A Revival of Turing's Forgotten Connectionist Ideas: Exploring Unorganized Machines

Christof Teuscher and Eduardo Sanchez
Logic Systems Laboratory, Swiss Federal Institute of Technology
CH – 1015 Lausanne, Switzerland
E-mail: `{name.surname}@epfl.ch`

Abstract

Turing had already investigated connectionist networks at the end of the forties and was probably the first person to consider building machines out of very simple, neuron-like elements connected together in a mostly random manner. The present paper aims to revive and shed more light on Turing's ideas. Turing's unorganized machines are analyzed and new types of machines are proposed by the authors. An example of a pattern classification task is presented using a Turing net and genetic algorithms for building and training the networks.

1. Introduction

In a little-known paper entitled "Intelligent Machinery" [7,15], Turing had already investigated connectionist networks at the end of the forties. His employer at the National Physical Laboratory in London, Sir Charles Darwin, dismissed the manuscript as a "schoolboy essay". Turing never had great interest in publicizing his ideas, so the paper went unpublished until 1968, 14 years after his death. Copeland and Proudfoot, directors of the *Turing Project*[1], a large ongoing project focused on Turing's life-work at the University of Canterbury, revived Turing's connectionist ideas in a Scientific American [5] publication.

In describing networks of artificial neurons connected in a random manner, Turing had written the first manifesto of the field of artificial intelligence. Many of the concepts that became later important to AI were introduced in this work. One of the questions he was always interested in was whether it is possible for machinery to show intelligent behavior or not. There is no doubt that Turing's work on neural networks goes far beyond the earlier work of McCulloch and Pitts [9]. However, it is interesting that Turing makes no reference in the 1948 report to their work. Turing's idea of using supervised interference to train an initially random arrangement of units to compute a specified function was new and revolutionary at that time and it is in fact not astonishing that his paper has been first dismissed as a "schoolboy essay".

Random boolean networks were seriously investigated many years later by Weisbuch [17] and Kauffman [8], who had probably never heard of Turing's networks that are in fact a subset of random boolean networks. Igor Aleksander is probably one of the few researchers that consider weightless neural architectures [1] as a serious

[1] http://www.alanturing.net

alternative to modern-day connectionist architectures. Other interesting work related to Turing's connectionism and random connected networks came from Allanson in 1956 [2], from Smith and Davidson in 1962 [13], from Rozonoér [11] in 1968, and in 1971 from Amari [3].

The present paper aims to revive and shed more light on Turing's ideas, to study, to simulate, and to analyze them. To the best of our knowledge, nobody, with the exception of the Turing project team and its former member Craig Webster[2], has ever analyzed in detail, simulated, implemented, and applied Turing's neural machines to pattern classification and other problems. Turing was not really interested in practical applications of his models. In a letter to Ashby, Turing remarked: "[...] I am more interested in the possibility of producing models of the action of the brain than in the practical applications to computing".

Our research about Turing nets goes far beyond Turing's proposals. Turing's unorganized machines, namely A-type and B-type networks, are analyzed and described in detail in section 2.. This section is a historical summary and provides the background for the rest of the paper. All the machines in section 2. were proposed by Turing. Section 3. introduces new machines, namely TB-type networks, proposed by the authors. Section 4. presents an example of a pattern classification problem on TB-type networks, and section 5. concludes the paper.

2. Unorganized Machines: Turing's Anticipation on Neural Networks

"Someday, perhaps soon, we will build a machine that will be able to perform the functions of a human mind, a thinking machine" [6]. This is the first sentence in Hillis' book on the *Connection Machine*, a legendary computing machine that provided a large number of tiny processors and memory cells connected by a programmable communications network. Much earlier in the 20^{th} century, Alan M. Turing described different types of machines—some well-known today, others almost forgotten. Turing's vision was probably very comparable to Hillis'. However, he only had pencil and paper at his disposal to verify the functioning of his machines. Throughout his entire life, Turing was interested in creating intelligent machines. One of the best-known, yet abstract, machines invented by Turing is certainly the classical Turing machine. Turing's lifelong interest in thinking machines and his concern with modelling the human mind using machines were probably at the origin of his rather little known paper on mechanical intelligence, a report for the National Physical Laboratory [15]. In the remainder of this section, we shall present the neuron-like machines Turing described in this paper.

2.1 Unorganized A-type and B-type Machines

The term *unorganized machines* was defined by Turing in a rather inaccurate and informal way. Turing's *unorganized machine* is a machine that is built up "[...] in

[2] http://home.clear.net.nz/pages/cw/

a comparatively unsystematic and random way from some kind of standard components" [15, p. 9]. Turing admitted that the same machine might be regarded by one man as organized and by another as unorganized. He gave an example of a typical unorganized machine:

> "The machine is made up from a rather large number N of similar units. Each unit has two input terminals, and has an output terminal which can be connected to input terminals of (0 or more) other units. We may imagine that for each integer $r, 1 \leq r \leq N$, two numbers $i(r)$ and $j(r)$ are chosen at random from $1...N$ and that we connect the inputs of unit r to the outputs of units $i(r)$ and $j(r)$. All of the units are connected to a central synchronizing unit from which synchronizing pulses are emitted at more or less equal intervals of time. The times when these pulses arrive will be called "moments". Each unit is capable of having two states at each moment. These states may be called 0 and 1".

The complete system is thus updated synchronously, in discrete time steps. The state of each unit is determined by the previous value of the two inputs and the transmission of information among neurons requires a unit time delay (a "moment"). In terms of a digital system, the basic unit which Turing describes can be straightforwardly defined as an edge-triggered D flip-flop preceded by a two-input NAND gate (Figure 1a).

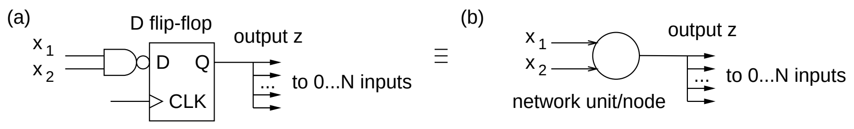
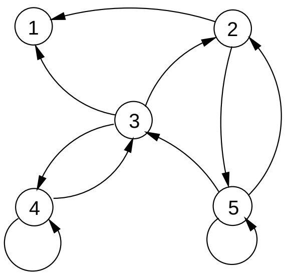


Figure 1: A basic unit regarded as a digital system: (a) D flip-flop preceded by a two-input NAND gate ([14]); (b) Symbolized network unit/node of an unorganized machine. Only this notation was used by Turing [15].

A positive-edge-triggered D flip-flop samples its D input and changes its Q output only at the rising edge of the controlling clock (CLK) signal [16]. In our case, the central synchronizing unit that emits the pulses is the global clock generator of the digital system.

In more modern terms, the unorganized machine Turing has proposed is a boolean neural network. However, modern networks are often organized in layers. For unorganized machines, the layered structure does not really exist since recurrent connections are allowed with no constraints. Furthermore, unorganized machines are normally constructed randomly and thus freely interconnected. The machines Turing proposed are in fact a subset of the random boolean networks described and investigated in detail by Stuart Kauffman [8].

Let us start with the simplest unorganized machine proposed by Turing. An A-*type unorganized machine* is a machine built up from units and connections as described in the example above. Figure 2 shows a five-unit A-type unorganized machine and a possible state sequence. Turing did not give any indication of how the inputs and outputs of his unorganized machines might interface with a given environment.



node	t	t+1	t+2	t+3	t+4	t+5
1	1	1	0	0	1	0
2	1	1	1	0	1	0
3	0	1	1	1	1	1
4	0	1	0	1	0	1
5	1	0	1	0	1	0

Figure 2: Example of an A-type unorganized machine built up from 5 units and a possible state sequence.

Turing suggested that an A-type unorganized machine is the simplest model of a nervous system with a random arrangement of neurons. Unfortunately, this statement came without any justification or proof.

A second type of unorganized machine described in his 1948 paper is called B-*type unorganized machine*. A B-type machine is an A-type machine where each connection between two nodes has been replaced by a small A-type machine called an *introverted pair*, as shown in Figure 3(a). This small A-type machine functions as a sort of switch (modifier) that is either opened or closed. In the next section, we will see how to operate this switch with an external stimulus. For now, this is not possible and the state of the switch (open or closed) only depends on the initial internal state of the small A-type machine that constitutes the interconnection. Turing simply wanted to create a switch based on the same basic element as the rest of the machine. He could have taken another element, i.e., a multiplexer, but he really wanted to construct the entire networks using the same basic element.

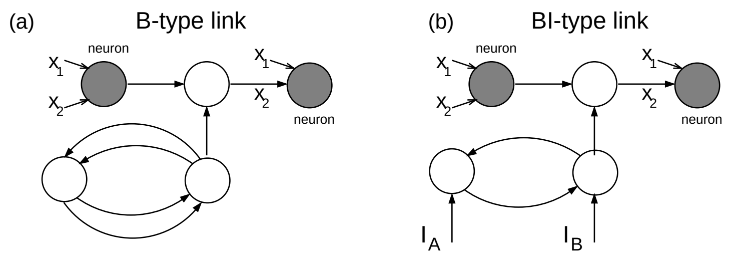


Figure 3: Turing's B-type links. Each B-type link is a small 3-node A-type machine also called an *introverted pair*. Part (b) shows a link with two interfering inputs.

The A-type machine that forms the B-type link can be in three different states of operation:

1. it may invert all signals (closed connection),
2. it may convert all signals into 1 (opened connection), or
3. it may act as in (1) and (2) alternately.

It can easily be seen that each B-type machine might be regarded as an A-type machine. The inverse statement does not hold since it is extremely unlikely that a B-type machine could be obtained by randomly assembling an A-type machine from a given number of units.

2.2 Turing's "Education": Organizing Unorganized Machines

So far, the unorganized machines we have described are, once initialized, no longer modifiable. It would clearly be interesting to modify the machine's interconnection switches (synapses) at runtime. Such a possibility would allow the use of an online learning algorithm that modifies the network's interconnections in order to train a net. Learning is undoubtedly the most difficult problem related to neural networks and much discussion is focused on it. Turing wrote: "It would be quite unfair to expect a machine straight from the factory to compete on equal terms with a university graduate" [15]. Thus, a machine, initially unorganized, must be organized: "Then by applying appropriate interference, mimicking education, we should hope to modify the machine until it could be relied on to produce definite reactions to certain commands" [15, p. 14]. This key idea is fascinating and can be found again in classical neural network learning algorithms. Turing went even further and wrote:

> "[...] with suitable initial conditions they [i.e., B-type machines] will do any required job, given sufficient time and provided the number of units is sufficient. In particular with a B-type unorganized machine with sufficient units one can find initial conditions which will make it into a universal machine with a given storage capacity" [15, p. 15].

However, Turing did not give a formal proof of this hypothesis because "[. . .] it lies rather too far outside the main argument". For a number of modern connectionist architectures such proofs have been given (see for example J. Pollack [10] and H. Siegelmann [12]).

In section 2.3 we will see that Turing was wrong and that it is impossible to compute every function with a B-type unorganized machine. Nevertheless, let us consider Turing's way of modifying a B-type interconnection. He simply proposed to replace a B-type link by a connection as shown in Figure 3(b). Each interconnection has two additional *interfering inputs* I_A and I_B, i.e., inputs that affect the internal state of the link. By supplying appropriate signals at I_A and I_B we can set the interconnection into state 1 or 2 (as explained in section 2.1). We will call this type of link a BI-type link (the I stands for "interference"). By means of this type of link, an external or internal agent can organize an initially random BI-type machine by disabling and enabling connections (switches). Neural networks usually have weighted connections with weights having real values. In contrast, there is nothing smooth in switching introverted pairs of a BI-type network: the connection is either enabled or disabled—a savage all-or-nothing shift.

2.3 The B-type Pitfall

It is easy to see that the principal computing element of an A-type network is the NAND operation since each node contains one 2-input NAND gate. Other logical functions thus have to be realized using NAND functions—which, as is a generally well known fact, is always possible since NAND units form a logical basis [16]. It thus directly follows that every logical function can be computed by an A-type network. The implementation of logical functions with lots of variables, however, can be very complex and tricky: the greatest difficulty lies in respecting the timing constraints due to the inherent delay of the D flip-flops.

As with A-type networks, the basic function of a B-type node is still a NAND operation. However, the two input signals of a node now pass by a B-type link that either inverts the signal (state 1) or always holds it to a 1 (state 2). One can see that a B-type node together with its associated input links is nothing more that a simple OR gate (when abstracting from the inherent delay)!

Since simple OR gates do not form a logical basis, it is now obvious that not all boolean functions can be computed with a B-type machine as defined by Turing. To understand the difficulty, one may attempt to design a B-type network that computes the XOR function or the NOT function. Turing's hypothesis about B-type machines stating that "[...] with suitable initial conditions they [i.e., B-type machines] will do any required job, given sufficient time and provided the number of units is sufficient" [15, p. 15] is wrong.

3. New Types of Unorganized Machines

In general, it is desirable to work with networks that offer universal computability, i.e., that are based on elements that form a logical basis. In this section, the authors present new types of machines that offer universal computability.

In order to obtain a computationally universal network, Copeland and Proudfoot [4] propose a remedy in form of the following B-type link modification: "[...] every unit-to-unit connection within an A-type machine is replaced by two of the devices [i.e., the introverted pairs, Figure 3(a)] [...] linked in series." Having two introverted pairs in each node-to-node connection adds useless nodes and connections in a network. In order to reduce the number of elements in the network, we propose to use another type of link—the TB-type link shown in Figure 4(a)—which is functionally equivalent to Copeland's and Proudfoot's proposal, but simpler. This simplification will be very useful for hardware implementations. The additional node simply inverts the input signal. One can see that a simple node together with its two associated TB-type links now forms a NAND operator. Since a NAND operator forms a logical basis, it is possible to realize any logical function with TB-type networks.

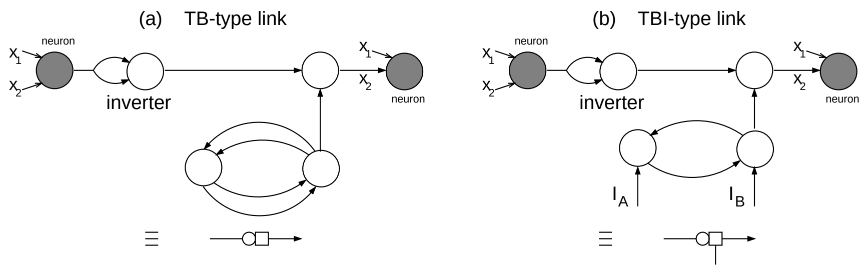


Figure 4: The TB-type (a) and TBI-type (b) link and notation proposed by Teuscher [14]. A simple node is used as an inverter in series with a normal introverted pair.

So far, we have seen that with a simple B-type or TB-type machine, the internal state of the links could not be modified due to the lack of interfering inputs. However, in section 2.2 we presented the BI-type machine that allows the interconnection switches to be modified by an external stimulus. It is easy to add this possibility also for TB-type links. Figure 4(b) shows how two interfering inputs may be added to the introverted pair of the TB-type link. The resulting link is called TBI-*type link*.

4. Pattern Classification using Turing Neural Networks and Genetic Algorithms

In order to simulate and test Turing neural networks, we implemented an object-oriented experimental framework [14]—called TNNSIM— that allows a detailed simulation of all Turing neural networks. In this section, we will present an experiment using genetic algorithms we performed using the TNNSIM simulator. Note that it was not the goal of this rather simple experiment to enter into competition with today's well-known and certainly better-performing pattern classification techniques.

The experiment consists in classifying into two classes (X and O) the twenty 16×16 dot patterns of Figure 5 without doing any explicit pre-processing. In order to determine the fitness of a network (individual of a GA population), all patterns are presented to the network. The network is set to its initial state (reset of all DFFs) before presenting a new pattern. The fitness of a network is defined as the number of correctly classified patterns. A steady-state genetic algorithm with a strong network encoding scheme has been used for this experience.

In this context, we call *learning* or *training* the process of modifying the interconnection switches since this process is very similar to the modification of synaptic weights in classical neural networks. We call *evolution* the process of building a suitable interconnection topology. During this phase, the switches are, in general, all closed.

A combination of topology evolution (with all connections enabled) and learning (connection modification only) was successful with a TBI-type network. A suitable interconnection topology was evolved during the first 200 generations. The topology was fixed, and only the switches were modified (enabled or disabled) during the learning phase. The network was built from 100 nodes. Figure 6 shows the fitness graph obtained with a population of 50 individuals (networks). This experiment showed that the initial interconnection topology is as least as important as the later learning phase. Learning is generally not successful if the underlying network structure is badly chosen.

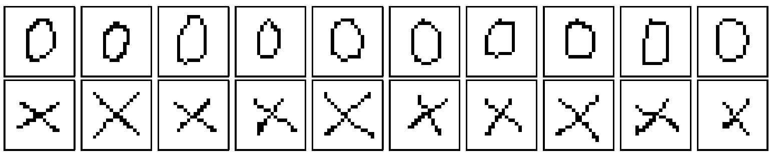


Figure 5: 16×16 dot patterns.

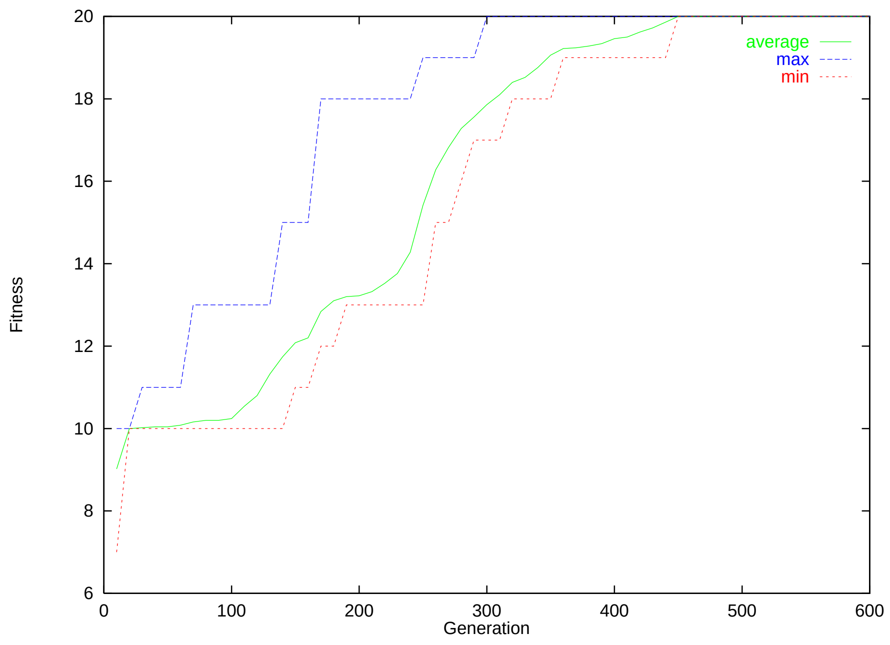


Figure 6: 16×16 dot pattern classification fitness with a TBI-type network. A network topology where each connection was enabled was first evolved. In a second step, only the connections were enabled or disabled. The population size was set to 50 individuals (networks).

5. Conclusion

Turing's idea of neural-like structures was new and promising in the forties. A universal and very simple basic element (neuron with associated input connections) in A-type and TB/TBI-type nets guarantees that, at least from a theoretical point of view, any function or system can be implemented using Turing's networks. The philosophy followed by Turing was to use the same type of node everywhere in the network, the reason why the modifiable interconnections are also built up from the same basic nodes.

We presented one example of a pattern classification experiment out of a series of experiments we performed with different characteristics and different levels of difficulty. It turned out that only the simpler pattern classification problems could be correctly solved by evolved Turing networks in a reasonable amount of time. However, due to the vast search spaces, it is often just a question of time to find a solution. We have shown that Turing's networks, like random boolean networks, "[...] inevitably classify and have internal models of their worlds" [8].

Turing's networks are probably more of theoretical than practical interest. Kauffman wrote: "[...] the classes formed by arbitrary Boolean networks are not necessarily natural, useful, or evolvable"[8]. However, randomly generated networks are very interesting to study the behavior of dynamic and complex systems, as Kauffman demonstrated impressively by his research.

Are Turing's nets of any interest for understanding biological systems? Yes, with limits. The neurons are far too simple to be compared with biological neurons. Turing himself, however, suggested that an A-type unorganized machine might be the simplest model of a nervous system with a random arrangement of neurons [15, p. 10]. The concept of using a large amount of identical and highly interconnected elements is powerful and could be the source of some insights on how biological systems might work and self-organize. Kauffman wrote [8]: "Most biological systems confront us with a vast number of connected elements. Typically we are ignorant of the details of the structure and logic by which the elements of such systems are coupled."

Turing might have asked if the networks give rise to higher-level cognitive functions, e.g., planning, abstraction, etc., since he was very interested in machines that think and demonstrate intelligent behavior. We have not focussed on that question so far.

Acknowledgements

The authors are grateful to Gianluca Tempesti for his careful reading of this work and his helpful comments.

References

1. I. Aleksander and H. Morton. *An Introduction to Neural Computing*. International Thomson Computer Press, London, UK, Boston, MA, USA, 2 edition, 1995.
2. J. T. Allanson. Some properties of randomly connected neural nets. In C. Cherry, editor, *Proceedings of the 3rd London Symposium on Information Theory*, pages 303–313, Butterworths, London, 1956.
3. S.-I. Amari. Characteristics of Randomly Connected Threshold-Element Networks and Network Systems. *Proceedings of the IEEE*, 59(1):35–47, January 1971.
4. B. J. Copeland and D. Proudfoot. On Alan Turing's anticipation of connectionsim. *Synthese: An International Journal for Epistemology, Methodology and Philosophy of Science*, 108:361–377, 1996. Kluwer Academic Publishers.
5. B. J. Copeland and D. Proudfoot. Alan Turing's Forgotten Ideas in Computer Science. *Scientific American*, pages 77–81, April 1999.
6. D. W. Hillis. *The Connection Machine*. The MIT Press, Cambridge, MA, London, England, 1985.
7. D. C. Ince, editor. *Mechanical Intelligence: Collected Works of A. M. Turing*, chapter Intelligent Machinery, pages 107–128. North-Holland, 1992.
8. S. A. Kauffman. *The Origins of Order: Self–Organization and Selection in Evolution*. Oxford University Press, New York, Oxford, 1993.
9. W. S. McCulloch and W. H. Pitts. A logical calculus of the ideas immanent in neural nets. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
10. J. B. Pollack. *On Connectionist Models of Neural Language Processing*. PhD thesis, Computer Science Department, University of Illinois, Urbana, 1987. (Available as MCCS-87-100, Computing Research Laboratory, NMSU, Las Cruces, NM).
11. L. I. Rozonoér. Random logical nets I. *Automation and Remote Control*, (5):773–781, 1969. Translation of Avtomatika i Telemekhanika.
12. H. T. Siegelmann and E. D. Sontag. On the Computational Power of Neural Nets. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 440–449, 1992.
13. D. R. Smith and C. H. Davidson. Maintened activity in neural nets. *JACM*, (9), 1962.
14. C. Teuscher. Study, Implementation, and Evolution of the Artificial Neural Networks Proposed by Alan M. Turing. A Revival of his "Schoolboy" Ideas. Master's thesis, Swiss Federal Institute of Technology Lausanne, Logic Systems Laboratory, EPFL-DI-LSL, CH-1015 Lausanne, February 2000. http://lslwww.epfl.ch/turing_nets.
15. A. M. Turing. Intelligent Machinery. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5 of *National Physical Laboratory Report*, pages 3–23. Edinburgh University Press, Edinburgh, 1969.
16. J. F. Wakerly. *Digital Design: Principles & Practices*. Prentice Hall International Inc., New Jersey, USA, 3rd edition, 2000.
17. G. Weisbuch. *Complex Systems Dynamics: An Introduction to Automata Networks*, volume 2 of *Lecture Notes, Santa Fe Institute, Studies in the Sciences of Complexity*. Addison-Wesley, Redwood City, CA, 1991.