Universal Learning Neurocomputers *

Victor Eliashberg †

Abstract

A universal learning neurocomputer is a brain–inspired information processing system that can be taught to perform, in principle, an arbitrary algorithm (universal in Turing's sense). Conventional computers and universal Turing machines are examples of universal programmable systems. However, the process of programming these systems does not match our intuitive notion of teaching and learning as applicable to the brain.

Some of the presently popular neural network models seem to qualify as brainlike learning systems. However, these models are not universal in Turing's sense, and can be taught to perform only very limited classes of functions. For example, the computing power of the perceptron–like feedforward networks does not exceed that of combinatorial machines. Accordingly, the computing power of such networks with a feedback can not be greater than the computing power of finite–state machines. In contrast, the human brain is "universal" and "learning" at the same time.

What is the simplest architecture of a universal learning neurocomputer? This paper is an attempt to address this basic issue.

1 INTRODUCTION

On Brain Hardware, Brain Software, and the Algorithms of Thinking

In the early days of cybernetics it was popular to invent and study different models of brain hardware. A number of interesting ideas were developed along this path [15, 17, 23, 27], and others. Researchers dedicated to this approach believed that understanding the brain hardware gives a clue to understanding the work of the brain and the creation of intelligent machines.

In the late sixties this basic belief was called into question by software–oriented AI. It was proposed that the algorithms of thinking underlying intelligent behavior are determined mainly by the symbolic structures stored in the brain's memory (brain software) rather than brain hardware. Accordingly, even a complete understanding of brain hardware may add very little (if anything at all) to the understanding of the structure of these algorithms.

Imagine an engineer trying to understand the work of a modern computer with sophisticated software by studying the wiring diagram of this computer and the transient responses in its gates and flip–flops. It was suggested by some AI methodologists that a researcher trying to understand the work of the brain by studying its neural circuitry is trapped in a similar pitfall.

This metaphor makes a strong point. To understand the work of a programmable computing system it is not sufficient to know how the information is stored in its memory (the states of flip–flops, the charges of capacitors, the conductivities of fuses or synapses, etc.) It is more important to know what is stored in this memory.

The metaphor has a subtle but serious flaw as well. It implies that the general relationship between brain software and brain hardware is somewhat similar to the corresponding general relationship in a conventional (von Neumann) computer. This implied analogy is highly misleading. The basic neurobiological phenomena in brain hardware are more then just im-

*Proceedings of the Fourth Annual Parallel Processing Symposium, Fullerton, CA, April 4-6, 1990, pp. 181-191

†Visit the web site www.brain0.com for information

plementations of minor steps of the algorithms of thinking per se. The mathematical expressions describing these phenomena constitute large and critically important integral parts of these algorithms. It can be very difficult to understand the meaning and the structure of the corresponding biophysical computations without a brain hardware heuristics.

Software oriented AI came up with a number of interesting ideas about the structure of the algorithms of thinking per se [9]. However, as a result of this research not much became clearer about the work of the biological brain. Brain hardware has avenged AI for neglect.

In the late seventies and the eighties the basic methodology of the software–oriented AI was challenged by brain hardware enthusiasts [10, 12, 14, 11, 22, 4, 5]. It was emphasized that most of the proposed AI algorithms cannot be accepted as adequate models of the real algorithm of thinking, because they are inconsistent with the known data about the brain hardware (the latter is too slow, not precise enough, highly parallel, built from analog rather than logical elements, etc.)

The pendulum of fashion began to swing to another extreme. The whole idea of symbolic brain software (brain code) was called into question. The approach known as neoconnectionism or PDP (parallel distributed processing) set out to show that cognitive phenomena can be explained without the use of the basic ideas associated with symbolic information processing paradigm.

In a connectionist system events are represented by subsets of active nodes, and knowledge is stored in the sets (matrices) of variable weights of connections between nodes. A node is a simple (often analog, neuron–like) processing element. Most of the processing is accomplished in a massively parallel way by activation spreading through connections. Importantly, there is no telecommunication among nodes via symbolic messages (one of the main attributes of symbolic information processing paradigm). Among the most significant original contributions of the neoconnectionism is the idea that neural networks perform optimization (constraint satisfaction) tasks via a relaxation process [14, 13].

During the last few years a number of interesting connectionist models have been developed and experimented with. However, none of these models were able to provide a satisfactory general explanation of how nonsymbolic brain hardware can produce nontrivial symbolic cognitive phenomena (such as language, logical reasoning, etc.) without symbolic brain software. See [20] for a discussion of the problem of "connections and symbols".

In writing this paper I want to promote the natural idea that the truth must lie somewhere between these two extremes. I argue that it is wrong, in principle, to ask how a nonsymbolic brain hardware can do without symbolic brain software and vice versa. Instead, one should ask how a nontrivial symbolic software can be formed in the process of learning, and how such software can efficiently interact with massively parallel nonsymbolic hardware.

It is this nontrivial interaction that has a clue to the problem of a universal learning machine.

2 AN ANALOG NEURAL NETWORK AS A SYMBOLIC MACHINE

2.1 McCulloch–Pitts Problem

Talking about universal neurocomputers, it is appropriate to start with the classical McCulloch–Pitts [15] problem:

How can an arbitrary finite–state machine be built from neuron–like elements ?

For the purpose of this paper we need to extend this problem in three important ways:

1. We are interested in a network universal with respect to the whole class of finite–state machines.

2. We want to build this network from analog rather than logical neurons.

3. We need to find a learning algorithm that would allow us to teach this network to simulate any finite–state machine from above class.

A solution to this problem can be obtained in two steps:

1. Describe a neural network capable of simulating an arbitrary combinatorial machine.

2. By introducing a delayed feedback, transform this network into a system capable of simulating an arbitrary finite–state machine

2.2 The Concept of a Combinatorial Machine

A *combinatorial machine* is a system $M = (\mathbf{X}, \mathbf{Y}, F)$,

where

- $\mathbf{X}$ and $\mathbf{Y}$ are finite nonempty sets, called the input and the output alphabets of M, respectively.
- $F : \mathbf{X} \to \mathbf{Y}$ is a function from $\mathbf{X}$ into $\mathbf{Y}$.

The machine works as follows:

If a symbol $x(\nu)$ from $\mathbf{X}$ is present in the input of M at cycle ν, then the output $y(\nu)$ of M during this cycle is equal to $y(\nu) = F(x(\nu))$.

For our purpose, it is convenient to represent function F as a *table of associations.*

EXAMPLE

Let $\mathbf{X}$={A,B,C},$\mathbf{Y}$={0,1}, F={(A,0),(B,1),(C,0)}

Some of the tables of associations corresponding to this machine are:

A	B	C
0	1	0

B	C	A
1	0	0

C	A	B
0	0	1

Let us think of a table of associations as a piece of software written in memory. Let $\mathbf{G}$ be the set of states of this memory. A *machine universal with respect to the class of combinatorial machines* is a system $MU = (\mathbf{X}, \mathbf{Y}, \mathbf{G}, FU)$, where $FU : \mathbf{X} \times \mathbf{G} \to \mathbf{Y}$ is a function which takes input symbol $x(\nu)$, looks for the entry of this symbol in the table of associations and retrieves the corresponding output symbol $y(\nu)$. FU will be referred to as the *decision making* procedure of machine MU.

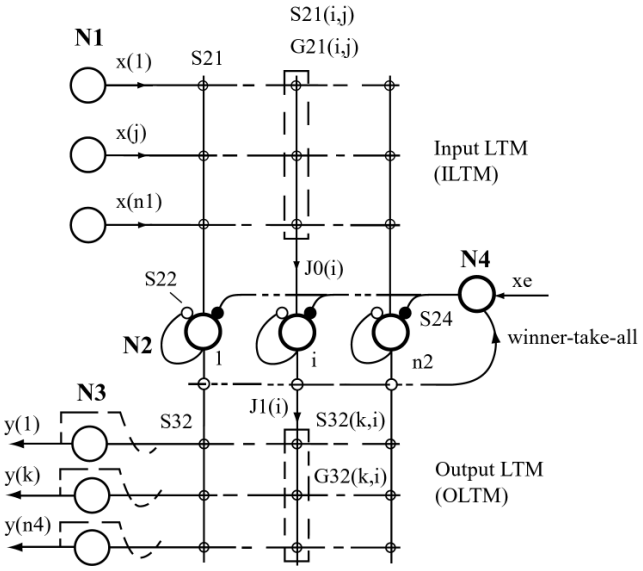


Figure 1: Analog neural network universal with respect to the class of combinatorial machines

2.3 A neural Network Universal with Respect to the Class of Combinatorial Machines

2.3.1 Neuromorphological Model

Consider the neural network schematically shown in Figure 1. Large circles with incoming and outgoing lines represent neurones with their dendrites and axons, respectively. Small white and black circles represent excitatory and inhibitory synapses, respectively. The network has three layers of neurons: input neurons N1, intermediate neurons N2, and output neurons N3. Neurons N2 have a global inhibitory feedback via neuron N4 and local excitatory feedbacks. It can be shown that in such a network neurons N2 can compete via reciprocal inhibition in the *winner–take–all* fashion [3, 4, 6]. A similar effect can be obtained in a network with lateral inhibitory feedbacks.

Notation. $Nj(i)$ is the $i-th$ neuron from the $j-th$ set. $Sji(m,n)$ is the synapse between the axon of neuron $Ni(n)$ and the dendrite or soma (body) of neuron $Nj(m)$. The $*$ substituted for

an index denotes the whole set of elements corresponding to this index. For example, $S21(i,*) = (S21(i,1),...S21(i,n1))$ is the set of input synapses of neuron $N2(i)$,
$S32(*,i) = (S32(1,i),...S32(n3,i))$ is the set of output synapses of neuron $N2(i)$, etc.

The long–term memory that stores table of associations is implemented in synaptic matrices S21 and S32. The vector of gains (weights) of synapses $S21(i,*)$ is treated in this section as an encoded symbol stored in the $i-th$ location of the *input long–term memory* (ILTM) of the network. The vector of gains of synapses $S32(*,i)$ is treated as an encoded symbol stored in the $i-th$ location of the *output long–term memory* (OLTM) of the network.

2.3.2 Functional Model

In what follows I present a simplified "phenomenological" description of the work of the network of Figure 1. A detailed study of different aspects of the neurodynamics of this network can be found in [4, 6].

DECODING. Input vector $x(*) = x(1),...x(n1)$ (the output vector of neurons N1) reaches all neurons N2. The net postsynaptic current of synapses $S21(i,*)$, denoted as $J0(i)$, is equal to the dot product of $x(*)$ and the vector of gains $G21(i,*)$ of these synapses (the vector stored in the $i-th$ location of ILTM).

$$J0(i) = \sum_{j=1}^{n1} G21(i,j) * x(j) = FS(G21(i,*), x(*)) \tag{1}$$

where $FS : \mathbf{X} \times \mathbf{X} \rightarrow \mathbf{R}$ ($\mathbf{R}$ is the set of real numbers) is a *similarity function* calculating a similarity between any two vectors from the set of input vectors $\mathbf{X}$. In this case FS is defined as the dot product of two vectors from $\mathbf{X}$.

I say that a pair $(FS, \mathbf{X})$ satisfies the *correct decoding condition* if

$$\text{for all} \quad x, z \in \mathbf{X} \ (x \neq z \rightarrow FS(x,x) > FS(x,z)) \tag{2}$$

That is, each vector from $\mathbf{X}$ is "more similar" (closer) to itself than to any other vector z from $\mathbf{X}$ not equal to x. For example, any set of normalized real vectors satisfies the correct decoding condition (2) with the similarity function FS defined as the dot product.

CHOICE. It can be shown that after the end of a transient response, layer N2 performs a random equally probable choice of a single location (call it $i0$) from the set of locations with the maximum value of $J0(i)$, assuming this value exceeds the inhibitory input xe. It is assumed that the threshold of N2(i) is equal to zero. In phenomenological terms, the corresponding operation of the random winner–take–all choice can be described as follows:

$$i0 :\in \text{MAXSET} = \{i : J0(i) = \max(J0(1),..J0(n2)) > xe\} \tag{3}$$

$$\text{if} \quad i = i0 \quad \text{then} \quad J1(i) = 1 \quad \text{else} \quad J1(i) = 0 \tag{4}$$

where PASCAL–like notation $i :\in M$ is used to denote a random equally probable choice of an element i from the set M. $J1(i)$ is the output of neuron $N2(i)$.

ENCODING. According to (3) and (4) $J1(i)$ is greater than zero for only one neuron $N2(i)$ with $i = i0$. Therefore, the output vector of the network is equal to the vector of gains of synapses $S32(*,i)$. This vector encodes the symbol stored in the $i0-th$ location of OLTM.

$$y(*) = G32(*, i0) \tag{5}$$

It is easy to see that Expressions (1)–(5) describe a decision making procedure sufficient to implement a machine universal with respect to the class of combinatorial machines.

LEARNING. Let us assume that the teacher can clamp the output centers N3 and force any desired output of these centers (supervised learning). Let us also assume that the sequences of input and output vectors of the described network are simply tape–recorded in its ILTM and OLTM (rote learning). In this paper, I am not concerned with a possible neural implementation of such an algorithm. It can be done in many different ways. A formal description of this very simple learning procedure is, presented later in

Section 4.3. I argue that this procedure is, in fact, not too bad as a zero–approximation model.

2.3.3 Periodic Inhibition

Layer N2 has a hysteresis. Once the winner $N2(i0)$ occupies the dominating position, it can hold this position even after $J0(*)$ has changed in such a way that $i0$ no longer belongs to MAXSET. To make the analog network of Figure 1 work as a discrete–time machine we can introduce a periodic global inhibition via input xe. Such a periodic inhibition with the period $T \gg \tau$, where τ is the time constant of $N2(i)$, restores equal initial conditions at the beginning of each cycle.

2.4 Simulation of Deterministic Finite–State Machines

Now that we have a neural network that can simulate an arbitrary combinatorial machine, the solution of the problem formulated in section 2.1 is trivial.

A *finite–state machine* is a system

$M = (\mathbf{X}, \mathbf{Y}, \mathbf{Q}, FQ, FY)$, where

- $\mathbf{X}$ and $\mathbf{Y}$ are finite nonempty sets of input and output symbols (input and output alphabets), respectively.
- $\mathbf{Q}$ is a finite set of internal states.
- $FQ : \mathbf{X} \times \mathbf{Q} \to \mathbf{Q}$ is the next–state function.
- $FY : \mathbf{X} \times \mathbf{Q} \to \mathbf{Y}$ is the output function. (This model is often called a Mealy machine).

Let $x(\nu), q(\nu), y(\nu)$ be respectively, the input, the state, and the output of M at the $\nu - th$ cycle. The work of M is described as follows:

$$q(\nu + 1) = FQ(x(\nu), q(\nu)) \quad (1)$$

$$y(\nu) = FY(x(\nu), q(\nu)) \quad (2)$$

Note. In this paper expressions are numbered locally within each section. When reference is made to expressions from other sections, the number of the section is added in front of the number of expression.

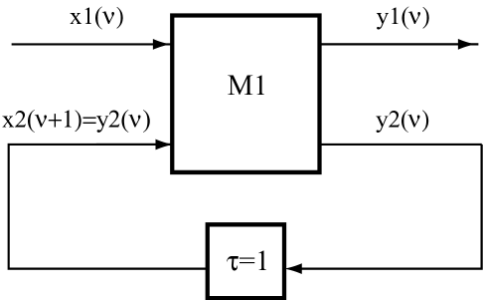


Figure 2: Transforming a combinatorial machine into a finite-state machine

For example, 2.3.2(3) is expression (3) from section 2.3.2.

It is easy to see that a finite–state machine is just a combinatorial machine with a delayed feedback. Let $M1 = (\mathbf{X1}, \mathbf{Y1}, F1)$ be a combinatorial machine with the input alphabet $\mathbf{X1} = \mathbf{X} \times \mathbf{Q}$ and the output alphabet $\mathbf{Y1} = \mathbf{Q} \times \mathbf{Y}$. Let us introduce the delayed feedback $x2(\nu + 1) = y2(\nu)$ and treat $x2(\nu)$ as the state variable $q(\nu)$. Let us treat $x1(\nu)$ and $y1\nu)$ as the input and output variables, respectively, and let F1 have projections FQ and FY. The resulting new machine shown in Figure 2 is described by expressions(1) and (2).

2.5 Simulation of Probabilistic Machines

Most of the tables of associations that can be written in ILTM and OLTM of Model 2.3 (the model described in section 2.3) correspond to probabilistic machines. For example, the table

A	B	A	C	A	C	A	C
0	0	1	0	1	1	1	1

Corresponds to the probabilistic combinatorial machine with conditional probabilities $P(1|A) = 3/4$, $P(0|A) = 1/4$, $P(1|B) = 0$, $P(0|B) = 1$, $P(1|C) = 2/3$, and $P(0|C) = 1/3$. The evaluation of the numbers of tables of associations corresponding to deterministic and probabilistic machines can be found in [4]. It is easy to prove that Model 2.3

can simulate an arbitrary probabilistic combinatorial machine with rational probabilities, provided $n2$ can be made arbitrarily large.

2.6 Generalization by Similarity

Model 2.3 can simulate an arbitrary combinatorial machine. However, this model is a little more than just a symbolic machine. If an input vector is not present in ILTM the model generalizes by similarity. In this case MAXSET is the set of locations containing vectors most similar to the input vector. The output vector is retrieved from a location of OLTM randomly selected from MAXSET.

Model 2.3 gives a simple example of a neural implementation of an associative memory. This implementation can be called "local", as opposed to distributed implementation represented by such models as, say, Hopfield [14] network. I argue that the idea of distributed associative memory "on local minima" represented by latter model is, essentially, inconsistent with the notion of the human brain as a universal learning system.

2.7 Is Model 2.3 a Connectionist System?

One can say that Model 2.3 is a connectionist system because its LTM is implemented on connections. This is not, however, what is meant by the term "connectionism" in its modern interpretation. The real issue is how Model 2.3 uses its memory, rather than how this memory is implemented. Model 2.3 uses its memory to store symbolic software. This is exactly what modern connectionism wants to avoid. The fact that the memory of Model 2.3 is represented by synaptic weights is irrelevant to the issue of "connections and symbols". For example, such traditional logical devices as bipolar PROMs and PALs use memory on connections formed by fuses.

History. Model 2.3 illustrates a simple but nontrivial integration of the three important brain hardware ideas:

1. A neuron as an elementary learning decoder (recognizer).
2. A neuron as an elementary learning encoder.
3. Reciprocal inhibition and fluctuation phenomena as the mechanism of choice.

All these ideas were known in the sixties [15, 23, 27, 25, 21]. A specific integration of these ideas represented by Model 2.3 was described in [3].

That is, neither historically, nor conceptually does the network of Figure 1 owe anything to neoconnectionism.

3 THE LEVELS OF COMPUTING POWER AND THE TYPES OF BEHAVIOR

Why is the level of computing power important?

In system design, the role of the basic constrains associated with the levels of computing power can be compared with the role of energy conservation law in mechanical design. A mechanical engineer has to reject a model of the Perpetual Motion Machine, because it violates the energy conservation law. Similarly, a system engineer has to reject a model of the brain if this model has the level of computing power lower than that of the brain.

3.1 Symbolic Read/Write Memory and the Levels of Computing Power

What is badly missing in Model 2.3 is a symbolic read/write memory. Without such a memory this model cannot have the level of computing power exceeding that of combinatorial machines. Accordingly, with a feedback it cannot exceed the computing power of finite–state machines. In Chomsky's classification [2], this level of computing power corresponds to the formal language of type 3. Such formal languages are just the sets of strings generated by *finite–state grammars*.

As is known, the next level of computing power corresponds to the languages of type 2 (the sets of strings generated by *context–free grammars*). This level requires a combination of a finite–state machine and a read/write memory. However, the finite–state machine is allowed to use the memory only in a last–in–first–out fashion (stack). Such systems, consisting of a finite–state machine coupled with a stack, are called *push–down automata*.

Removing all limitations on the type of memory access gives the system, consisting of a finite–state machine and memory, the highest possible level of computing power. This level corresponds to the class of Turing machines or formal language of type 0.

Note. I have intentionally skipped type 1 (context–sensitive grammars). I believe, that unlike types 3, 2, and 0 this type of behavior does not have a psychologically useful interpretation.

3.2 A Psychological Interpretation of the Basic Types of Behavior

Because of the critical importance of the notion of the levels of computing power, in what follows I present a useful psychological interpretation of the behavior of type 3, 2, and 0. I believe this interpretation may help some neural network researchers, not familiar with Chomsky's theory [2], to get a clear intuitive understanding of the fundamental nature of the corresponding constrains.

TYPE 3. It is convenient to divide this type into two subtypes:

a) $S \rightarrow R$ type of behavior: If you see (hear, etc.) a stimulus S, produce a response R associated with S. This is similar to the behavior of combinatorial machines. There is no internal feedback.

b) $SQ \rightarrow Q'R$ type of behavior: If you see S and keep in mind Q, respond with R and change your mind to Q'. In this case there is an internal feedback. However, the important thing is that Q must be kept in mind for only one cycle. Once the state of mind is changed the previous state is completely forgotten.

TYPE 2. The same as type 3 but you are allowed to have a notebook. This notebook can be used only in the following limited way:

Suppose you start some activity A1 of type 3, and at some point want to interrupt this activity and perform some other activity, A2, of type 3. After finishing A2 you want to continue A1. A system of type 3 cannot do this, since, once A2 is started, the system forgets that it must eventually return to A1.

With the notebook you can easily solve this problem. When interrupting A1, make a note in the notebook about the place where activity A1 was interrupted (return address of A1). While doing A2 you may want to interrupt A2 and start A3, etc. Now you must also save the return address of A2, and so on. The return address of the most recently interrupted activity is the last address written in notebook. This last written address is needed first to return to the most resently interrupted activity. In the general case to return to any number of interrupted activities it is sufficient to use memory in the last–in–first–out fashion (stack).

It is easy to notice that when we talk we often need to make a note in an "internal stack" to return to a temporarily interrupted line. This is an example of a behavior of type 2 performed by the brain without any external memory aid. Most linguists agree that at least this level of computing power is required of the model of the human brain to explain our ability to generate grammatically correct sentences.

TYPE 0. Remove limitations on the use of the external memory aid, and you can perform, in principle, any possible computation. A formalization of this type of behavior lead Turing [26] to the idea of his famous machine.

3.3 What Level of Computing Power does a Model of the Human Brain Need?

The following observation shows that an adequate model of human brain must have the highest level of computing power (type 0).

A person performing calculations with the use of

an external memory device (a piece of paper, etc.) learns to perform similar mental calculations. It is easy to notice that this effect is achieved because the person learns to imagine (mentally simulate) an external symbolic read/write memory.

A chess player learns to move pieces on an imaginary chess board. An abacus user learns to use an imaginary abacus [1], etc. Ignoring some severe, but theoretically unimportant limitations on the size of the working space available via this mechanism of mental imagery, such observations mean that the human brain must be treated by a cognitive modeler as a system with the highest level of computing power (type 0).

One can argue that a Turing machine with a finite tape is, theoretically, a finite–state machine (type 3). This popular argument misses the main point. It is the way a system uses its read/write memory rather than the size of this memory that determines the type of the system's behavior and the level of its computing power. An attempt to describe the work of a Turing machine with a finite tape in terms of a less powerful computational model, e.g., a finite–state machine, immediately leads to problem of a combinatorial explosion of the size of such representation (the tape with n squares, each square capable of storing m symbols, has m^n states).

3.4 Does the Brain have a Conventional Symbolic Memory Buffer?

A neural modeler can simply say: let us implement a read/write memory buffer as a neural network. This can be done in many different ways. Such a solution, however, would leave open most of the problems associated with the brain's read/write short–term memory.

- *Why does information "decay" in STM?*
- *What is the meaning of the "memory span"?*[16]
- *Why do we "see more than can be remembered"?*[24]
- *What is mental imagery?*
- *How do we learn to imagine an external read/write memory aid?*

One should also keep in mind the problem of mental set and context [28]. How does the brain change its attitude depending on context? How does it solve the problem of combinatorial explosion associated with a combinatorial number of different contexts?

The simple idea described in the next section explains how an effect of symbolic read/write STM can be produced without a conventional memory buffer. This idea was previously referred to in [3] as the concept of an associative automation and more recently, as the concept of an E–machine [4, 5, 7]. Unfortunately, I found it very difficult to communicate this "symbolic/nonsymbolic" idea to either "symbolic" AI researchers or "nonsymbolic" neural modelers.

4 "RESIDUAL EXCITATION" AS THE MECHANISM OF STM AND MENTAL SET

4.1 A Seemingly Inefficient Idea

Suppose we have a tape with n squares, each square capable of storing any of m symbols from a set S. How can we simulate such a tape? The obvious way is to have a random–access memory (RAM) with n locations and assign one location to each square.

Another more sophisticated and outwardly less efficient possibility can be described as follows (see Figure 3):

Let us have a read–only memory (ROM) with n^*m locations and assign m locations to each square of the tape. Let us store all symbols from S in each subset of m locations assigned to the squares. In addition to this symbolic ROM, let us have a nonsymbolic memory to store a "characteristic function" representing the levels of "residual excitation" assigned to different locations of ROM. Let $G = G(1), ...G(n^*m)$ denote the state of the symbolic ROM, where $G(i)$ is the symbol written in the i-th location of this ROM. Let $E = E(1), ...E(m^*n)$ denote the state of the above

nonsymbolic memory (call it an E–state), where $E(i)$ is the level of residual excitation assigned to the i-th location of ROM.

It is easy to see that pair (G,E) with a fixed G and variable E can represent all possible states of the above tape. This result is illustrated in Figure 3a,b. Note that pair (G,E) also represents some "nondeterministic" states which cannot be interpreted as the states of a deterministic tape (Figure 3c).

To write a symbol into a square of such a simulated tape means to increase the level of residual excitation of the location of ROM corresponding to the selected square with the above symbol. How can this general idea be applied to neural networks? Figure 4 gives one of the simplest answers to this question.

The network shown in Figure 4 is similar to that of Figure 1 except it has two layers of intermediate neurons N2 and N3. N2 serve as decoders, whereas N3 serve as encoders. The nonsymbolic E–state is associated with synapses S32 between N2 and N3. It is sufficient to postulate that there is a temporary facilitation in these synapses to get the desired effect of a read/write symbolic memory [4, 7].

In the next sections I show how the network of

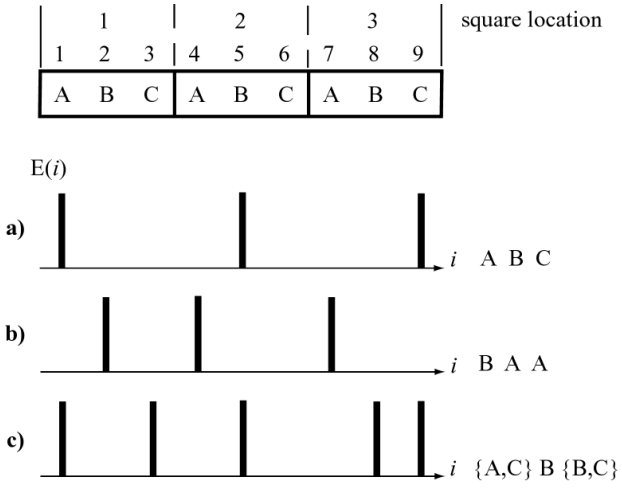


Figure 3: Simulating a read/write tape by combining a symbolic ROM with a nonsymbolic STM

Figure 4 provides a simplified but nontrivial explanation of our ability to learn to imagine an external read/write memory aid. This in turn gives a clue to understanding our ability to learn to perform, in principle, arbitrary mental calculations.

4.2 A Simple Mental Imagery Experiment

Consider a tape with three squares {1,2,3}. (The number three has no special meaning and is selected to simplify the experiment). Assume there is a set of symbols {A,B,C} which you can write in these squares. (The number of symbols does not have to

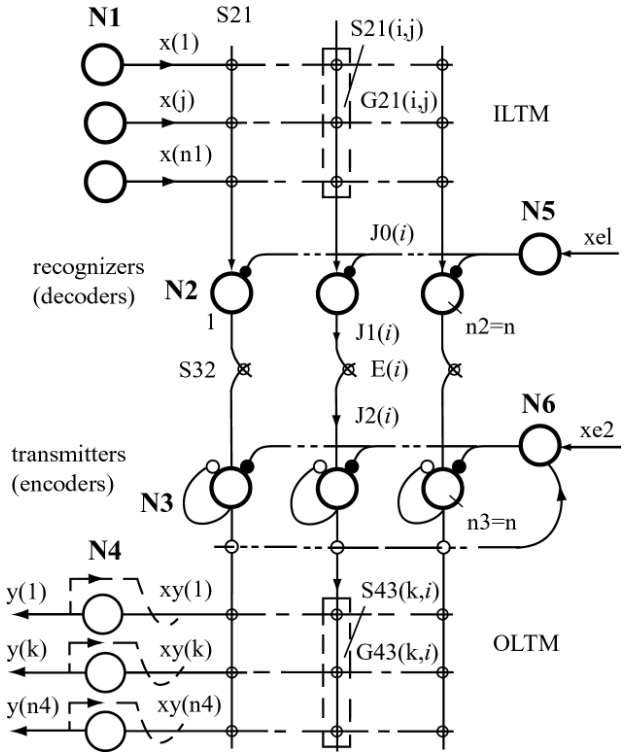


Figure 4: Temporary facilitation in synapses S32 in combination with a one time programmable ILTM and OLTM is sufficient to simulate a symbolic read/write memory

be the same as the number of locations). Close your eyes and imagine an empty tape. Look at square 1 of this imaginary tape and mentally write symbol A into this square. It helps if you actually move your hand as if you are writing. If you are an experienced computer user imagine that you are typing on the screen. Move your hand as if you are actually using a keyboard. Symbol A appears in square 1 of the imaginary tape.

Write different symbols in square 1, 2, and 3. After each round of writing, scan the imaginary tape. If your mental imagery is the same as mine, you always see the last symbol written in a square. To make this effect more distinct imagine that squares are sufficiently separated, so you can see only one square at a time.

If you can perform this mental experiment, you can mentally simulate a symbolic read/write memory with three locations $i = 1, 2, 3$, each location capable of holding any of three symbols A,B,C.

How can the network of Figure 4 explain this effect?

4.3 Functional Model

Let us use the following simple functional model of the network Figure 4.

DECODING. Let us assume that the set of input centers N1 is divided into p subsets (modalities). Let xj be the input symbol represented by an input vector (subvector) of the j-th modality. For the sake of simplicity, let us assume that symbols are encoded as normalized orthogonal vectors. Let $GX(i)$ denote the vector stored in the i-th location of ILTM ($GX(i)$ is the same as $G21(i, *)$), and let $GXj(i)$ denote the j-th subvector of this vector. Instead of 2.3.2(1) we have:

$$J0(i) = \sum_{j=1}^{p} \lceil x_j = GX_j \rceil \tag{1}$$

where $\lceil P \rceil$=1 if predicate P is true. Otherwise, $\lceil P \rceil$=0

$$\begin{array}{rl} \text{if} & J0(i) > xe1 \\ \text{then} & J1(i) = J0(i) - xe1 \text{ else } \quad J1(i) = 0 \end{array} \tag{2}$$

MODULATION. The temporary facilitation in synapse $S32(i, i)$ is described by a single E–state:

$$J2(i) = J1(i) * (1 + \beta * E(i)) \tag{3}$$

CHOICE

$$\begin{array}{l} i0 :\in \text{MAXSET} = \\ \qquad \{i : J2(i) = \max(J2(1), ..J2(n)) > xe2\} \end{array} \tag{4}$$

ENCODING

$$y = GY(i0) \tag{5}$$

NEXT G–STATE PROCEDURE (learning). As in section 2.3 let us assume that the sequences of input and output symbols are just tape–recorded in ILTM and OLTM. To formally describe this mechanism, without going into details of its possible neural implementation, we need two additional "phenomenological" variables: write pointer wp and write enable wen. We also need to assume that there is an input xy allowing one to write symbols into OLTM.

$$\begin{array}{l} \text{if } \ wen = 1 \ \text{ then } \textbf{begin } \ GX'(wp) = x; \\ \qquad GY'(wp) = xy; \quad wp' = wp + 1; \ \textbf{end} \end{array} \tag{6}$$

where the single quote ($'$) denotes the value of a state variable at the next cycle, that is, wp' is the same as $wp(\nu + 1)$, etc., where ν is discrete time.

Note. I have argued [4, 7] that this very simple learning procedure (rote learning) is not too bad as a "zero–approximation" learning algorithm for a model of the brain. The great advantage of this procedure is that it is *universal in the broadest possible sense*.

Such popular nonsymbolic learning algorithms as backpropagation, simulated annealing, etc. are not universal and not compatible with the concept of a universal learning machine. At the same time, the more sophisticated symbolic learning procedures proposed by traditional AI are too complex to be implemented in brain hardware.

The concept of "tape–recording" can be dramatically enhanced by introducing hash–recording, hierarchical learning, selection by novelty, statistical selection, attention, and other mechanisms implementable in brain hardware and compatible with the general idea of a universal learning machine.

I believe, it is important to avoid the pitfall of overestimating the cognitive possibilities of different sophisticated data storage procedures. Our ability to learn depends on the contents of our LTM (we learn how to learn, and even how to learn how to learn). Accordingly, no matter how smart and sophisticated, a hardware determined data storage procedure (as well as any other fixed learning algorithm) can never be smart enough. Therefore, it better be dumb but universal, so any kind of software can be put into LTM.

It is software, created in the course of learning, that must be able to decide what to learn.

NEXT E–STATE PROCEDURE. Let us assume that the process of facilitation in synapse $S32(i,i)$ is described by the following simple mechanism. More sophisticated models would work as well. The psychological effect of symbolic read/write memory discussed in the next section is not sensitive to many details of this neurobiological mechanism.

$$\begin{array}{l} \text{if} \quad J1(i) > E(i) \\ \text{then} \quad E'(i) = J1(i) \text{ else} \quad E'(i) = \gamma * E(i) \end{array} \tag{7}$$

where $1 > \gamma > 0$

There is a rapid "charging" of $E(i)$ if $J1(i) > E(i)$. Left alone, $E(i)$ "discharges" with the time constant $\tau = 1/(1-\gamma)$.

Note. Expressions (1)-(7) describe a simple example of a system which is called a *primitive E– machine* [4] or an *associative field* [3]. In the general case, the next E–state procedure of a primitive E–machine can be considerably more complex than the one described above. Recently I discussed the possibility of connecting the dynamics of the macroscopic E–states with the statistical dynamics of the conformations of the protein molecules in neural membranes [8]. This possibility would justify quite complex assumptions about the dynamic properties of E–states.

4.4 An Effect of a Symbolic Read/Write STM and Mental Imagery

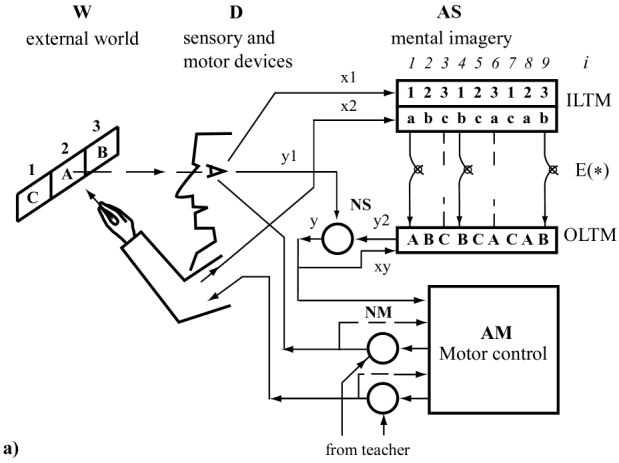
How can Model 4.3 explain the effect of STM and mental imagery described in section 4.2? To get a simplified but nontrivial answer to this question consider the system (ROBOT, EXTERNAL WORLD) schematically shown in Figure 5, where W is the external world in the form of a tape with three squares, D is the set of the robot's sensory and motor devices, and (NS,NM,AS,AM) is the robot's brain. The brain consists of sensory centers NS, motor centers NM, associative learning system AS and associative learning system AM. System AS is responsible for the phenomena of STM and mental imagery, whereas system AM is responsible for motor control. Model 4.3 with p=2 in 4.3(1) is used as system AS. The internal structure of AM is not important for the purpose of this section. It can also be arranged as Model 4.3. The input and output variables of AS shown in Figure 5 have the following meaning:

- $x1 \in \{1,2,3\}$ is an encoded motor input representing positions of the eye corresponding to different squares.
- $x2 \in \{a,b,c\}$ is an encoded motor input representing the actions of the hand when writing (or typing) symbols A,B,C on the tape.
- $y1 \in \{A,B,C\}$ is the output of the eye representing a symbol read from the external tape.
- $y2 \in \{A,B,C\}$ is the output of AS representing a symbol read from the imaginary tape.

Let us assume that NS works in such a way that its output is $y = y1$, if the eye is open, and $y = y2$, if the eye is closed (xy is the output needed to write visual symbols into the OLTM of AS).

System AS forms Motor→Sensory (M→S) associations since its goal is to simulate the work of the external system (W,D) as it is seen by system AM. In a more complex case AS may form MS→S associations. System AM forms S→M or, in a more complex case, SM→M associations.

To simplify understanding, let us divide the experiment with the robot into two stages: learning and examination.



a)

	i	1	2	3	4	5	6	7	8	9
	J1(i)	2.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00	0.00
ν=0 x1=1 x2=a	E(i)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	J2(i)	2.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00	0.00
	J1(i)	1.00	1.00	0.00	2.00	0.00	0.00	1.00	0.00	1.00
ν=1 x1=1 x2=b	E(i)	2.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00	0.00
	J2(i)	1.40	1.00	0.00	2.40	0.00	0.00	1.20	0.00	1.00
	J1(i)	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ν=2 x1=1 x2=0	E(i)	1.80	1.00	0.00	2.00	0.00	0.90	1.00	0.90	1.00
	J2(i)	1.36	0.00	0.00	1.40	0.00	0.00	1.20	0.00	0.00
	J1(i)	1.00	0.00	1.00	1.00	1.00	0.00	2.00	0.00	0.00
ν=3 x1=1 x2=c	E(i)	1.62	0.90	0.00	1.80	0.00	0.81	1.00	0.81	0.90
	J2(i)	1.32	0.00	1.00	1.36	1.00	0.00	2.40	0.00	0.00
	J1(i)	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
ν=4 x1=1 x2=0	E(i)	1.46	0.81	1.00	1.62	1.00	0.73	2.00	0.73	0.81
	J2(i)	1.29	0.00	0.00	1.32	0.00	0.00	1.40	0.00	0.00

b)

Figure 5: An explanation of the effect of mental imagery discussed in section 4.2.
a) Motor symbols representing the eye position and the actions of the hand retrieve visual information associated with these symbols. System AS learns to simulate the external world, W, as it appears to system AM via sensory and motor devices, D.
b) The values of variables $J1(i)$, $E(i)$ and $J2(i)$ during the experiment.

At the stage of learning the robot writes symbols A,B,C, then B,C,A, and then C,A,B in squares 1, 2 and 3, respectively. Each symbol has to be written at least once in each square. At this stage writing is enabled $wen = 1$, so at the end of learning the table of associations shown in Figure 5a is stored in ILTM and OLTM of AS.

At the stage of examination, writing in LTM is disabled $wen = 0$. (It could be enabled, but an analysis of the experiment would become a little more complex). Let $xe1 = xe2 = 0$, and let, for the sake of concreteness, β=.2, γ=.9, that is $\tau = 1/(1-\gamma) = 10$. Let at $\nu = 0$, $E(i) = 0$ for $i = 1, ..9$. The robot closes its eye and performs the following actions:

- $\nu = 0$: The robot writes symbol A in square 1. $x1 = 1$, $x2 = a$, $i0 = 1$, $y = A$. System AM sees symbol A in square 1 of imaginary tape. The values of $J1(i)$, $E(i)$ and $J2(i)$ explaining this result are presented in the table of Figure 5b.
- $\nu = 1$: The robot writes symbol B in square 1. $x1 = 1$, $x2 = b$, $i0 = 4$, $y = B$. AM sees symbol B in square 1.
- $\nu = 2$: The robot looks at square 1 without writing anything. $x1 = 1$, $x2 = 0$ (no signal), $i0 = 4$, $y = B$. AM sees symbol B in square 1.
- $\nu = 3$: The robot writes symbol C in square 1. $x1 = 1$, $x2 = c$, $i0 = 7$, $y = C$. AM sees symbol C in square 1.

The experiment can be continued for other squares. The robot (system AM) always sees the last symbol written in square, because the corresponding location in the table of associations of AS has a higher level of residual excitation than any of the competing locations. This effect of symbolic STM lasts while residual excitation is higher than a certain level E_0 (in this case $E_0 = 1$). Once the residual excitation decays below this level, the STM is lost. This gives a qualitative explanation of the memory span and the magical number effect [16]. Since variable E is a continuous function of time, theoretically, one can get an arbitrarily large memory span by selecting the appropriate values of β and γ [4]. In practice, there are strong limitations associated with the level of noise.

4.5 Nonsymbolic Mental Set and Symbolic Society of Mind

Let us assume that the time constant of decay of $E(i)$ is sufficiently big ($\tau >> 1$). Then different profiles of $E(*)$ will create different biases in Model 4.3. Let $\tau = \infty$. In this case it can be rigorously proved that Model 4.3 with given table of associations (software) can be transformed into different combinatorial machines by changing $E(*)$. In fact one can fix the contents of LTM and transform Model 4.3 into an arbitrary combinatorial machine by simply changing E–states. The number of deterministic combinatorial machines with m input symbols and k output symbols is k^m. Let us say that each E–state represents a certain mental set of Model 4.3. Then at the behavioral level we will observe such different mental states as different symbolic machines.

Imagine a cognitive modeler trying to find a phenomenological description of the symbolic behavior of Model 4.3. Suppose this modeler has succeeded in deciphering the description of this behavior corresponding to a given E–state. Such a phenomenological theory would describe only k^{-m}-th part of the whole symbolic behavior of this model.

This reveals a problem associated with an attempt to develop purely phenomenological symbolic theories of human cognition. An attempt to describe the "society of mind" in purely symbolic terms leads to a combinatorial explosion of the number of the required "mental agents". The terms are borrowed from [18].

4.6 The Pitfall of a Wrong Dimensionality

Section 4.5 provided an illustration of a pitfall associated with an attempt to represent a part of behavior in a space of inadequately large dimensionality. This pitfall of a *wrong dimensionality* can be formulated as the following "Catch 22":
An attempt to reduce the dimensionality of a problem, to simplify the problem, makes the problem more complex.

Recall, for example, the relationship between the complex function and the real and imaginary parts of this function. The whole can be easier to describe than its parts.

Another example is given by the Maxwell equations. There exists a simple formal representation of the whole behavior of the electromagnetic field. In nontrivial cases, however, it is practically impossible to find separate descriptions of the behavior of either the electric or the magnetic field.

The metaphor "the brain as an E–machine" suggests that, similarly, it is impossible to separate the symbolic and nonsymbolic parts of the phenomenon of information processing in the brain.

4.7 Mental Imagery and Universality

The main idea of the robot shown in Figure 5 can be developed to produce a simplified explanation of our ability to perform, in principle, arbitrary mental calculations. Since AS learns to simulate external tape, it is sufficient for AM to learn to simulate an arbitrary finite–state machine to make the robot of Figure 5 a universal learning system. The simplest example of such system was described and studied in [4]. The model was aimed at illustrating the following ideas about the brain as a universal learning system:

1. The information processing universality of the brain is a result of interaction of two associative learning systems: the "active" SM→M system controlling our reactions (representing our "I") and the "passive" MS→S system simulating the external world as it is seen by the "active" system via sensory and motor devices. Both such systems can be arranged on the principle of E–machine outline in this paper.

2. Motor symbols serve as "nonterminals" that structure sensory data. A similar idea underlies the scanpath theory of [19].

3. By working with an external memory aid the brain learns to use its internal memory.

5 CONCLUSION

I have outlined what can be called the bare–bone architecture of a universal learning neurocomputer.

Such neurocomputer uses a very simple learning algorithm (rote learning) and can be taught, in an experiment of supervised learning, to simulate, in principle, an arbitrary Turing machine with finite tape. I have a working model of this neurocomputer, so there is no doubt that the basic ideas outlined in this paper do work.

A bare–bone conventional computer consists of a processor and memory. The power of this basic universal architecture can be dramatically increased by integrating additional computational resources.

A bare–bone universal learning neurocomputer consists of just two context–sensitive associative memories (or primitive E–machines). Importantly, conventional associative memories do not have enough functionality to implement such a neurocomputer.

The power of the bare–bone universal learning neurocomputer can be dramatically increased by integrating additional brain–inspired mechanisms. Some of such mechanisms were discussed in [4].

References

[1] Baddeley, A.D. 1982. Your memory: A user's guide. *MacMillan Publishing Co., Inc.*

[2] Chomsky, N. 1956. Three models for description of language. *I.R.E. Transactions on Information Theory, IT-2*, 113-124.

[3] Eliashberg, V. 1967. On a class of learning machines. *Moscow: Proceedings of VNIIB, #54*, 350-398.

[4] Eliashberg, V. 1979. The concept of E-machine and the problem of context-dependent behavior. *TXU 40-320, US Copyright Office.*

[5] Eliashberg, V. 1981. The concept of E-machine: On brain hardware and algorithms of thinking. *Proceedings of of the Third Annual Meeting of Cognitive Science Soc.*, 289-291.

[6] Eliashberg, V. 1988. Neuron layer with reciprocal inhibition as a mechanism of rendom choice. *Proceedings of the IEEE ICNN 88*, 17-25.

[7] Eliashberg, V. 1989. Context–sensitive associative memory: "Residual excitation" in neural networks as the mechanism of STM and mental set. *Proceedings of IJCNN-89, Vol 1*, 67-75.

[8] Eliashberg, V. 1990. Molecular dynamics of short–term memory. *Chicago, Illinoise: Proceedings of the 7-th International Conference on Math. and Computer Modeling*, 295-299.

[9] Feigenbaum, E.A., et al 1981. The handbook of artificial intelligence. *William Kaufmann, Inc.*

[10] Feldman, J.A. 1981. A connectionist model of visual memory. *In Hinton, E., & Anderson, 1981*, 49-81.

[11] Grossberg, S. 1982. Studies of mind and brain. *Boston: Reidel Press.*

[12] Hinton, E. & Anderson J. 1981. Parallel models of associative memory. *Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.*

[13] Hinton, G.E., Sejnowski, T.J., & Ackley, D.H. 1984. Bolzman machines: Constraint satisfaction networks that learn. *Pittsburg, PA: Carnegie–Mellon University, Dept. of Comp. Science*, Tech. Rep. No CMU-CS-84-119.

[14] Hopfield, J.J. 1982. Neural networks and physical systems with emergent collective computational abilitis. *Proceedings of the National Academy of Science USA, 79*, 2554-2558.

[15] McCulloch, W.S.,& Pitts, W. 1943. A logic calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*, 115-133.

[16] Miller, G.A. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63.* 81-97.

[17] Minsky, M. 1954. Neural nets and the brain–model problem. *Unpublished doctoral dissertation, Princeton University.*

[18] Minsky, M. 1986. The Society of Mind. *New York: Simon and Schuster.*

[19] Norton, D. & Stark, L. 1971. Scapaths in the eye movements during pattern perception.*Science, Vol. 171*, 308-311.

[20] Pinker, S. & Mehler, J. Eds. 1988. Connections and symbols.*Cambridge MA: MIT Press.*

[21] Reichardt, W., MacGinitie, G. 1962. Zur Theory der Lateralen Inhibition*Kybernetik, Vol. 1, Nr. 4.*

[22] Rumelhart, D.E., McClelland, J.L., and the PDP research group. 1986. Parallel distributed processing. *Cambridge MA: MIT Press.*

[23] Rosenblatt, F. 1962. Principles of neurodynamica. *Washington DC: Spartan Books.*

[24] Sperling, G.A. 1960. The information avalable in brief presentations. *Psychological Monographs, 74, No. 498.*

[25] Steinbuch, K. 1961. "Die Lernmatrix". *Kybernetik, Vol. 1*, 36-45.

[26] Turing, A.M. 1936. On computable numbers with an application to the Entsheidungsproblem . *Proc. London Math. Society, Ser. 2, 42.*

[27] Widrow, B. 1962. Generalisation and information storage in networks of Adaline neurons. In Self–organizing systems. *Washington DC: Spartan Books.*

[28] Zopf, G.W. 1962. Attitude and Context. In "Principles of Self–organization". *Pergamon Press*, 325-346.