\varphi a}) = d_{kj}^{\varphi a \mathbf{b}} \in \{0, \ldots, 2^r - 1\}$ such that $j = \varphi_k^a([d_{kj}^{\varphi a \mathbf{b}}]^r, \mathbf{b}) \in \{0, \ldots, 2^{p_1} - 1\}$ for $\mathbf{b} \in \{0,1\}^{p_2}$. Note that $d_{kj}^{\varphi a \mathbf{b}}$ is uniquely defined according to (8). It follows that for given $\mathbf{u}' \in \{0,1\}^r$ and $\mathbf{v} \in \{0,1\}^{p_2}$, neuron $\gamma_{kj}^{\varphi a}$ fires iff $\sum_{i=1}^{r} w(u_i, \gamma_{kj}^{\varphi a}) y_{u_i} + \sum_{\mathbf{b} \in \{0,1\}^{p_2}} w(\mu_{\langle \mathbf{b} \rangle}, \gamma_{kj}^{\varphi a}) y_{\mu_{\langle \mathbf{b} \rangle}} \geq 0$ iff $\sum_{i=1}^{r} 2^{i-1} u_i' - d_{kj}^{\varphi a \mathbf{v}} \geq 0$ iff $\langle \mathbf{u}' \rangle \geq d_{kj}^{\varphi a \mathbf{v}}$, since $y_{\mu_{\langle \mathbf{b} \rangle}} = 1$ iff $\mathbf{b} = \mathbf{v}$. Similarly, neuron $\lambda_{kj}^{\varphi a}$ is active iff $\langle \mathbf{u}' \rangle \leq d_{kj}^{\varphi a \mathbf{v}}$. Hence, both neurons $\gamma_{kj}^{\varphi a}$ and $\lambda_{kj}^{\varphi a}$ fire at the same time iff $\langle \mathbf{u}' \rangle = d_{kj}^{\varphi a \mathbf{v}}$ iff $j = \varphi_k^a(\mathbf{u}', \mathbf{v})$, which implements function $\varphi_k^a(\mathbf{u}', \mathbf{v})$. Functions $\psi_k^a(\mathbf{u}', \mathbf{v})$ for any $1 \leq k \leq p$ and

$a \in \{0,1\}$ are implemented analogously (replace φ by ψ above) using 2^{p_1} pairs of neurons $\gamma^{\psi a}_{kj}, \lambda^{\psi a}_{kj} \in \nu_{21}$ for $j = 0, \ldots, 2^{p_1}-1$, that is, both units $\gamma^{\psi a}_{kj}$ and $\lambda^{\psi a}_{kj}$ are active iff $j = \psi^a_k(\mathbf{u}', \mathbf{v})$.

We employ control units $\kappa^a_i \in \nu_{12} \cup \nu_{22}$ for $a \in \{0,1\}$ and $i = 0, \ldots, d+1$, for synchronizing the computation of functions $\varphi^a_k(\mathbf{u}', \mathbf{v})$, $\psi^a_k(\mathbf{u}', \mathbf{v})$ by neurons from ν_{21} so that their energy consumption is bounded by $e+2$. For this purpose, we split set $\nu_{21} = \nu^0_{21} \cup \nu^1_{21}$ into two parts $\nu^a_{21} = \{\gamma^{\varphi a}_{kj}, \lambda^{\varphi a}_{kj}, \gamma^{\psi a}_{kj}, \lambda^{\psi a}_{kj} \,|\, 1 \leq k \leq p\,,\ j = 0, \ldots, 2^{p_1}-1\}$ of size $4p2^{p_1}$ according to $a \in \{0,1\}$, and each such part is further partitioned into d blocks of size at most $2e$, that is $\nu^a_{21} = \bigcup_{i=1}^{d} \beta^a_i$ where $|\beta^a_i| \leq 2e$. In addition, we require for every $i = 1, \ldots, d$, if $\gamma^{\varphi a}_{kj} \in \beta^a_i$, then $\lambda^{\varphi a}_{kj} \in \beta^a_i$, and if $\gamma^{\psi a}_{kj} \in \beta^a_i$, then $\lambda^{\psi a}_{kj} \in \beta^a_i$. For any $1 \leq i \leq d$ and $a \in \{0,1\}$, the neurons in block β^a_i are activated by control unit κ^a_{i-1} using the weights $w(\kappa^a_{i-1}, j) = W$ for all $j \in \beta^a_i$, while all neurons $j \in \nu_{21}$ are blocked by thresholds $h(j) = W$ where $W = 2^r$ if there is no support from a corresponding control unit. For current input bit $y_{\text{in}} = a \in \{0,1\}$, the control units $\kappa^a_0, \ldots, \kappa^a_{d+1}$ fire successively one by one, which is achieved by weights $w(\kappa^a_i, \kappa^a_{i+1}) = 1$ for $i = 0, \ldots, d$, $w(\kappa^a_i, \kappa^a_0) = -1$ for $i = 0, \ldots, d+1$, and thresholds $h(\kappa^a_i) = 1$ for $i = 1, \ldots, d+1$. This ensures that only the neurons from one block β^a_i of size at most $2e$ can fire at the same time. In fact, we know that just one unit of each pair $\gamma^{\varphi a}_{kj}, \lambda^{\varphi a}_{kj} \in \beta^a_i$ or $\gamma^{\psi a}_{kj}, \lambda^{\psi a}_{kj} \in \beta^a_i$ is active except for the special pairs of both firing units $\gamma^{\varphi a}_{kj_\varphi}, \lambda^{\varphi a}_{kj_\varphi}$ and $\gamma^{\psi a}_{kj_\psi}, \lambda^{\psi a}_{kj_\psi}$ such that $\varphi^a_k(\mathbf{u}', \mathbf{v}) = j_\varphi$ and $\psi^a_k(\mathbf{u}', \mathbf{v}) = j_\psi$, respectively. Hence, the energy consumption of ν_{21} is bounded by $e+2$. Finally, we must also guarantee that the resulting function values $\varphi^a_k(\mathbf{u}', \mathbf{v}) = j_\varphi$, $\psi^a_k(\mathbf{u}', \mathbf{v}) = j_\psi$ are stored, that is, neurons $\gamma^{\varphi a}_{kj_\varphi}, \lambda^{\varphi a}_{kj_\varphi}, \gamma^{\psi a}_{kj_\psi}, \lambda^{\psi a}_{kj_\psi}$ remain active without any support from corresponding control units until all blocks perform computation which is indicated by control unit κ^a_{d+1}. Neuron κ^a_{d+1} then resets all neurons in ν_{21} before becoming itself passive. This is implemented by symmetric weights $w(\gamma^{\varphi a}_{kj}, \lambda^{\varphi a}_{kj}) = w(\lambda^{\varphi a}_{kj}, \gamma^{\varphi a}_{kj}) = w(\gamma^{\psi a}_{kj}, \lambda^{\psi a}_{kj}) = w(\lambda^{\psi a}_{kj}, \gamma^{\psi a}_{kj}) = W$ for $a \in \{0,1\}$, $k = 1, \ldots, p$, $j = 0, \ldots, 2^{p_1}-1$, and $w(\kappa^a_{d+1}, j) = -W$ for all $j \in \nu_{21}$.

Finally, layer $\nu_3 = \{\pi_{k\langle \mathbf{c}\rangle}, \varrho_{k\langle \mathbf{c}\rangle} \,|\, 1 \leq k \leq p\,,\ \mathbf{c} \in \{0,1\}^{p_3}\}$ is composed of 2^{p_3} pairs of neurons $\pi_{k\langle \mathbf{c}\rangle}, \varrho_{k\langle \mathbf{c}\rangle}$ for each $k = 1, \ldots, p$ which compute $([\varphi^a_k(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c}\rangle} \wedge \neg([\psi^a_k(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c}\rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i)$ and $\neg([\varphi^a_k(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c}\rangle} \wedge ([\psi^a_k(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c}\rangle} \wedge \bigwedge_{i=1}^{p_3} \ell_{c_i}(z_i)$ from (9), respectively, for current input $y_{\text{in}} = a \in \{0,1\}$ by using the states of neurons from ν_{03} and the outputs of units $\gamma^{\varphi a}_{kj}, \lambda^{\varphi a}_{kj}, \gamma^{\psi a}_{kj}, \lambda^{\psi a}_{kj} \in \nu_{21}$ for $j = 0, \ldots, 2^{p_1}-1$ after κ^a_{d+1} fires. For $\mathbf{c} \in \{0,1\}^{p_3}$, we define weights $w(\gamma^{\varphi a}_{kj}, \pi_{k\langle \mathbf{c}\rangle}) = w(\lambda^{\varphi a}_{kj}, \pi_{k\langle \mathbf{c}\rangle}) = -w(\gamma^{\psi a}_{kj}, \pi_{k\langle \mathbf{c}\rangle}) = -w(\lambda^{\psi a}_{kj}, \pi_{k\langle \mathbf{c}\rangle}) = -w(\gamma^{\varphi a}_{kj}, \varrho_{k\langle \mathbf{c}\rangle}) = -w(\lambda^{\varphi a}_{kj}, \varrho_{k\langle \mathbf{c}\rangle}) = w(\gamma^{\psi a}_{kj}, \varrho_{k\langle \mathbf{c}\rangle}) = w(\lambda^{\psi a}_{kj}, \varrho_{k\langle \mathbf{c}\rangle}) = ([j]^{p_1})_{\langle \mathbf{c}\rangle}$ for $a \in \{0,1\}$, $j = 0, \ldots, 2^{p_1}-1$, and $w(z_i, \pi_{k\langle \mathbf{c}\rangle}) = w(z_i, \varrho_{k\langle \mathbf{c}\rangle}) = 2c_i - 1$ for $i = 1, \ldots, p_3$, and threshold $h(\pi_{k\langle \mathbf{c}\rangle}) = h(\varrho_{k\langle \mathbf{c}\rangle}) = 1 + \sum_{i=1}^{p_3} c_i$. Hence, neuron $\pi_{k\langle \mathbf{c}\rangle}$ is active iff $([\varphi^a_k(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c}\rangle} = 1$ and $([\psi^a_k(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c}\rangle} = 0$ for $y_{\text{in}} = a$, and $y_{z_i} = c_i$ for $i = 1, \ldots, p_3$, since only one pair of neurons $\gamma^{\varphi a}_{kj_\varphi}, \lambda^{\varphi a}_{kj_\varphi}$ for $0 \leq j_\varphi \leq 2^{p_1}-1$ fires such that $j_\varphi = \varphi^a_k(\mathbf{u}', \mathbf{v})$ and only one pair of units $\gamma^{\psi a}_{kj_\psi}, \lambda^{\psi a}_{kj_\psi}$ for $0 \leq j_\psi \leq 2^{p_1}-1$ is active such that $j_\psi = \psi^a_k(\mathbf{u}', \mathbf{v})$, while the remaining units in v_{21} are passive after κ^a_{d+1} fires. Analogously, neuron $\varrho_{k\langle \mathbf{c}\rangle}$

fires iff $([\varphi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle} = 0$ and $([\psi_k^a(\mathbf{u}', \mathbf{v})]^{p_1})_{\langle \mathbf{c} \rangle} = 1$ for $y_{\text{in}} = a$, and $y_{z_i} = c_i$ for $i = 1, \ldots, p_3$.

It follows that for any $1 \leq k \leq p$, at most one unit among $\pi_{k\langle \mathbf{c} \rangle}, \varrho_{k\langle \mathbf{c} \rangle} \in \nu_3$ for $\mathbf{c} \in \{0,1\}^{p_3}$ is active, which determines the value of $f_k(a, \mathbf{u}', \mathbf{v}, \mathbf{z})$ for $y_{\text{in}} = a$ according (9). Thus, a binary encoding $\mathbf{f}(a, \mathbf{u}', \mathbf{v}, \mathbf{z})$ of the new state of automaton A is computed as disjunctions (9) for $k = 1, \ldots, p$ by units from $\nu_0 \setminus \{\text{in}\}$ (which rewrite the old state of A) using the recurrent connections leading from neurons of ν_3. After re-indexing the units in layer $\nu_0 \setminus \{\text{in}\} = \{1, \ldots, p\}$ properly, for each $k = 1, \ldots, p$, we define weights $w(\pi_{k\langle \mathbf{c} \rangle}, k) = w(\varrho_{k\langle \mathbf{c} \rangle}, k) = 1$ for every $\mathbf{c} \in \{0,1\}^{p_3}$, and threshold $h(k) = 1$.

Now we specify the computational dynamics of neural network N simulating the finite automaton A. At the beginning, the states of neurons from $\nu_0 \setminus \{\text{in}\}$ are placed in an initial state of A. Each bit x_i $(1 \leq i \leq n)$ of input word $\mathbf{x} = x_1, \ldots, x_n$, which is read by input neuron $\text{in} \in \nu_0$ at time instant $\tau(i-1)$ (i.e. $y_{\text{in}}^{(\tau(i-1))} = x_i$), is being processed by N within the desired period of $\tau = d + 4 = O(p2^{p_1}/e) = O(\sqrt{2^p}/e) = O(\sqrt{m}/e)$ time steps. The states of neurons in N are successively updated in the order following the architecture of layers. Thus, we define sets α_t of units updated at time instants $t \geq 1$ as $\alpha_{\tau(i-1)+1} = \nu_1$, $\alpha_{\tau(i-1)+j+1} = \nu_{12} \cup \nu_2$ for $j = 1, \ldots, d+1$, $\alpha_{\tau(i-1)+d+3} = \nu_{12} \cup \nu_2 \cup \nu_3$, and $\alpha_{\tau i} = \nu_0 \setminus \{\text{in}\}$, for $i = 1, \ldots, n$. Eventually, the output neuron $\text{out} \in \nu_0$ signals at time instant τn whether input word $\mathbf{x}$ belongs to underlying language L, that is, $y_{\text{out}}^{(\tau n)} = 1$ iff $\mathbf{x} \in L$.

The size of N simulating the finite automaton A with m states can be expressed as $s = |\nu_0| + |\nu_1| + |\nu_2| + |\nu_3| = p + 1 + 2^{p_2} + 2 + 8p2^{p_1} + 2(d+1) + p2^{p_3} = O(\sqrt{2^p}) = O(\sqrt{m})$ in terms of m, which matches the known lower bound [2, 3]. Finally, the energy consumption can be bounded for particular layers as follows. Layer ν_0 can possibly require all $p+1$ units to fire for storing the binary encoding of a current automaton state. Moreover, there is only one active unit among neurons in ν_{11} which serve for evaluating all possible monomials over input variables $\mathbf{v}$, and also only one control unit from $\nu_{12} \cup \nu_{22}$ fires at one time instant. In addition, we know that the energy consumption by ν_{21} is at most $e + 2$, and at most p neurons among $\pi_{k\langle \mathbf{c} \rangle}, \varrho_{k\langle \mathbf{c} \rangle}$ from ν_3 fire (one for each $k = 1, \ldots, p$). Altogether, the global energy consumption of N is bounded by $e + 2p + 5 = O(e + \log s) = O(e)$ as $e = \Omega(\log s)$ is assumed. This completes the proof of the theorem. □

5 Conclusions

We have, for the first time, applied the energy complexity measure to recurrent neural nets. This measure has recently been introduced and studied for feedforward perceptron networks. The binary-state recurrent neural networks recognize exactly the regular languages so we have investigated their energy consumption of simulating the finite automata with the asymptotically optimal number of neurons. We have presented a low-energy implementation of finite automata by optimal-size neural nets with the tradeoff between the time overhead for processing one input bit and the energy varying from the logarithm

to the full network size. In the full paper, we will also present lower bounds on the energy complexity of neural network automata. In particular, for time overhead $\tau = O(1)$, the energy satisfies $e \geq s^{\varepsilon}$ for some real constant ε such that $0 < \varepsilon < 1$, and for infinitely many s, while for $\tau = O(\log^{\varepsilon} s)$, we have shown that $e = \Omega_{\infty}(s^{\log\log s/\log^{\eta} s})$ for any $\eta > \varepsilon$. An open problem remains for further research whether these bounds can be improved.

References

1. Alon, N., Dewdney, A.K., Ott, T.J.: Efficient simulation of finite automata by neural nets. Journal of the ACM 14(2), 495–514 (1991)
2. 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)
3. Indyk, P.: Optimal simulation of automata by neural nets. In: Mayr, E.W., Puech, C. (eds.) Proceedings of the STACS 1995 Twelfth Annual Symposium on Theoretical Aspects of Computer Science. LNCS, vol. 900, pp. 337–348 (1995)
4. Lennie, P.: The cost of cortical computation. Current Biology 13(6),493–497 (2003)
5. Lupanov, O.: On the synthesis of threshold circuits. Problemy Kibernetiki 26, 109–140 (1973)
6. Minsky, M.: Computations: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs (1967)
7. Siegelmann, H.T., Sontag, E.D.: Computational power of neural networks. Journal of Computer System Science 50(1), 132–150 (1995)
8. Šíma, J., Orponen, P.: General-purpose computation with neural networks: A survey of complexity theoretic results. Neural Computation 15(12), 2727–2778 (2003)
9. Šíma, J., Wiedermann, J.: Theory of neuromata. Journal of the ACM 45(1), 155–178 (1998)
10. Suzuki, A., Uchizawa, K., Zhou, X.: Energy and fan-in of threshold circuits computing Mod functions. In: Ogihara, M., Tarui, J. (eds.) Proceedings of the TAMC 2011 Eight Annual Conference on Theory and Applications of Models of Computation. LNCS, vol. 6648, pp. 154–163 (2011)
11. Uchizawa, K., Douglas, R., Maass, W.: On the computational power of threshold circuits with sparse activity. Neural Computation 18(12), 2994–3008 (2006)
12. Uchizawa, K., Nishizeki, T., Takimoto E.: Energy and depth of threshold circuits. Theoretical Computer Science 411(44-46), 3938–3946 (2010)
13. Uchizawa, K., Takimoto, E.: Exponential lower bounds on the size of constant-depth threshold circuits with small energy complexity. Theoretical Computer Science 407(1-3), 474–487 (2008)
14. Uchizawa, K., Takimoto, E.: Lower bounds for linear decision trees via an energy complexity argument. In: Murlak, F., Sankowski, P. (eds.): Proceedings of the MFCS 2011 Thirty-Sixth International Symposium on Mathematical Foundations of Computer Science. LNCS, vol. 6907, pp. 568-579 (2011)
15. Uchizawa, K., Takimoto, E., Nishizeki, T.: Size-energy tradeoffs for unate circuits computing symmetric Boolean functions. Theoretical Computer Science 412(8-10), 773–782 (2011)