Efficient Simulation of Finite Automata by Neural Nets

NOGA ALON

Tel Aviv University, Ramat Aviv, Tel Aviv, Israel

A. K. DEWDNEY

University of Western Ontario, London, Ontario, Canada

AND

TEUNIS J. OTT

Bell Communications Research, Morristown, New Jersey

Abstract. Let $K(m)$ denote the smallest number with the property that every m-state finite automaton can be built as a neural net using $K(m)$ or fewer neurons. A counting argument shows that $K(m)$ is at least $\Omega((m \log m)^{1/3})$, and a construction shows that $K(m)$ is at most $O(m^{3/4})$. The counting argument and the construction allow neural nets with arbitrarily complex local structure and thus may require neurons that themselves amount to complicated networks. Mild, and in practical situations almost necessary, constraints on the local structure of the network give, again by a counting argument and a construction, lower and upper bounds for $K(m)$ that are both linear in m.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*automata, relations between models*

General Terms: Neural nets, Finite automata

Additional Key Words and Phrases: Mealy Machines

1. *Introduction*

It has been known since 1954 (see Minsky [9, 10]) that any (deterministic) finite automaton can be realized or simulated by a neural net of the original type specified by McCullough and Pitts [7]. The simulation is quite simple: A finite automaton with m states is replaced by a network containing $2m + 1$ neurons.

Part of this research was done while N. Alon and A. K. Dewdney were visiting Bellcore in the Summer of 1987.

Authors' addresses: N. Alon, Department of Mathematics, Sackler Faculty of Exact Sciences, Tel Aviv University, Ramat Aviv, Tel Aviv, Israel 69978; A. K. Dewdney, Department of Computer Science, University of Western Ontario, London, Ont., Canada N6A 5B7; T. J. Ott, Bell Communications Research, 435 South Street, Morristown, NJ 07960.

However, if one considers that a network containing n neurons is capable of 2^n states, one might wonder whether a more efficient simulation is possible. In particular, it might be expected that an arbitrary m-state automaton could be simulated by a neural net having $O(\log m)$ neurons. This turns out to be very far from the truth. Counting arguments developed here show that in general one cannot expect to use fewer neurons than some low-order fractional power of m, along with a logarithmic factor. Depending on what restrictions are placed on the network, one might do no better than $O((m \log m)^{1/3})$. With mild restrictions the exponent changes to $\frac{1}{2}$. Finally, applying the kind of restrictions that would typify a realistic hardware implementation of a network, we find ourselves back in linear territory.

The final result is somewhat disappointing since it means that parallel hardware inspired by this somewhat primitive neural model is unlikely to host arbitrary finite automata with uniform high efficiency: Some automata will require networks that require about as many neurons as the automata have states! On the other hand, certain automata may be simulable by networks having a logarithmic number of neurons. A careful reading of this paper may produce the impression that more complicated neural network models will suffer from the same inherent inefficiency. We see nothing to contradict this impression at the present time, see also the conclusions section in this paper. As one of the referees put it: "The paper gives the impression of establishing the disturbing result that there may have to be as many neurons in our head as there are possible states to our mind". Although it is totally unclear how to define the "state" of our mind, and also while the neurons in our brains may very well be more complicated than the "neurons" in this paper, we can not help but agree that this is the kind of conclusion our findings lead up to.

The finite state machines studied in this paper are of the type called "Mealy Machines," see Mealy [8] or Hopcroft and Ullman [4]. Section 3 contains a self-contained brief discussion of Mealy machines. In order to prevent confusion with the closely related "Neural Nets," as currently defined in Cognitive Psychology and Artificial Intelligence (see Rumelhart and McClelland [11]), we have renamed Neural Nets and Neurons (as used in this paper) "Threshold Machines" and "Threshold Cells," respectively. Definitions will be given in Section 4. It will be seen that classical AND, OR, and NOT gates are special cases of the threshold cells used in this paper, so that the negative results in this paper carry over to machines based on such classical gates. It is not clear whether this is also true for the positive (constructive) results.

The problem addressed thus becomes: How many threshold cells may be needed to build a threshold machine which acts "exactly like" a prespecified m-state machine.

The exact formulation of this question of course depends on the definitions as given in the Sections 3 and 4 and on a convention, to be given in Section 3, for when two machines are said to be acting the same. A more intuitive description might be as follows:

At each point in time t ($t = 0, 1, \ldots$: time is discrete) each of the threshold cells or neurons of the threshold machine is in one of two states: either it does, or it does not, fire. If the machine has K cells then, since each of the cells is in one of two states, the whole machine has 2^K possible states. Hence, every K-cell threshold machine is a finite state machine with 2^K states.

On the other hand, every m-state Mealy machine can be built as a threshold machine (see Minsky [9, 10] or Section 5 in this paper). Clearly, there exist m-state machines which can be built using only $\log m$ (take $m = 2^K$) threshold cells. Would this be almost true for every m-state machine? Unfortunately, the answer is negative. The actual number of threshold cells needed to build a specific machine depends on how complicated individual cells are allowed to be (this is expressed in the size of their weight alphabet, their threshold alphabet, and their Fan-in and Fan-out, see Section 4). With more complicated cells, the same machine can be built with fewer cells. Typical results are:

THEOREM 1.1. (see Dewdney [1]). *There exists a $C_1 > 0$ such that as long as there are no restrictions on how complicated individual cells can be, every m-state Mealy Machine can be built with*

$$\text{at most } C_1 \cdot m^{3/4} \text{ threshold cells.} \tag{1.1}$$

THEOREM 1.2. *There exists a $C_2 > 0$ such that even if there are no restrictions on how complicated individual cells can be, for every sufficiently large m there exists an m-state Mealy Machine which in order to be built needs*

$$\text{at least } C_2 \cdot (m \log m)^{1/3} \text{ threshold cells.} \tag{1.2}$$

The lower bound in (1.2) becomes higher when restrictions are placed on how complicated the individual cells can be.

Even with only mild restrictions on those cells, the number of cells needed to build a m-state machine can become linear in m.

For example:

THEOREM 1.3. *If either there is a limit on the Fan-in of individual cells, or there are simultaneous limits on the Fan-out of cells, the weight alphabet, and the threshold alphabet, then there exists a $C_3 > 0$ such that for every sufficiently large m there exists an m-state Mealy Machine which in order to be built as a threshold machine satisfying those restrictions needs*

$$\text{at least } C_3 \cdot m \text{ threshold cells.} \tag{1.3}$$

The following result shows that at least for the second set of conditions Theorem 1.3 is about as sharp as possible.

THEOREM 1.4. *Every m-state Mealy machine can be built as a threshold machine that contains only $2m + 1$ threshold cells. This threshold machine has a fixed weight alphabet $\{-1, 0, 1\}$ and a fixed threshold alphabet $\{1, 2\}$ (both independent of m). This threshold machine also has the property that all cells have a Fan-out of either 2 or 3.*

The proof of Theorem 1.4 is by construction. The construction in Section 5 is essentially the construction in Minsky [9], modified because we have binary input. The threshold machine constructed in the proof need not have bounded fan-in for its cells. This means that we can not exclude the possibility that a

restriction on the fan-in of cells leads to m-state machines which in order to be built need a superlinear (in m) number of cells.

Section 2 describes the idea of the proof of Theorems 1.2 and 1.3. Section 3 discusses Mealy Machines and Section 4 discusses threshold machines and proves Theorems 1.2 and 1.3. Theorems 1.1 and 1.4 are known results (see Dewdney [1]). Section 5 gives an outline of the proofs of these theorems. The proofs are based on constructions. Not surprisingly, the construction in the proof of Theorem 1.1 uses weight alphabets and threshold alphabets that depend on m.

2. *The Lower Bound*

Definition 2.1. For each natural number m, $K(m)$ is the smallest number such that every Mealy Machine with m or fewer states can be built as a threshold machine using $K(m)$ or fewer threshold cells. Theorems 1.1 and 1.2 say that

$$C_2(m \log m)^{1/3} \leq K(m) \leq C_1 m^{3/4}. \tag{2.1}$$

The proof of the lower bounds in (2.1) works by counting the number of "really different" m-state Mealy Machines and the number of "really different" Mealy Machines that can be built as a threshold machine using K or fewer cells. We still need to define when two Mealy Machines are "really different" (the definition actually used will be given in Section 3), but for any reasonable definition we trivially have:

Let $L(m)$ be the number of "really different" Mealy Machines with m or fewer states. Let $U(K)$ be the number of different Mealy Machines that can be built as a threshold machine using K or fewer cells. Then:

$$U\big(K(m)\big) \geq L(m). \tag{2.2}$$

In Section 3, we define when machines are "really different" and we derive a lower bound for $L(m)$. In Section 4, we derive an upper bound for $U(K)$, and we combine the lower and upper bounds to obtain bounds for $K(m)$. In Section 4, we also obtain alternative bounds for $U(K)$, with additional restrictions on what kind of threshold machines are allowed, and use these to obtain stronger lower bounds for $K(m)$, based of course on stronger requirements.

3. *Mealy Machines*

An m-state, binary Mealy Machine is a deterministic finite automaton which at each point in time t ($t = 0, 1, \ldots$: time is discrete) is in a state $S(t) \in \{1, 2, \ldots, m\}$ ($\{1, 2, \ldots, m\}$ is the state-space) and which at each point in time t receives an input $I(t) \in \{0, 1\}$ ($\{0, 1\}$ is the input alphabet) and generates an output $O(t) \in \{0, 1\}$ ($\{0, 1\}$ is the output alphabet).

The output $O(t)$ depends on the state $S(t)$ and the input $I(t)$ through maps

$$g_i\colon \{1, 2, \ldots, m\} \to \{0, 1\} \qquad \big(0 \in \{0, 1\}\big), \tag{3.1}$$

where

$$O(t) = g_{I(t)}\big(S(t)\big). \tag{3.2}$$

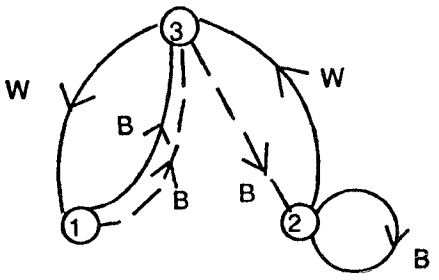

FIGURE 1

The new state $S(t+1)$ depends on the old state $S(t)$ and the input $I(t)$ through the maps

$$f_i\colon \{1,2,\ldots,m\} \to \{1,2,\ldots,m\} \qquad (i \in \{0,1\}), \tag{3.3}$$

where

$$S(t+1) = f_{I(t)}(S(t)). \tag{3.4}$$

A Mealy Machine is entirely determined by its state space $\{1,2,\ldots,m\}$, the input alphabet, the output alphabet, and the maps f_i, g_i. Throughout this paper, both input alphabet and output alphabet always are $\{0, 1\}$ ("one bit input, one bit output").

The input alphabet can of course be replaced by any other two symbols, for example {solid, dotted}, and in the same way the output alphabet can be replaced by {bell, whistle}. Thus, the m-state Mealy machine can be thought of as a directed graph with m nodes, where every node has two outgoing arcs: one "solid" and one "dotted," and where each of the $2m$ arcs is labeled with either a "bell" or a "whistle." An example, with $m = 3$, is shown in Figure 1.

If, at time t, the machine is in state s and the input is "solid," then the machine moves along the solid outgoing arc from state s to the new state, and the output is either a bell or a whistle, depending on the label of that arc.

A very interesting problem is how to decide whether two Mealy Machines are the same or not. For example, we could say that two Mealy machines, say machine 1 with $m^{(1)}$, $f_i^{(1)}$, $g_i^{(1)}$ and machine 2 with $m^{(2)}$, $f_i^{(2)}$, $g_i^{(3)}$ are "the same" only if

$$m^{(1)} = m^{(2)} \tag{3.5}$$

and

$$f_i^{(1)}(s) = f_i^{(2)}(s), \qquad g_i^{(1)}(s) = g_i^{(2)}(s), \tag{3.6}$$

both for all s and all i.

This very simple definition leads to the obvious conclusion that there are

$$\text{exactly } (2m)^{2m} \text{ different } m\text{-state machines.} \tag{3.7}$$

This definition clearly is not acceptable. For example, a relabeling of the m states certainly does not produce a different machine, while with high probability it changes some $f_i(s)$ or $g_i(s)$. According to this train of thought,

$(2m)^{2m} \cdot (m!)^{-1}$ is an appealing approximation for the number of different m-state machines. For a more rigorous treatment, we ask ourselves the question: When is machine 2 an implementation of machine 1? In other words, when is it true that any copy of machine 1 can be replaced by a copy of machine 2?

Definition 3.1. Machine 2, with $m^{(2)}$, $f_t^{(2)}$, $g_t^{(2)}$, is an implementation of machine 1 with $m^{(1)}$, $f_t^{(1)}$, $g_t^{(1)}$ whenever there exists a map (called the canonical map) Can: $\{1, 2, \ldots, m^{(1)}\} \to \{1, 2, \ldots, m^{(2)}\}$ with the property that whenever machine 1 starts in any state $s^{(1)} \in \{1, 2, \ldots, m^{(1)}\}$ ($S^{(1)}(0) = s^{(1)}$) and machine 2 starts in state $s^{(2)} = Can\ (s^{(1)})$ ($S^{(2)}(0) = s^{(2)} = Can(s^{(1)})$) then for every common input sequence $(I(t))_{t=0}^{\infty}$, as long as both machines receive that input sequence ($I^{(1)}(t) = I^{(2)}(t) = I(t)$ for all t) both machines generate the same output sequence ($O^{(1)}(t) = O^{(2)}(t)$ for all t). (It is sometimes preferable to define Can as a point-to-set map that assigns to each $s^{(1)} \in \{1, 2, \ldots, m^{(1)}\}$ a nonempty subset $Can(s^{(1)})$ of $\{1, 2, \ldots, m^{(2)}\}$. This does not change anything in any essential way.)

This paper does not need a solution to the problem of how to determine, for a given pair of machines, whether one is an implementation of the other, and of how to find the canonical map. However, it is easy to find the following characterization:

Machine 2 is an implementation of machine 1 if and only if there exists a point-to-set map Can that assigns to each $s^{(1)} \in \{1, 2, \ldots, m^{(1)}\}$ a nonempty subset $Can(s^{(1)})$ of $\{1, 2, \ldots, m^{(2)}\}$ and that satisfies

$$g_t^{(1)}(s^{(1)}) = g_t^{(2)}(s^{(2)}) \qquad \text{for all} \quad s^{(2)} \in Can(s^{(1)}),$$

and

$$\bigcup_{s^{(2)} \in Can(s^{(1)})} \{f_t^{(2)}(s^{(2)})\} \subset Can(f_t^{(1)}(s^{(1)})).$$

The requirement in the following definition is probably slightly stronger than necessary. It is chosen to facilitate the argument in Section 4. It formalizes the notion of "really different" used in Section 2:

Definition 3.2. Two machines, say with $m^{(j)}$, $f_t^{(j)}$, $g_t^{(j)}$ ($j = 1, 2$) are "divergent" if for every pair $(s^{(1)}, s^{(2)})$, $s^{(j)} \in \{1, 2, \ldots, m^{(j)}\}$, there exists a common input sequence $I(t)$ such that if machine j starts in state $s^{(j)}$ ($S^{(j)}(0) = s^{(j)}$) and both machines get input sequence $(I(t))_{t=0}^{\infty}$, then for some $t \geq 0$

$$O^{(1)}(t) \neq O^{(2)}(t). \tag{3.8}$$

Suppose we have three machines, with $m^{(j)}$, $f_t^{(j)}$, $g_t^{(j)}$. Suppose that the machines 1 and 2 are "divergent" (in the sense of Definition 3.2) and that machine 3 is an implementation of both machines 1 and 2. We thus have two canonical maps Can_j: $\{1, 2, \ldots, m^{(j)}\} \to \{1, 2, \ldots, m^{(3)}\}$ ($j = 1, 2$). Clearly, $Can_j(\{1, 2, \ldots, m^{(j)}\})$ is a subset of $\{1, 2, \ldots, m^{(3)}\}$. Equally clearly, the fact that the machines 1 and 2 are divergent means that these two subsets of $\{1, 2, \ldots, m^{(3)}\}$ are disjoint.

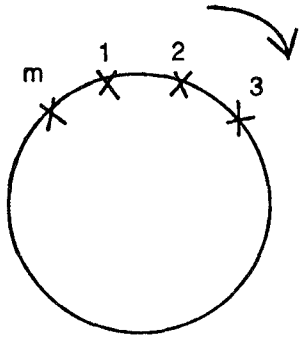

FIGURE 2

By repeating this argument we prove:

LEMMA 3.1. *Suppose there exists a set of U Mealy Machines that are pairwise divergent, and suppose there exists an m-state machine which is an implementation of all those U machines. Then*

$$U \leq m. \tag{3.9}$$

We are now ready to proceed to the main topic of this section and obtain two lower bounds for the number of (pairwise) divergent m-state Mealy Machines that can be found.

The first lower bound is weaker than the second. We present it because its proof nicely illustrates the concepts involved.

THEOREM 3.1. *For every $m \geq 1$, there exists a system of 2^m (pairwise) divergent m-state Mealy Machines.*

PROOF. The proof consists of establishing a system of 2^m (pairwise) divergent m-state Mealy Machines. This is done by describing the maps f_i, g_i that are allowed. We only allow maps f_i for which

$$f_i(s) = s + 1 \bmod m \qquad \text{for all} \quad 1 \leq s \leq m, \text{ all } i. \tag{3.10}$$

Pictorially (see Figure 2) this means that the m states are arranged in a circle and, independent of the inputs, the state of the system moves around the circle.

Further, we only allow maps g_0 for which

$$g_0(s) = \delta_{1,s} \text{ (Kronecker delta)}, \tag{3.11}$$

and finally, we do not put any restrictions on the map $g_1(\cdot)$. Since, for each s, there are two possible values of $g_1(s)$, this gives us exactly 2^m machines. All we need to do is prove that these 2^m machines are pairwise divergent. This will be done by contradiction.

Suppose that two of these machines, say machine 1 and machine 2, are not "divergent." Then there exist states $s^{(j)}$, $j = 1, 2$, such that if machine j starts in state $s^{(j)}$ and both machines get the same (arbitrary) input sequence, then both get the same output sequence.

First, give both machines input sequence $(0, 0, \ldots)$ (zeros only). At time t, machine j is in state $s^{(j)} + t \bmod m$ and, by (3.11), has output $\delta_{1,s^{(j)}+t \bmod m}$. Hence, $s^{(1)} = s^{(2)} = s_0$. Next, choose any $\tau \in \{1, 2, \ldots, m\}$ and let $d = \tau - s_0 \bmod m$ $(0 \leq d \leq m - 1)$. Give both machines as input d zeros followed by a

one. At time $t = d$, both machines are in state τ and since their outputs are the same we have $g_1^{(1)}(\tau) = g_1^{(2)}(\tau)$. Since τ was arbitrary, the two machines therefore have identical maps $g_1(\cdot)$. This completes the proof. □

THEOREM 3.2. *If m is prime, then there exists a system of $(2m)^m (2^m - 2)/m$ (pairwise) divergent m-state Mealy Machines.*

PROOF. Our constraints on the maps f_i, g_i now are as follows:

$$f_0(s) = s + 1 \bmod m, \tag{3.12}$$

$$g_0(s), \text{ for } s = 1, 2, \ldots, m, \text{ are not all the same}, \tag{3.13}$$

$$f_1(\cdot): \text{arbitrary}, \tag{3.14}$$

$$g_1(\cdot): \text{arbitrary}. \tag{3.15}$$

The constraints (3.12)–(3.15) define a class of Mealy machines. Clearly, there are $(2m)^m(2^m - 2)$ machines in this class. The factor m^{-1} will be introduced, and the theorem will be proven, by showing that if any two machines in this class are not divergent then one is a rotation of the other, in the sense that there exists a d, $0 \le d \le m - 1$, such that if the states of machine 2 are relabeled: state s is relabeled as state $s + d \bmod m$, then after the relabeling the machines 1 and 2 have identical $f_i(\cdot)$ and $g_i(\cdot)$, see also Figure 3.

Suppose two machines, say machines 1 and 2, from the class are not divergent. That means there are states $s^{(1)}$, $s^{(2)}$ such, that if machine j starts in state $s^{(j)}$ and both machines receive the same, but arbitrary, input stream then their output streams are identical. By doing a rotation or relabeling of the nodes, this time for *both* machines, we can ensure that $s^{(1)} = s^{(2)} = 1$.

First, give both machines input stream (0, 0, ...) (zeros only). Both machines, at time t, are in state $1 + t \bmod m$ and the outputs at time t are $g_0^{(1)}(1 + t \bmod m)$ and $g_0^{(2)}(1 + t \bmod m)$, which must be equal. Hence:

$$g_0^{(1)}(s) = g_0^{(2)}(s) = g_0(s) \qquad \text{for all } s. \tag{3.16}$$

Next, give as input $s - 1$ zeros followed by a one. At time $s - 1$, both machines are in state s and the outputs are $g_1^{(j)}(s)$, which must be equal, so that

$$g_1^{(1)}(s) = g_1^{(2)}(s) = g_1(s) \qquad \text{for all } s. \tag{3.17}$$

Finally, choose any s_0, $1 \le s_0 \le m$ and give as common input $s_0 - 1$ zeros, then a one, and then further only zeros. At time $s_0 - 1$, both machines are in state s_0. At time s_0, machine j is in state $f_1^{(j)}(s_0)$, and at time $s_0 + t$ $(t \ge 0)$ machine j is in state $(f_1^{(j)}(s_0) + t) \bmod m$. We need to prove that $f_1^{(1)}(s_0) = f_1^{(2)}(s_0)$. Let

$$s_j = f_1^{(j)}(s_0) \tag{3.18}$$

and let

$$d = s_2 - s_1 \bmod m, \qquad 0 \le d \le m - 1. \tag{3.19}$$

It will turn out that $d = 0$.

Because of the way f_0 is defined,

$$g_0(s_1 + t \bmod m) = g_0(s_1 + d + t \bmod m) \qquad \text{for all } t \ge 0. \tag{3.20}$$

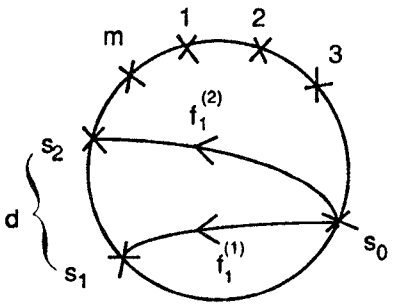

FIGURE 3

In particular:

$$g_0(s_1) = g_0(s_1 + d \bmod m) = g_0(s_1 + 2d \bmod m) = \cdots \tag{3.21}$$

and in fact

$$g_0(s_1) = g_0(s_1 + kd \bmod m) \qquad \text{for} \quad \text{all integer } k. \tag{3.22}$$

Since m is prime, $d \neq 0$ in (3.22) implies that $g_0(s) = g_0(s_1)$ for all s, which contradicts (3.13). Hence, $d = 0$, that is, $f_1^{(1)}(s_0) = f_1^{(2)}(s_0)$, which completes the proof. □

4. *Threshold Machines*

A K-cell Threshold machine has $K + 1$ threshold cells, numbered $0, 1, \ldots, K$, interconnected by lines of various weights as explained below. At each point in time $t = 0, 1, \ldots,$ each cell either does or does not fire. We use variables $x_i(t)$ to describe this:

$$x_i(t) = \begin{cases} 1 & \text{cell } i \text{ fires at time } t, \\ 0 & \text{cell } i \text{ does not fire at time } t, \end{cases} \tag{4.1}$$

$x_0(t) = I(t)$, the input at time t, and this variable is externally given: Cell zero is the input cell. For $j \geq 1$, the value of $x_j(t)$ depends on $(x_i(t-1))_{i=0}^{K}$ through weights $w_{i,j}$, $0 \leq i \leq K$, $1 \leq j \leq K$, and thresholds θ_j, $1 \leq j \leq K$. Many authors require those weights and thresholds to be integer. In this paper we do not put any restrictions on the weights and thresholds (apart from the fact that they must be real). The dynamics of the threshold machine is given by

$$x_j(t+1) = \begin{cases} 1 & \text{if} \quad \sum_{i=0}^{K} x_i(t) w_{i,j} \geq \theta_j, \\ 0 & \text{else}, \end{cases} \tag{4.2}$$

hence, the names threshold machine and threshold logic. Clearly, every AND, OR, or NOT gate can be implemented by a single threshold cell (or threshold gate), while every threshold cell can be implemented using a (possibly large) number of AND, OR, and NOT gates. Hence, all the negative results in this paper translate into negative results about the power of machines based on AND, OR, and NOT gates. The positive (constructive) results do not necessarily carry over in the same way. Outputs are given by

$$O(t) = x_K(t), \tag{4.3}$$

i.e., cell K is chosen to be the output neuron or output cell. This definition shows that the input cell is not really a threshold cell: There are no inputs into this cell and no threshold associated with this cell.

With this description, it is clear that every threshold K-machine indeed is a 2^K-state Mealy Machine. It is easily seen (see Section 5) that on the other hand every m-state Mealy Machine can be built as a threshold machine with $2m + 1$ cells.

A threshold machine is entirely described by K and its weights $w_{i,j}$ and thresholds θ_j. On the other hand, two different sets of weights and thresholds can easily generate the same machine. Suppose we have the machines 1 and 2 with the same numbers of cells and with weights and thresholds $w_{i,j}^{(v)}$, $\theta_j^{(v)}$, $v = 1, 2$. We define these machines to be the same if for every $j \in \{1, 2, \ldots, K\}$, for every sequence $(\delta_0, \delta_1, \ldots, \delta_K) \in \{0, 1\}^{K+1}$,

$$\sum_{i=0}^{K} \delta_i w_{i,j}^{(1)} \geq \theta_j^{(1)} \Leftrightarrow \sum_{i=0}^{K} \delta_i w_{i,j}^{(2)} \geq \theta_j^{(2)}. \tag{4.4}$$

Clearly, two machines are the same under this definition if and only if they have the property that for every subset S_0 of $\{1, 2, \ldots, K\}$ and for every input sequence $(I(t))_{t=0}^{\infty}$, if for both machines at time $t = 0$ exactly the neurons in S_0 fire, and both machines receive the same input sequence $(I(t))_{t=0}^{\infty}$, then at every point in time $t \geq 0$ the two machines have exactly the same set S_t of firing neurons. Also, with this definition, "being the same" defines an equivalence relation on the set of all K-cell threshold machines. Machines not in the same equivalence class can still be "the same" in some weaker sense than defined above (see Section 3). However, the total number of equivalence classes certainly gives an upper bound for the number of in any reasonable sense different K-cell machines. We have the following result:

LEMMA 4.1. *The number of equivalence classes in the relation above is less than* $2^{(K+1)^2K+K}$.

PROOF. We choose a specific j, $1 \leq j \leq K$, and study the number of ways we can choose $(w_{i,j})_{i=0}^{K}$ and θ_j. Any hyperplane

$$\sum_{i=0}^{K} x_i w_i = \theta \tag{4.5}$$

generates two such ways: $w_0, \ldots, w_K; \theta$ and $-w_0, \ldots, -w_K; -\theta$. Two such hyperplanes are different (in the sense of the definition above) if they split the 2^{K+1} points in $\{0, 1\}^{K+1}$ in two different ways into two subsets. A result by Harding [2] shows that a set of n points in $\mathbf{R}^d$ can be divided by a hyperplane into two subsets in at most

$$\sum_{i=0}^{d} \binom{n-1}{i} \tag{4.6}$$

different ways. For $N \geq 1$, fixed, we easily prove by induction to L that $\sum_{i=0}^{L} \binom{2^N-1}{i} \leq 2^{NL}$. Hence, for j fixed, we can choose $(w_{i,j}, \theta_j)$ in at most

$$2 \sum_{i=0}^{K+1} \binom{2^{K+1}-1}{i} \leq 2^{(K+1)^2+1} \tag{4.7}$$

effectively different ways. Since we can do this for $j = 1, 2, \ldots, K$, we immediately have that the total number of effectively different ways to choose the $w_{i,j}$, θ_j is at most $(2^{(K+1)^2+1})^K = 2^{(K+1)^2K+K}$.

Each K-cell threshold machine is a 2^K state Mealy machine which by Lemma 3.1 implements at most 2^K (pairwise) divergent machines. With Lemma 4.1, this leads to:

THEOREM 4.1. *The set of all Mealy machines that can be implemented by a K-cell threshold machine can not contain more than* $2^{(K+1)^3}$ *pairwise divergent machines.*

Namely, in the language of Section 2, we have

$$U(K) \leq 2^{(K+1)^2K} \cdot 2^{2K} < 2^{(K+1)^3}. \tag{4.8}$$

Combining, as promised eq. (4.8) with Theorem 3.1, we get

$$2^{(K(m)+1)^3} \geq U\big(K(m)\big) \geq L(m) \geq 2^m, \tag{4.9}$$

or

$$K(m) \geq m^{1/3} - 1. \tag{4.10}$$

This result is not quite as good as promised in Theorem 1.2. To prove Theorem 1.2, we must combine Theorem 4.1 with Theorem 3.2. Let $p(m)$ be the largest prime $\leq m$. Clearly, any Mealy machine with $p(m)$ or fewer states can be implemented as a threshold machine with $K(m)$ or less cells. Hence

$$\begin{aligned} 2^{(K(m)+1)^3} &\geq U\big(K(m)\big) \geq L\big(p(m)\big) \\ &\geq \big(2p(m)\big)^{p(m)} \frac{2^{p(m)} - 2}{p(m)}, \end{aligned} \tag{4.11}$$

or for m sufficiently large:

$$\begin{aligned} \big(K(m) + 1\big)^3 \log 2 &> \big(p(m) - 1\big)\big(\log 4 + \log p(m)\big), \\ K(m) &\geq \left(\big(p(m) - 1\big) \cdot \left(\frac{\log p(m)}{\log 2} + 2\right)\right)^{1/3} - 1. \end{aligned} \tag{4.12}$$

Since it is well known (see, e.g., Hardy and Wright [3]) that

$$\lim_{m \to \infty} \frac{p(m)}{m} = 1, \tag{4.13}$$

this proves Theorem 1.2.

Theorem 4.1 puts no restriction on the set of weights that are allowed, the set of thresholds that are allowed, or the Fan-in or Fan-out of cells. This means that it is very hard, probably impossible, to "mass produce" one standard generic threshold cell that then can be used to build all desired threshold machines.

Next, we investigate the consequences of being restricted to a prespecified set of weights, independent of the number of cells or the number of states in the machine.

THEOREM 4.2. *Consider a prespecified set* $W = \{w_1, \ldots, w_L\}$ *and consider the family of threshold machines that have weights from* W *only. We call* W *the weight alphabet of this family of machines. On this family, the function* U *satisfies*:

$$U(K) \leq |W|^{K(K+1)} \cdot (2^{K+1} + 1)^K 2^K. \tag{4.14}$$

By combining this with Theorem 3.2, *we get exactly as in* (4.11)–(4.13):

$$K(m) \geq C_4 (m \log m)^{1/2}. \tag{4.15}$$

PROOF. We only need to prove (4.14). In constructing the threshold machine, for each of the $K(K+1)$ weights there are $|W|$ possible choices, this gives the factor $|W|^{K(K+1)}$. For each j, $\sum_i \delta_i w_{i,j}$ $((\delta_0, \delta_1, \ldots, \delta_K) \in \{0, 1\}^{K+1})$ has at most $2^{(K+1)}$ different values so that there are at most $2^{K+1} + 1$ effectively different values for θ_j. This gives the factor $(2^{K+1} + 1)^K$. Finally, the factor 2^K comes from Lemma 3.1.

A further practical restriction on threshold machines is the Fan-in and the Fan-out of cells (see, e.g., Savage [12]). The Fan-in of cell j is defined as

$$\left|\{i : w_{i,j} \neq 0\}\right|, \tag{4.16}$$

and the Fan-out of cell i is defined as

$$\left|\{j : w_{i,j} \neq 0\}\right|. \tag{4.17}$$

The following two results show that a constraint on the fan-in of cells is much more restrictive than a constraint on the fan-out of cells:

THEOREM 4.3. *Suppose there exists a prespecified number* F *and cells* $i \geq 1$ *are only allowed to have Fan-out* $\leq F$ (*the input cell is allowed unlimited fan-out*). *Then*:

$$U(K) \leq \binom{K}{F}^K \cdot 2^{K^2(F+1)^2} \cdot 2^{2K} \tag{4.18}$$

and there exists a $C_5 > 0$ *such that for every sufficiently large* m *there exists an* m*-state machine which* (*with the restriction on Fan-out above*) *in order to be built as a threshold machine needs*

$$\text{at least } C_5 (m \log m)^{1/2} \text{ threshold cells}. \tag{4.19}$$

PROOF. For each cell $i \geq 1$, choose F cells j for which $w_{i,j}$ is allowed to be nonzero. This gives the factor $\binom{K}{F}^K$. Given these choices, let A_j be the maximal possible Fan-in of cell j. Clearly,

$$A_j \geq 0, \qquad \sum_{j=1}^{K} A_j = KF + K = K(F+1), \tag{4.20}$$

so that

$$\sum_{j=1}^{K} A_j^2 \leq K^2 (F+1)^2. \tag{4.21}$$

By the same argument as in the proof of Lemma 4.1, we see that for any j the weights $w_{i,j}$ and the threshold t_j can be chosen in less than $2^{A_j^2+1}$ different ways. This, with (4.21), gives the factor $2^{K^2(F+1)^2+K}$, and Lemma 3.1 adds another factor 2^K. This completes the proof of (4.18). The proof now is completed by combining (4.18) with Theorem 3.2, using the fact that $\binom{K}{F}^K \leq (F!)^{-K} \cdot K^{FK}$. □

The following two results show that under certain circumstances the number of cells needed to build an m-state machine may grow linearly in m.

THEOREM 4.4. *Suppose there exists a prespecified number F, and only threshold machines where all the cells have a Fan-in $\leq F$ are allowed. Then*

$$U(K) \leq \left(\binom{K+1}{F} 2^{F^2+1}\right)^K \cdot 2^K \tag{4.22}$$

and there exists a $C_6 > 0$ such that for every sufficiently large m there exists an m-state machine that (with the restriction on Fan-in above) in order to be built as a threshold machine needs

$$\text{at least } C_6 \cdot m \text{ threshold cells.} \tag{4.23}$$

PROOF. First, we prove (4.22). For any cell $j \geq 1$, choose a set of F cells i for which $w_{i,j}$ is allowed to be nonzero. This gives the factor $\binom{K+1}{F}$. For that j, the same argument as used in Lemma 4.1 shows that θ_j and the nonzero $w_{i,j}$ can be chosen in at most 2^{F^2+1} effectively different ways. We can do this for every j (this gives the power K). The factor 2^K follows from Lemma 3.1. This proves (4.22). The bound (4.23) is obtained by combining (4.22) with Theorem 3.2. □

THEOREM 4.5. *Suppose we only allow threshold machines that satisfy the following three restrictions*:

(*i*) *Each cell i ($1 \leq i \leq K$) has Fan-out $\leq F$ (the input cell has unlimited Fan-out).*

(*ii*) *There exists a prespecified finite set T, and all thresholds θ_j must be from that set (T is the threshold alphabet that is allowed).*

(*iii*) *There exists a prespecified set $W = \{0, w_1, w_2, \ldots, w_L\}$, and all weights $w_{i,j}$ must be from that set. Then*:

$$U(K) \leq \left(\binom{K}{F} |W|^F\right)^K |W|^K T^K 2^K \tag{4.24}$$

and there exists a $C_7 > 0$ such, that for every sufficiently large m there exists an m-state machine that (with the restrictions above) in order to be built as a threshold machine needs

$$\text{at least } C_7 \cdot m \text{ threshold cells.} \tag{4.25}$$

PROOF. For each cell $i \geq 1$, choose F cells j for which w_{ij} is allowed to be zero. This gives the factor $\binom{K}{F}^K$. Each of these w_{ij} can have $|W|$ different

values, this gives the factor $|W|^{FK}$. The factor $|W|^{K}$ is due to the freedom in choosing $w_{0,j}$ $(1 \leq j \leq k)$, the factor T^K is due to the freedom in choosing the thresholds θ_j, and the factor 2^K follows from Lemma 3.1. This proves (4.24). The bound (4.25) is obtained by combining (4.24) with Theorem 3.2.

It is somewhat surprising that a restriction on the Fan-in of cells leads to a strong increase in $K(m)$, while a similar restriction on the Fan-out needs additional conditions before it leads to the same result. In an attempt to explain this asymmetry, we define the Fan-in and Fan-out of a state in a Mealy Machine:

The Fan-in of state s is the number of pairs (i, s') with $f_i(s') = s$.

The Fan-out of state s is the number of states in $\{f_i(s): i = 0, 1\}$.

Clearly, each state has Fan-out either 1 or 2, while the Fan-in can be anywhere between zero and $2m$ (both included). This observation may explain the asymmetry in the powers of restrictions on the Fan-in and Fan-out. An interesting question therefore is: Given that all states of an m-state Mealy machine have Fan-in $\leq F$, how many cells may be needed to build it as a threshold machine? An interesting related question is: How many (pairwise) divergent m-state Mealy machines can be found, if we allow only machines for which all states have Fan-in $\leq F$?

In Theorem 1.4, which will be proved in the next section, we promised a construction that builds every m-state Mealy machine using exactly $2m + 1$ cells. In light of the question raised above, it is interesting to observe that in that construction, for every state s with Fan-in F_s, the construction uses two cells $(s, 0)$ and $(s, 1)$, which each have Fan-in equal to $F_s + 1$. In addition, there is an output cell of which the Fan-in may be as high as $2m$.

5. *Constructing Efficient Neural Nets*

The fewer neurons that we are allowed in constructing a network that simulates an m-state Mealy machine, the more difficult the job becomes. For a Mealy machine on m states, there is no problem constructing a simulating network that has $2m + 2$ neurons. This includes the input neuron and, in the language of Section 4, it is therefore a $2m + 1$ cell machine.

In this construction, each state is represented by two neurons. State s $(1 \leq s \leq m)$ is represented by the neurons $(s, 0)$ and $(s, 1)$ and we build the threshold machine in such a way that at time t neuron (s, i) fires if and only if the original Mealy machine at time t is in state s and receives input i. The construction is illustrated in Figure 4. In addition to the input neuron and the $2m$ neurons obtained as above there is a $(2m + 1)$th neuron called the output neuron..

First, we describe the weights. For any two states x and s, $w_{(x,i),(s,0)} = w_{(x,i),(s,1)} \in \{0, 1\}$, and this equals 1 if and only if in the original finite state machine state x with input i leads to the next state s (in the language of Section 3: $f_i(x) = s$). For inputs, $w_{0,(s,0)} = -1$ and $w_{0,(s,1)} = +1$ for all s. For outputs, $w_{(s,i),2m+1} = g_i(s)$ ($g_i(s)$ as in Section 3). All weights not yet mentioned are equal to zero. The thresholds are defined by $\theta_{(s,1)} = 2$, $\theta_{(s,0)} = 1$, and $\theta_{2m+1} = 1$.

It is clear that if at time $t = 0$ exactly one of the neurons (s, i), $1 \leq s \leq m$, $0 \leq i \leq 1$, fires then at any time $t \geq 1$ exactly 1 of those neurons will fire, and the dynamics is exactly the same as that for the Mealy machine. It is not clear

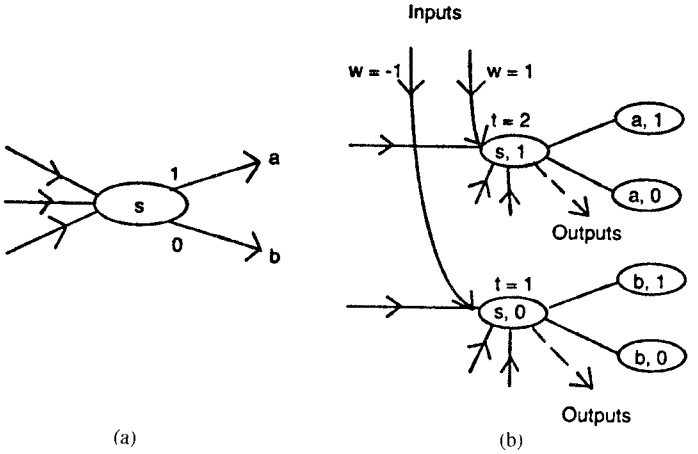

FIG. 4. (a) In-state machine. (b) Neural net

whether, if by accident multiple neurons fire simultaneously, we ever reach the situation where exactly one neuron fires. The outputs of the threshold machine just constructed lag one time unit behind the outputs of the Mealy machine.

Of the 2^{2m} states defined by the neurons (s, i), only $2m$ are used to implement the m-state machine. This enormous redundancy suggests that it must be possible to implement the machine using much fewer than $2m$ neurons. Indeed, we know that some m state machines can be built using only log m neurons, and Theorem 1.1 shows that there always is a significant improvement available over the number $2m + 1$. However, the Theorems 1.2, 1.3, 4.2, 4.3, 4.4, and 4.5 also show that often we can not get at all close to the ideal of log m, and in particular when there are limits on the nature of the threshold cells that can be used, the possible improvement over $2m + 1$ may be quite limited.

Even when there is no restriction on the type of threshold cell that can be used, it requires effort to do better than a linear number of neurons. Take, for example, the construction about to be given. It represents the best result so far in this direction: We begin with a Mealy machine M which has m states. There are no restrictions on the structure of M. We show that any such Mealy machine can be implemented using at most $O(m^{3/4})$ neurons. This construction uses neurons with two different functionalities: there are *state-neurons* and *transition neurons*. The simulating network will have $2k$ state-neurons where $k = \lceil m^{1/2} \rceil$. These state neurons fire in pairs, in accordance with the scheme laid out in Figure 5.

The state neurons are labeled $a_1, \ldots, a_k$; $b_1, \ldots, b_k$, the a-set representing rows of an implicit matrix, the b-set representing columns. A given state q is simulated when one of the pairs (a_i, b_j) fires. In Figure 5, for example, the state q corresponds to the simultaneous firing of a_2 and b_1. This is the easy part of the construction. Now we must make sure that whenever the Mealy machine is in a particular state the correct two state neurons fire. This we do by building, for each state neuron, a "black box" of *transition neurons* that make sure that the state neuron fires if and only if the Mealy Machine is in one of the states in the row or column of the state neuron. The threshold machine thus becomes periodic. By *periodic*, we mean that the threshold machine alternates between two phases: a *transition* phase, during which only transition

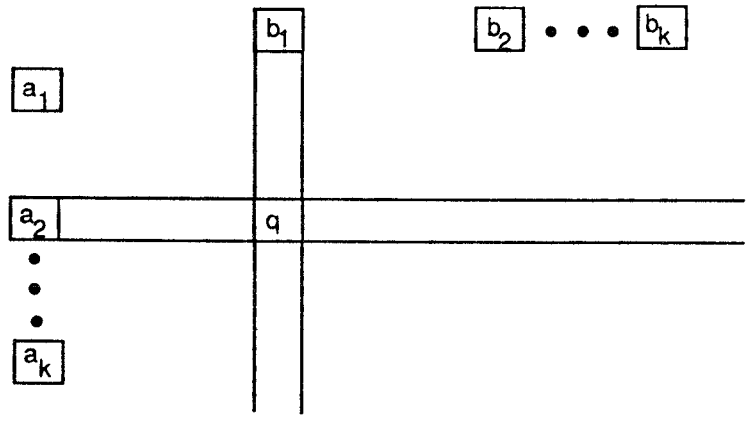

FIGURE 5

neurons fire, followed by a *state* phase, during which only state neurons fire. The state phase is exactly one time unit long, the transition phase will be seen to be exactly two time units long. To simplify comparison between the Mealy Machine and the Threshold Machine, we allow cells to fire at time t for t integer (at those times only state neurons can fire) and for $t = \frac{1}{3} \bmod 1$ or $t = \frac{2}{3} \bmod 1$ (at those times only transition neurons can fire). The transition neurons will be organized in two layers. We also introduce an output neuron that can fire at time t integer only, and outputs of the threshold machine lag behind outputs of the Mealy Machine by one time unit. We now describe the construction of the black boxes. In fact, there are two black boxes for each state neuron: One to be used if the previous input was zero, the other if the previous input was one. To simplify the language, we only describe the black box for a "row" state neuron (say state neuron a_x), to be used when the previous input was zero. Let $S(x, 0)$ be the set of states with the property that if at time $t - 1$ the Mealy machine is in one of those states and the input is zero, then at time t the Machine is in one of the states in the row of x. $S(x, 0)$ is a set of states and hence $k \times k$ incidence matrix, or a set of pairs (a_i, b_j). The black box must have the property that at time t neuron a_x fires if and only if at time $t - 1$ for some pair (a_i, b_j) in $S(x, 0)$ both a_i and b_j fired. The black box is a network that recognizes whether a pair in $S(x, 0)$ just fired and is accordingly called the recognition network for cell a_x, for input zero.

We now discuss some easily recognizable special incidence matrices. A set U of pairs (a_i, b_j) is called *simple* if there exist nonnegative vectors $(A_1, \ldots, A_k)$ and $(B_1, \ldots, B_k)$, and a positive threshold T, with the property that

$$(a_i, b_j) \in U \qquad \text{if and only if} \quad A_i + B_j \geq T. \tag{5.1}$$

As long as exactly one row state neuron a_i and exactly one column state neuron b_j fire, any *simple* set U can be recognized by a single neuron or cell C_U, with weights and threshold

$$w_{a_i,U} = A_i, \qquad w_{b_j,U} = B_j, \qquad \theta_U = T. \tag{5.2}$$

This cell recognizes set U by firing if and only if a state in U just occurred.

A set U of pairs (a_i, b_j), i.e., a $k \times k$ incidence matrix U, is called a "Northwest corner" matrix if there exist indices α_i,

$$k \geq \alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k \geq 0, \tag{5.3}$$

with

$$U_{i,j} = 1 \qquad \text{if and only if} \quad j \leq \alpha_i. \tag{5.4}$$

Equivalently, we could have required that there exist indices β_j,

$$k \geq \beta_1 \geq \beta_2 \geq \cdots \geq \beta_k \geq 0, \tag{5.5}$$

with

$$U_{i,j} = 1 \qquad \text{if and only if} \quad i \leq \beta_j \tag{5.6}$$

(with $\alpha_i = \max\{j\colon U_{i,j} = 1\}$ and $\beta_j = \max\{i\colon U_{i,j} = 1\}$). Similarly, we could define "NE", "SW", and "SE" corner matrices. Instead, we define a $k \times k$ incidence matrix U to be a "proto corner-matrix" if there exist permutations σ and τ of $\{1, 2, \ldots, k\}$ such that $U_{\sigma(i),\tau(j)}$ is a (NW) corner matrix.

It is easily verified that every $k \times k$ NW corner matrix is simple, and that the weights A_i, B_j and the threshold T all can be chosen in $\{0, 1, \ldots, k\}$ and such that

$$A_1 \geq A_2 \geq \cdots \geq A_k \geq 0, \qquad B_1 \geq B_2 \geq \cdots \geq B_k \geq 0. \tag{5.7}$$

Hence, also every proto corner matrix is simple. In fact, it is easily seen that an incidence matrix is simple if and only if it is a proto corner matrix. (Once $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$ are known, use permutations σ and τ such that (5.7) holds for the reordered matrix.) A set Q of pairs (a_i, b_j) is called *semi-simple* if it is the intersection of two *simple* sets U and V. Clearly, any *semi-simple* set Q can be recognized using three neurons or cells: Cells C_U and C_V as above, and a cell C_Q with weights and threshold

$$w_{U,Q} = w_{V,Q} = 1, \qquad \theta_Q = 2, \tag{5.8}$$

so that cell C_Q "ANDs" the cells C_U and C_V. Cell C_Q fires if and only if two time units ago a pair (a_i, b_j) in Q fired simultaneously. Now suppose we can write

$$S(x, 0) = \bigcup_{l=1}^{L} S_l, \tag{5.9}$$

where $S_1, S_2, \ldots, S_L$ (which need not be disjoint) are semi-simple in the sense above. In that case, we can recognize $S(x, 0)$ using $3L$ neurons. Of these, the $2L$ "V" and "U" neurons can fire at time $t = \frac{1}{3}$ mod 1 only, and the L neurons for $S_1, S_2, \ldots, S_L$ can fire at time $t = \frac{2}{3}$ mod 1 only. Between the S_l neurons and neuron a_x we have weights and threshold

$$w_{S_l,a_x} = 1, \qquad \theta_{a_x} = 1, \tag{5.10}$$

so that neuron a_x "ORs" the neurons $c_{S_1}, \ldots, c_{S_L}$. By this construction, neuron a_x fires at time t if and only if at time $t - 1$ a pair (a_i, b_j) in $S(x, 0)$ both fired. In order to complete the proof of Theorem 1.1, we need an upper bound for the number of semi-simple sets (or semi-simple incidence matrices)

needed to cover a set (or incidence matrix) S. As in Dewdney [1], we define a *line matrix* to be an incidence matrix that either is a subset of some row (all ones are in that row), or is a subset of some column (all ones are in that column) or is a *transversal*, i.e., no two or more ones in any row or column. It is easily seen that every line matrix is either simple or semi-simple: Any incidence matrix that is a subset of some row or column is simple, while every transversal is the intersection of two proto corner matrices and therefore is semi-simple. Dewdney [1] proved that any set S of pairs (a_i, b_j) can be covered by a set of L line matrices, with

$$L \leq \lceil \sqrt{|S|} \rceil . \tag{5.11}$$

The proof is based on "greedily" covering S by a sequence of line matrices $B_1, B_2, \ldots$ and applying Konig's theorem (Konig [6]) to the matrices B_t individually, see Dewdney [1] for details. Equation (5.11) gives an upper bound for the number of line matrices that may be needed to cover S, and therefore an upper bound for the number of semi-simple matrices that may be so needed. The remainder of the proof shows that any improvement in (5.11) produces an improvement in the final result. It seems likely that an improvement is indeed possible by using more general intersections of pairs of proto corner matrices than just line matrices.

Let $n_x = |S(x, 0)|$. Since each state q is in the set $S(x, 0)$ for exactly one a_x, we have

$$\sum_{x=1}^{k} n_x = m, \quad 0 \leq n_x \leq m \qquad \text{for all } x. \tag{5.12}$$

This shows that

$$\begin{aligned} \sum_{x=1}^{k} \lceil \sqrt{n_x} \rceil &\leq k + \sum_{x=1}^{k} \sqrt{n_x} \leq k + k\sqrt{\frac{m}{k}} \\ &= k + \sqrt{mk} = \lceil \sqrt{m} \rceil + \sqrt{m} \cdot \lceil \sqrt{m} \rceil^{1/2} - m^{3/4}. \end{aligned} \tag{5.13}$$

The total number of transition neurons in the black boxes for the row state neurons, for input zero, is at most of the order $3 \cdot m^{3/4}$. The same argument holds for transition neurons for column state neurons, and for input 1. We still must make sure that the "V" and "U" neurons in the black box to be used with input zero do not fire when the input is one (and vice versa). This is done in the same way as in the construction of the $2m + 1$ cell machine, earlier in this section. Finally, for the output neuron, we also build two black boxes; one for input zero, one for input one. Each of these black boxes contains at most $3k$ neurons. In the construction above, the a_i and b_j neurons can have Fan-out at least as high as $2k$ (probably more if the sets U and V for different transversals are not disjoint), and that the "U" and "V" cells can have Fan-in as high as $2k$. From Dewdney [1] or the argument preceding (5.7) above it is clear that the weights A_i, B_j take values in $\{0, 1, \ldots, k\}$ and that the thresholds T for the "U" and "V" cells take values in $\{1, 2, \ldots, k\}$. The construction above clearly, and not surprisingly, violates the conditions of Theorems 1.3, 4.2, 4.3, 4.4, and 4.5.

A final note concerns possible improvements to this construction. It would seem that one may build recognition networks (to drive each state neuron) that are far more efficient than the ones described here.

Can one, for example, "recognize" an arbitrary pattern of n 1's by using no more that $O(\log\ n)$ neurons in the recognition network? If so, the overall complexity for a network simulating an m-state Mealy machine would drop to $O(m^{1/2}\log\ m)$. As often happens in the closely related field of logic optimization (see, for example, Savage [11]), there may be a trade-off between depth and overall complexity: In other words, one may have to accept recognition subnetworks that have a nonconstant depth, which slowly increases as m gets large.

Another possibility involves a construction that uses a 3-dimensional state matrix governed by $3k$ neurons, where

$$k = \lceil \sqrt[3]{m} \rceil . \tag{5.14}$$

But, according to Theorem 1.2, we can hardly expect to do better than this since it implies a construction having

$$O(m^{1/3}\log\ m)$$

neurons, quite close to the lower bound of $c_2 \cdot (m\ \log\ m)^{1/3}$ neurons.

It is intriguing, nevertheless, that we apparently cannot usefully employ a construction that implies a dimension higher than three!

6. *Conclusion*

The results in this paper amount to a limitation on the power of neural nets. In spite of the current climate of optimism, and in spite of successes in limited categories of computation, one cannot look to neural networks as the parallel panacea of the future.

The limitations demonstrated here for the classic McCullough–Pitts networks apply with some force, obviously, to the case of connectionist mission. For example, a Hopfield-style network (see Hopfield and Tank, [5]), having as its goal a solution of the traveling salesman problem amounts to a finite automaton in which all states (hopefully) lead to one or just a few states.

We saw that while some m-state machines can be built using very few (in the order of log m) neurons, some need many more (namely, $K(m)$) neurons to be built. A naturally arising question is whether there are any interesting classes of finite automata that can be efficiently built as neural nets; namely, with a number of neurons that is very small compared with the size of the automata. Preliminary research by two of the authors of this paper indicates that while there are classes of finite state machines that can be efficiently simulated in this sense, all really interesting classes considered so far are too rich in structure and size. This research is continuing.

In addition, there is a large number of technical questions about the bounds presented in this paper. For example, we suspect (see Theorem 4.4) that a limitation on the Fan-in of neurons may lead to m-state machines that can be simulated only by a superlinear (in m) number of neurons.

ACKNOWLEDGMENT. We thank the two anonymous referees for a very careful reading of the paper and a large number of constructive comments.

REFERENCES

1. DEWDNEY, A. K. Threshold matrices and the state assignment problem for neural nets. In *Proceedings of the 8th SouthEastern Conference on Combinatorics, Graph Theory and Computing*. Louisiana State University, Baton Rouge, La., pp. 227–245.

2. HARDING, E. F. The number of partitions of a set of n points in k dimensions induced by hyperplanes. *Proc. Edinburgh Math. Soc. II, 15* (1966–67), 285–289.
3. HARDY, G. H., AND WRIGHT, E. M. *An Introduction to the Theory of Numbers*. Oxford University Press, Oxford, England, 1938.
4. HOPCROFT, J. E., AND ULLMAN, J. D. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass., 1979.
5. HOPFIELD, J. J., AND TANK, D. W. 'Neural' computation of decisions in optimization problems. *Biological Cybernetics 52* (1985), 141–152.
6. KÖNIG, D. Graphen und Matrizen. *Mat. Fiz. Lapok. 38* (1931), 116–119.
7. MCCULLOUCH, W. S., AND PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics, 5* (1943), 115–133.
8. MEALY, G. H. A method for synthesizing sequential circuits. *Bell Syst. Tech. J. 34* (1955), 1045–1079.
9. MINSKY, M. L. *Neural Nets and the brain model problem*. Ph.D dissertation, Princeton Univ., Princeton, N.J., 1954. (Available from University Microfilms, Ann Arbor, Mich.)
10. MINSKY, M. L. *Computation: Finite and Infinite Machines*. Prentice-Hall, New York, 1967.
11. RUMELHART, D. E., MCCLELLAND, J. L., AND THE PDP RESEARCH GROUP. *Parallel Distributed Processing,* vols. I and II. MIT Press, Cambridge, Mass., 1986.
12. SAVAGE, J. E. *The Complexity of Computing*, 2nd ed. Krieger, Malabar, Fla., 1987.

RECEIVED JULY 1988; REVISED JUNE AND NOVEMBER 1989, ACCEPTED MARCH 1990