ON ALAN TURING'S ANTICIPATION OF CONNECTIONISM

B. Jack Copeland and Diane Proudfoot

IN MEMORY OF ROBIN GANDY

ABSTRACT

It is not widely realised that Turing was probably the first person to consider building computing machines out of simple, neuron-like elements connected together into networks in a largely random manner. Turing called his networks 'unorganised machines'. By the application of what he described as 'appropriate interference, mimicking education' an unorganised machine can be trained to perform any task that a Turing machine can carry out, provided the number of 'neurons' is sufficient. Turing proposed simulating both the behaviour of the network and the training process by means of a computer program. We outline Turing's connectionist project of 1948.

1. Introduction

In a lecture given in London in 1947 Turing described the human brain as a 'digital computing machine' (1947, p.111). When he spoke of digital computing machines he had in mind a range of architectures considerably wider than the class of (what we would now call) von Neumann machines and their near relatives.[1] Turing was probably the first person to consider building computing machines out of simple, neuron-like elements connected together into networks in a largely random manner. Turing called his networks ‘unorganised machines’. His only published discussion of them occurs in a little-known report written in 1948 and entitled 'Intelligent Machinery'.

Turing describes three types of unorganised machine. A-type and B-type unorganised machines consist of randomly connected two-state 'neurons' whose operation is synchronised by means of a central digital clock. By the application of 'appropriate interference, mimicking education' a B-type machine can be trained to 'do any required job, given sufficient time and provided the number of units is sufficient' (1948, pp.14-15). His P-type unorganised machines, which are not neuron-like, have 'only two interfering inputs, one for "pleasure" or "reward" . . . and the other for "pain" or "punishment" ' (1948, p.17). Turing studied P-types in the hope of discovering training procedures 'analogous to the kind of process by which a child would really be taught' (1948, p.20). It is a P-type machine that Turing was speaking of when, in the course of his famous discussion of strategies for building machines to pass the Turing Test, he said ‘I have done some

experiments with one such child-machine, and succeeded in teaching it a few things' (1950, p.457). A-type, B-type and P-type machines are described more fully in what follows.

Turing had no doubts concerning the significance of his unorganised machines.

> [M]achines of this character can behave in a very complicated manner when the number of units is large . . . A-type unorganised machines are of interest as being about the simplest model of a nervous system with a random arrangement of neurons. It would therefore be of very great interest to find out something about their behaviour. (1948, p.10.)

He theorized that 'the cortex of the infant is an unorganised machine, which can be organised by suitable interfering training' (1948, p.16). Turing found 'this picture of the cortex as an unorganised machine . . . very satisfactory from the point of view of evolution and genetics' (1948, pp.16-17).

The 1948 report was prepared for the National Physical Laboratory, London, where Turing was employed as chief architect of the proposed Automatic Computing Engine or ACE. (He resigned from the NPL in 1948 to take up the position of Deputy Director of the Computing Laboratory at Manchester University.) Nominally the report was an account of research undertaken by Turing during a year he spent at Cambridge on sabbatical from the NPL; in fact it is a wide-ranging and strikingly original survey of the prospects for machine intelligence. In it

Turing anticipated many key developments in the field, including the theorem-proving approach to problem-solving (1948, p.22). Also present is the idea, subsequently made popular by Newell and Simon (1957, 1961, 1976), that 'intellectual activity consists mainly of various kinds of search' (1948, p.23). In 1969 the report appeared in the journal *Machine Intelligence*, but unfortunately this reprinting attracted little discussion. The report will become more widely known now that it is available in D.C. Ince's *Collected Works of A.M. Turing: Mechanical Intelligence*. It merits the same degree of attention that Turing's other major paper on artificial intelligence has attracted (Turing 1950).

2. Other Early Work On Neuron-Like Computation

As a result of his lukewarm interest in publication Turing's work on neuron-like computation remained unknown to others working in the area. His unorganised machines are not mentioned by the other pioneers of neuron-like computation in Britain, Ashby, Beurle, Taylor, and Uttley (Ashby 1952, Beurle 1957, Taylor 1956, Uttley 1956a, 1956b, 1959). The situation was the same on the other side of the Atlantic. Rosenblatt - the inventor of the perceptron and first to use the term 'connectionist' - seems not to have heard of Turing's unorganised machines (Rosenblatt 1957, 1958a, 1958b, 1959, 1962 esp. pp.5 and 12ff).[2] Nor is Turing's work mentioned in Hebb's influential book *The Organization of Behavior* (1949), the source of the so-called Hebbian approach to neural learning studied in connectionism today. Discussions of the history of connectionism by Rumelhart, McClelland et al (1986) show no awareness of Turing's early contribution to the field (see for example pp.41ff, 152ff, 424).

The pioneering work of Beurle, Taylor and Uttley has been neglected almost to the same extent as Turing's. According to connectionist folklore it was Rosenblatt, influenced by McCulloch, Pitts and Hebb, who originated the field of neuron-like computation. This is not the case. Rosenblatt records that the 'groundwork of perceptron theory was laid in 1957' (1962, p.27). A series of memoranda by Uttley concerning his probabilistic approach to neuron-like computation survives from as early as 1954 (Uttley 1954a-d) and published accounts of the work of all three men appeared prior to 1957 (see the references given above).

In 1958 Rosenblatt travelled to London and unveiled the perceptron at a symposium held at the National Physical Laboratory. He rashly claimed that 'the only machine prior to the perceptron which has shown itself to be capable of *spontaneous* improvement (as opposed to learning under the tutelage of an experimenter) has been Ashby's homeostat' (1959, p.424). In a frosty reply to Rosenblatt's paper Stafford Beer remarked:

> [T]here are tendencies to exaggerated claims to be found in the paper. ... [O]ne which ought really to be mentioned is that, apart from the Homeostat, the perceptron is the first machine 'to show spontaneous improvement'. This is not so. But perhaps Dr. Rosenblatt rightly assumed that, after all, everyone here must certainly have heard of Dr. Uttley's work. (Rosenblatt 1959, Discussion, p.463.)

Subsequent to the symposium Rosenblatt wrote generously of Uttley's work, describing him, along with Hebb, Hayek and Ashby, as having 'elaborated [the position] ... upon which the theory of the perceptron is based' (1958b, p.388). Concerning Taylor's work Rosenblatt wrote:

> Clearly our neuron models are very similar ... The one important difference which I see in our neuron models is in the choice of a suitable memory variable -- in Dr. Taylor's case the threshold, and in my own case, the strength or 'value' of the output signal. (1959, p.471.)

Rosenblatt's work was also prefigured in the U.S. by that of Clark and Farley (Farley and Clark 1954, Clark and Farley 1955), who in 1954 simulated a network of threshold units with variable connection weights. The training algorithm, or 'modifier', that they employed to adjust the weights during learning is similar to the algorithms subsequently investigated by Rosenblatt. Rosenblatt acknowledged that 'the mechanism for pattern generalisation proposed by Clark and Farley is essentially identical to that found in simple perceptrons' (1962, p.24).

3. Turing's Unorganised Machines

Turing introduces the idea of an unorganised machine by means of an example.

> A typical example of an unorganised machine would be as follows. 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 the 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. (1948, pp.9-10.)

Turing then gives (what would now be called) a propagation rule and an activation rule for the network. A propagation rule calculates the net input into a unit, and an activation rule calculates what the new state of a unit is to be, given its net input.

Propagation rule

The net input into unit r at moment m, $\mathbf{net}(r, m)$, is the product of the state of $i(r)$ at $m-1$ and the state of $j(r)$ at $m-1$.

Activation rule

The state of r at m is $1-\mathbf{net}(r, m)$.

A network of the sort described whose behaviour is determined by these two rules is an A-type unorganised machine.

In modern terminology an A-type machine is a collection of NAND units. (PNANDQ is $\neg(P \& Q)$.) The propagation rule in effect takes the conjunction of the values on the unit's two input lines, and the activation rule forms the negation of this value. Alternative choices of propagation rule and/or activation rule will cause the units to perform other Boolean operations. As is well known, NAND is more fundamental than certain other binary operators (including conjunction, material

implication, material equivalence, and inclusive and exclusive disjunction) in the sense that any Boolean operation can be performed by a circuit consisting entirely of NAND units. Thus any such operation can be performed by an A-type machine.

When considering A-type circuits one has to take account of the fact that each unit introduces a delay of one moment into the circuit. Suppose, for example, that the job of some particular unit U in the circuit is to compute XNANDY for some pair of specific truth-functions X and Y. The sub-circuits that compute X and Y may deliver their outputs at different moments yet obviously the values of X and Y must reach U at the *same* moment. Turing does not mention how this is to be achieved. A nowadays familiar solution to this problem - which also arises in connection with cpu design - involves the concept of a ‘cycle of operation’ of n moments duration. Input to the machine is held constant, or ‘clamped’, throughout each cycle and output is not read until the end of a cycle. Provided n is large enough then by the end of a cycle the output signal will have the desired value.

4. Trainable Boolean Networks

The most significant aspect of Turing's discussion of unorganised machines is undoubtedly his idea that an initially random network can be organised to perform a specified task by means of what he describes as 'interfering training' (1948, p.16).

> Many unorganised machines have configurations such that if once that configuration is reached, and if the interference

> thereafter is appropriately restricted, the machine behaves as one organised for some definite purpose. (1948, pp.14-15.)

Turing illustrates his idea by means of the circuit shown in figure 1 (1948, pp.10-11). (He stresses that this particular circuit is employed 'for illustrative purposes' and not because it is 'of any great intrinsic importance'.) We will call a pair of units connected in the way shown an 'introverted pair'. By means of external interference the state of unit *A* may be set to either 0 or 1. The state selected will be referred to as the 'determining condition' of the pair. (Concerning specific interfering mechanisms for changing the state of *A* Turing remarks '[i]t is . . . not difficult to think of appropriate methods by which this could be done' (1948, p.15). He gives one simple example of such a mechanism (ibid.).)

FIGURE 1 ABOUT HERE

As the reader may verify, the signal produced in unit *B*'s free output connection will be constant from moment to moment and the polarity of the signal will depend only upon the determining condition of the pair. Thus an introverted pair functions as an elementary memory.

Turing defines B-type machines in terms of a certain process of substitution applied to A-type machines: a B-type results if every unit-to-unit connection within an A-type machine is replaced by the device shown in figure 2 (1948, p.11). That is to say, what is in the A-type a simple connection between points *D* and *E* now passes via the depicted device. (Notice that all B-types are A-types.)

FIGURE 2 ABOUT HERE

Depending on the polarity of the constant signal at *C*, the signal at *E* will either (i) be 1 if the signal at *D* is 0 and 0 if the signal at *D* is 1, or (ii) always be 1 no matter what the signal at *D*. If by means of interference the state of *A* is changed from moment to moment, the device will cycle through these two alternatives. (This interference may be supplied either from outside or from within the network.)

In the first of these cases the device functions as a negation module. In the second case the device in effect disables the connection to which it is attached. That is to say, a unit with the device attached to one of its input connections delivers an output that is a unary function of the signal arriving along its other input connection. (If the devices on both the unit's input connections are placed in disable mode then the unit's output is always 0.) By means of these devices an external agent can organise an initially random B-type machine by selectively disabling and enabling connections within it. This arrangement is functionally equivalent to one in which the stored information takes the form of new connections within the network.[3]

Turing claims that it is a 'general property of B-type machines ... that with suitable initial [i.e. determining] conditions they will do any required job, given sufficient time and provided the number of units is sufficient' (1948, p.15). This follows from the more specific claim that given 'a B-type unorganised machine with sufficient units one can find initial conditions which will make it into a universal [Turing] machine with a given storage capacity' (1948, p.15).

Concerning the latter claim Turing remarks: ‘A formal proof to this effect might be of some interest, or even a demonstration of it starting with a particular unorganized B-type machine, but I am not giving it as it lies rather too far outside the main argument’ (1948, p.15).[4] It is a pity that Turing does not give any details of the proof, for this might have cast some light on what appears to be an inconsistency in his paper. It is reasonably obvious that not all Boolean functions can be computed by B-type machines as defined and thus that each of the claims quoted in the preceding paragraph is false. (A good way to get a feel for the difficulty is to attempt to design a B-type circuit for computing exclusive disjunction.) The simplest remedy seems to be to modify the substitution in terms of which B-type machines are defined: a B-type results if every unit-to-unit connection within an A-type machine is replaced by *two* of the devices shown in figure 2 linked in series. That is to say, what is in the A-type a simple connection between two units now passes through two additional units each with its own introverted pair attached. It is trivially the case that if a function can be computed by some A-type machine then it can also be computed by some machine satisfying the modified definition of a B-type. However, Turing’s paper contains no clues as to his own procedure.

As already indicated, the training process for a B-type unorganised machine consists in an external agent setting the determining condition of each introverted pair. In modern architectures repeated applications of a training algorithm - for example the back propagation algorithm - cause the encoding of the problem solution to 'evolve' gradually within the network during the training phase. Turing

had no algorithm for training his B-types. He regarded the development of training algorithms for unorganised machines as a central problem and he envisaged the procedure - first implemented by Farley and Clark and nowadays used extensively by connectionists - of programming the training algorithm into a computer simulation of the unorganised machine. With characteristic farsightedness Turing ends his discussion of unorganised machines by sketching the research programme that connectionists are now pursuing:

> I feel that more should be done on these lines. I would like to investigate other types of unorganised machines . . . When some electronic machines are in actual operation I hope that they will make this more feasible. It should be easy to make a model of any particular machine that one wishes to work on within such a UPCM [universal practical computing machine] instead of having to work with a paper machine as at present. If also one decided on quite definite 'teaching policies' these could also be programmed into the machine. One would then allow the whole system to run for an appreciable period, and then break in as a kind of 'inspector of schools' and see what progress had been made. (1948, pp.20-21.)

The importance that this project held for Turing may be gauged from a remark in a letter that he wrote to Ashby at about the same time:

> In working on the ACE I am more interested in the possibility of producing models of the action of the brain than in the practical applications to computing.[5]

Turing himself was unable to pursue his proposed research programme very far. It must be remembered that at the time he wrote, the only electronic stored-program computer in existence on either side of the Atlantic was a tiny pilot version of the Manchester Mark I.[6] (The pilot version of the ACE did not run its first program until 1950.) By the time he did have access to real computing power (at the University of Manchester) his interests had shifted and he devoted his time to modelling biological growth. It was not until the year of Turing's death that Farley and Clark, working at MIT, succeeded in running the first computer simulation of a small neural network (Farley and Clark 1954).

5. P-Type Unorganised Machines

Turing's main purpose in studying P-type machines seems to have been to search for general training procedures. A P-type machine is not a neural network but a modified Turing machine. Chief among the modifications is the addition of two input lines: the pleasure (or reward) line and the pain (or punishment) line. (Turing considers other modifications, in particular sensory input lines and internal memory units; in the interest of simplicity we say nothing further concerning these.) Unlike standard Turing machines a P-type has no tape.

Initially the P-type machine is unorganised in the sense that its machine table is 'largely incomplete' (1948, p.18). Application of either pleasure or pain by the trainer serves to alter an incomplete table to

some successor table. After sufficient training a complete table will emerge.

As is well known the machine table of a standard Turing machine consists of quintuples of the form: <state, symbol under head, symbol to be written, new state, direction of movement>. The P-types that Turing explicitly considers have machine tables consisting of triples. There follows an example of such a table. (This is a simplified version of Turing's own example on pp.18-19.)

State	Control Symbol	External Action
1	U	A
2	D0	B
3	T1	B
4	U	A
5	D1	B

'U' means 'uncertain', 'T' means 'tentative' and 'D' means 'definite'. (It is unnecessary to specify the nature of the external actions.) This table is incomplete in that no control symbol at all is specified for states 1 and 4 and the control symbol 1 has been entered only tentatively in the line for state 3. Only in the case of states 2 and 5 are definite control symbols listed. The table is complete only when a definite control symbol has been specified for each state.

The control symbol determines the state the machine is to go into once it has performed the specified external action. The rules that Turing gives governing the state transitions are:

(a) If the control symbol is 1 (either definitely or tentatively) then **next state** is the remainder of (2 × **present state**) + 1 on division by the total number of states (in this case 5).

For example, if the machine is in state 3 then **next state** is 2.

(b) If the control symbol is 0 (again, either definitely or tentatively) then **next state** is the remainder of 2 × **present state** on division by the total number of states.

For example, if the machine is in state 2 then **next state** is 4.

Let us suppose that the machine is set in motion in state 2. It performs the external action B, shifts to state 4, and performs the action A. No control symbol is specified in state 4. In this case the machine selects a binary digit at random, say 0, and replaces U by T0. The choice of control symbol determines the next state, in this case 3.

The trainer may apply a pleasure or pain stimulus at any time, with the effect that '[w]hen a pain stimulus occurs all tentative entries are cancelled, and when a pleasure stimulus occurs they are all made permanent' (1948, p.18). In other words pleasure replaces every T in the table by D and pain replaces all occurrences of T0 and T1 by U.

Turing suggests that 'it is probably possible to organise these P-type machines into universal machines' but warns that this 'is not easy' (1948, p.19). He continues:

> If, however, we supply the P-type machine with a systematic external memory this organising becomes quite feasible. Such a memory could be provided in the form of a tape, and the [external actions] could include movement to right and left

> along the tape, and altering the symbol on the tape to 0 or 1 . . . I have succeeded in organising such a (paper) machine into a universal machine . . . This P-type machine with external memory has, it must be admitted, considerably more 'organisation' than say the A-type unorganised machine. Nevertheless the fact that it can be organised into a universal machine still remains interesting. (1948, pp.19-20.)

As a search for 'teaching policies' Turing's experiments with P-types were not a success. The method he used to train the P-type with external memory required considerable intelligence on the part of the trainer and he describes it as 'perhaps a little disappointing', remarking that '[i]t is not sufficiently analogous to the kind of process by which a child would really be taught' (1948, p.20).

The key to success in the search for training algorithms was the use of weighted connections or some equivalent device such as variable thresholds. During training the algorithm increments or decrements the values of the weights by some small fixed amount. The relatively small magnitude of the increment or decrement at each step makes possible a smooth convergence towards the desired configuration. In contrast there is nothing smooth about the atomic steps involved in training a B-type. Switching the determining condition of an introverted pair from 0 to 1 or vice versa is a savage all-or-nothing shift. Turing seems not to have considered employing weighted connections or variable thresholds.

6. McCulloch-Pitts Networks

It is interesting that Turing makes no reference in the 1948 report to the work of McCulloch and Pitts, itself influenced by his own 1936 paper. Their 1943 article represents the first attempt to apply what they refer to as 'the Turing definition of computability' to the study of neuronal function (1943, p.129). McCulloch stressed the extent to which his and Pitts' work is indebted to Turing in the course of some autobiographical remarks made during the public discussion of a lecture given by von Neumann in 1948:

> I started at entirely the wrong angle ... and it was not until I saw Turing's paper [Turing 1936] that I began to get going the right way around, and with Pitts' help formulated the required logical calculus. What we thought we were doing (and I think we succeeded fairly well) was treating the brain as a Turing machine. (Von Neumann 1961, p.319.)

Like Turing, McCulloch and Pitts consider Boolean nets of simple two-state 'neurons'. They show (i) that such a net augmented by an external tape can compute all and only numbers that can be computed by Turing machines, and (ii) that without the external tape some but not all of these numbers, and no other numbers, can be computed by nets (ibid.). They give no discussion of universal machines.

McCulloch and Pitts make no use of weighted connections or variable thresholds. Part of the burden of their argument is to show that the behaviour of a net of binary units with variable thresholds can

be exactly mimicked by a simple Boolean net without thresholds 'provided the exact time for impulses to pass through the whole net is not crucial' (1943, pp.119, 123-4). They establish the same result in the case of various other phenomena associated with human neurons, in particular relative inhibition, extinction, temporal summation, and the formation of new synapses (ibid.).

McCulloch and Pitts say that two nets are 'equivalent in the extended sense' when the one exactly mimics the input/output behaviour of the other save possibly for the exact time it takes impulses to pass through the net (1943, pp.119, 123). There are some differences between the Boolean architectures of Turing and McCulloch and Pitts but there is equivalence in the extended sense. For example, inhibitory synapses are a primitive feature of McCulloch-Pitts nets but not of A-types and B-types. (An input of 1 at an inhibitory synapse at moment m unconditionally sets the output of the unit to 0 at m+1.) An inhibitory synapse can be mimicked by an arrangement consisting essentially of three introverted pairs, one modified in such a way that each unit of the pair has an external input. Working in the opposite direction, an introverted pair can be mimicked by one of the class of nets that McCulloch and Pitts call 'Nets with Circles'.

Turing had undoubtedly heard something of the work of McCulloch and Pitts. Wiener would almost certainly have mentioned McCulloch in the course of his 'talk over the fundamental ideas of cybernetics with Mr Turing' at the NPL in the spring of 1947 (Wiener 1948, p.32). (Wiener and McCulloch were founding members of the cybernetics movement.) Moreover von Neumann mentions the

McCulloch-Pitts article - albeit very briefly - in the 'First Draft of a Report on the EDVAC', which Turing read in 1945. (Von Neumann appears to have employed a modified version of their diagrammatic notation for neural nets in order to depict the EDVAC's logic gates. Turing went on to extend considerably the notation he found in the 'First Draft' (Carpenter and Doran 1986, p.277; Hartree 1949, pp.97, 102).) Turing and McCulloch seem not to have met until 1949. After their meeting Turing spoke dismissively of McCulloch, referring to him as a charlatan.[7] It is an open question whether the work of McCulloch and Pitts had any influence whatsoever on the development of the ideas presented in the 1948 report.

Over the years a number of commentators on the history of electronic computing machinery, especially those whose primary concern has been with developments in the United States, have found Turing's contributions hard to place. It has often been falsely assumed that most of the early work done in Britain on logical and electronic design was derived from work carried out in the United States, via von Neumann's report on the EDVAC (1945) and the Moore School lecture series of 1946. Turing did read von Neumann's report before writing his 'Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE)' but his design is strikingly different from von Neumann's (see Carpenter and Doran 1986, 1977; Huskey 1984). (Max Newman remarked in his obituary of Turing (*Manchester Guardian*, 11 June 1954) 'It was, perhaps, a defect of his qualities that he found it hard to accept the work of others, preferring to work things out for himself'.[8]) A fundamental difference between the machine that Turing

proposed and both the EDVAC and its British derivative the EDSAC was that in Turing's design complex behaviour was to be achieved by complex programming rather than by complex equipment. We know of Turing's opinion of Maurice Wilkes' design for the EDSAC from a memo Turing wrote in late 1946 or early 1947 to Womersley at the National Physical Laboratory: Wilkes' proposals are 'very contrary to the line of development here, and much more in the American tradition of solving one's difficulties by means of much equipment rather than by thought'.[9]

Turing followed his own path from the abstract machines of his 1936 paper to the idea of a high-speed stored-program computer and contemporaneous research in the United States meant little to him.[10] The same may well have been true in the case of his work on neuron-like computation.

Whatever the influences were on Turing at that time, there is no doubt that his work on neural nets goes importantly beyond the earlier work of McCulloch and Pitts. The latter give only a perfunctory discussion of learning, saying no more than that the mechanisms supposedly underlying learning in the brain - they specifically mention threshold change and the formation of new synapses - can be mimicked by means of nets whose connections and thresholds remain unaltered (1943, pp.117, 124). Turing’s idea of using supervised interference to train an initially random arrangement of units to compute a specified function is nowhere prefigured.[11]

NOTES

1. That Turing anticipated connectionism was first suggested to us by Justin Leiber in correspondence. Leiber gives a brief discussion on pp.117-18 and p.158 of his 1991 and on p.59 of his 1995. We cannot endorse Leiber's claim that Turing made use of weighted connections (1991, p.118).

2. Rosenblatt introduces the term ‘connectionist’ in the following way: ‘[According to] theorists in the empiricist tradition ... the stored information takes the form of new connections, or transmission channels in the nervous system (or the creation of conditions which are functionally equivalent to new connections) ... The theory to be presented here takes the empiricist, or “connectionist” position ... The theory has been developed for a hypothetical nervous system, or machine, called a *perceptron*’ (1958b, p.387).

3. See note 2.

4. Such proofs have been given for a number of modern connectionist architectures, for example by Pollack 1987 and Siegelmann and Sontag 1992. The latter establish the existence of a network capable of simulating a finite-tape universal Turing machine in linear time. They are able to give an upper bound on the size of the network: at most 1058 units are required.

5. The letter is held in the NPL archive, Science Museum, South Kensington, London. It is undated but was written while Turing was working at the NPL.

6. Turing wrote the report during July and August of 1948 (Hodges 1983, p.377). The prototype Manchester machine ran its first progam in June 1948 and the Cambridge EDSAC not until May 1949 (Kilburn and

Williams 1953, p.120; Wilkes, M. 1985, p.142). The American ENIAC first went into operation in November 1945 but was not stored-program. It was not until September 1948 that the ENIAC began operating with program code stored in its function tables (Goldstine 1972, p.233; Metropolis and Worlton 1980, pp.53-54). However, since the function tables were read-only, variable addressing was not possible and so the ENIAC was never stored-program in the full sense. The American BINAC, which was stored-program, was first tested in August 1949 (Stern 1979, pp.12-13). The IBM SSEC, which first ran in public in January 1948, was stored-program but was not fully electronic, being largely electromechanical (Bowden 1953, pp.174-5; Eckert 1948). Despite a widespread belief to the contrary, the ENIAC was not the first electronic computing machine. This distinction belongs to the Colossi, constructed at Bletchley Park, Buckinghamshire, from 1943 onwards for use against the German 'fish' codes. The Colossus was program-controlled rather than stored-program. It performed Boolean operations and had some arithmetical capability (Randell 1980). The Colossus was designed and built by Flowers and Newman with - according to Donald Michie, himself a junior member of Newman's team - some assistance from Turing. (Possibly Turing's contributions concerned the methods of search employed by the Colossus.) The existence of the Colossus was kept secret by the British government for a number of years after the war and so it was that von Neumann and others, in lectures and public addresses, told the world that the ENIAC was 'the first electronic computing machine' (von Neumann 1954, pp.238-9). (The Manchester firm Ferranti built a production version of the Manchester Mark I. The

first was installed in February 1951, two months before the first UNIVAC (Lavington 1975, p.20).)

7. This is Robin Gandy's recollection.

8. Quoted in Kleene 1987, p.492.

9. The memo is reproduced in Huskey 1984, p.354.

10. Turing's wartime involvement with electronics was no doubt the key link between his earlier theoretical work and the ACE: by means of this new technology the abstract stored-program machines of his 1936 paper could be turned into a reality. (Very probably his wartime involvement with the Colossus at Bletchley Park was a particularly important influence in that respect.) Here is Turing's own statement of the relationship between the universal Turing machine and the ACE.

> Some years ago I was researching on what might now be described as an investigation of the theoretical possibilities and limitations of digital computing machines. I considered a type of machine which had a central mechanism, and an infinite memory which was contained on an infinite tape . . . Machines such as the ACE may be regarded as practical versions of this same type of machine. (1947, pp.106-7.)

In the previously mentioned letter to Ashby Turing says:

> The ACE is in fact, analogous to the "universal machine" described in my paper on conputable [sic] numbers . . . [W]ithout altering the design of the machine itself, it can, in theory at any rate, be used as a model of any other machine, by making it remember a suitable set of instructions.

As Hodges suggests (1983, p.556) Sara Turing was probably more or less quoting her son's own words when she wrote in connection with the

ACE project that 'his aim [was] to see his logical theory of a universal machine, previously set out in his paper "Computable Numbers" [sic] . . . take concrete form in an actual machine' (1959, p.78).

11. We are indebted to John Andreae, Sean Broadley, Nat Gilman, Simeon Lodge, Donald Michie, Seth Wagoner and Justin Zajac for discussion, and especially so to Craig Webster. John Andreae, Kevin Korb and Craig Webster commented helpfully on earlier versions of this paper. Craig and Bruce Webster wrote a B-type simulator for us.

REFERENCES

Ashby, W.R. 1952. ‘Design for a Brain’. London: Chapman and Hall.

Beurle, R.L. 1957. ‘Properties of a Mass of Cells Capable of Regenerating Pulses’. *Philosophical Transactions of the Royal Society of London*, Series B, 240, pp.55-94.

Bowden, B.V. (ed.) 1953. *Faster than Thought*. London: Pitman.

Carpenter, B.E., Doran, R.W. 1977. 'The Other Turing Machine'. *Computer Journal*, 20, pp.269-79.

Carpenter, B.E., Doran, R.W. (eds) 1986. *A.M. Turing's ACE Report of 1946 and Other Papers*. Cambridge, Mass.: MIT Press.

Clark, W.A., Farley, B.G. 1955. ‘Generalisation of Pattern Recognition in a Self-Organising System’. *Proceedings of the Western Joint Computer Conference*, pp.86-91.

Eckert, W.J. 1948. 'Electrons and Computation'. In Randell, B. (ed) 1982. *The Origins of Digital Computers*. Berlin: Springer-Verlag, pp.219-28.

Farley, B.G., Clark, W.A. 1954. ‘Simulation of Self-Organising Systems by Digital Computer’. *Institute of Radio Engineers Transactions on Information Theory*, 4, pp.76-84.

Goldstine, H.H. 1972. *The Computer from Pascal to von Neumann*. Princeton: Princeton University Press.

Hartree, D.R. 1949. *Calculating Instruments and Machines*. Illinois: University of Illinois Press.

Hebb, D.O. 1949. *The Organization of Behavior: A Neuropsychological Theory*. New York: John Wiley.

Hodges, A. 1983. *Alan Turing: The Enigma*. London: Burnett.

Huskey, H.D. 1984. 'From ACE to the G-15'. *Annals of the History of Computing*, 6, pp.350-71.

Ince, D.C. (ed.) 1992. *Collected Works of A.M. Turing: Mechanical Intelligence*. Amsterdam: North Holland.

Kilburn, T., Williams, F.C. 1953. 'The University of Manchester Computing Machine'. In Bowden 1953, pp.117-129.

Kleene, S.C. 1987. 'Reflections on Church's Thesis'. *Notre Dame Journal of Formal Logic*, 28, pp.490-98.

Lavington, S.H. 1975. *A History of Manchester Computers*. Manchester: NCC Publications.

Leiber, J. 1991. *An Invitation to Cognitive Science*. Oxford: Basil Blackwell.

Leiber, J. 1995. 'On Turing's Turing Test and Why the Matter Matters'. *Synthese*, 104, pp.59-69.

McCulloch, W.S., Pitts, W. 1943. 'A Logical Calculus of the Ideas Immanent in Nervous Activity'. *Bulletin of Mathematical Biophysics*, 5, pp.115-33.

Minsky, M.L., Papert, S. 1969. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, Mass.: MIT Press.

Metropolis, N., Worlton, J. 1980. 'A Trilogy of Errors in the History of Computing', *Annals of the History of Computing*, 2, pp.49-59.

Newell, A., Shaw, J.C., Simon, H.A. 1957. 'Empirical Explorations with the Logic Theory Machine: a Case Study in Heuristics'. In Feigenbaum, E.A., Feldman, J. (eds) 1963. *Computers and Thought*. New York: McGraw-Hill, pp.109-133.

Newell, A., Simon, H.A. 1961. 'GPS, a Program that Simulates Human Thought'. In Feigenbaum, E.A., Feldman, J. (eds) 1963. *Computers and Thought*. New York: McGraw-Hill, pp.279-293.

Newell, A., Simon, H.A. 1976. 'Computer Science as Empirical Inquiry: Symbols and Search'. In Haugeland, J. 1981. *Mind Design: Philosophy, Psychology, Artificial Intelligence.* Cambridge, Mass.: MIT Press, pp.35-66.

Pollack, J.B. 1987. *On Connectionist Models of Natural Language Processing*. Ph.D. Dissertation, University of Illinois, Urbana.

Randell, B. 1980. 'The Colossus'. In Metropolis, N., Howlett, J., Rota, G.C. (eds) 1980. *A History of Computing in the Twentieth Century.* New York: Academic Press, pp.47-92.

Rosenblatt, F. 1957. *The Perceptron, a Perceiving and Recognizing Automaton*. Cornell Aeronautical Laboratory Report No. 85-460-1.

Rosenblatt, F. 1958a. *The Perceptron: a Theory of Statistical Separability in Cognitive Systems.* Cornell Aeronautical Laboratory Report No. VG-1196-G-1.

Rosenblatt, F. 1958b. 'The Perceptron: a Probabilistic Model for Information Storage and Organisation in the Brain'. *Psychological Review*, 65, pp.386-408.

Rosenblatt, F. 1959. 'Two Theorems of Statistical Separability in the Perceptron'. In *Mechanisation of Thought Processes*, Vol.1. London: H.M. Stationery Office, pp.419-72.

Rosenblatt, F. 1962. *Principles of Neurodynamics.* Washington, D.C.: Spartan.

Rumelhart, D.E., McClelland, J.L., and the PDP Research Group 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Vol.1: *Foundations*. Cambridge, Mass.: MIT Press.

Siegelmann, H.T., Sontag, E.D. 1992. 'On the Computational Power of Neural Nets'. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp.440-449.

Stern, N. 1979. 'The BINAC: A Case Study in the History of Technology'. *Annals of the History of Computing*, 1, pp.9-20.

Taylor, W.K. 1956. 'Electrical Simulation of Some Nervous System Functional Activities'. In Cherry, C. (ed.) 1956. *Information Theory*. London: Butterworths, pp.314-28.

Turing, A.M. 1936. 'On Computable Numbers, with an Application to the Entscheidungsproblem'. *Proceedings of the London Mathematical Society*, Series 2, 42 (1936-37), pp.230-65.

Turing, A.M. 1946. 'Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE)'. In Carpenter and Doran 1986, pp.20-105.

Turing, A.M. 1947. 'Lecture to the London Mathematical Society on 20 February 1947'. In Carpenter and Doran 1986, pp.106-24.

Turing, A.M. 1948. 'Intelligent Machinery'. National Physical Laboratory Report. In Meltzer, B., Michie, D. (eds) 1969. *Machine Intelligence 5*. Edinburgh: Edinburgh University Press, pp.3-23. Reproduced with the same pagination in Ince 1992.

Turing, A.M. 1950. 'Computing Machinery and Intelligence'. *Mind* 59, pp.433-60.

Turing, S. 1959. *Alan M. Turing*. Cambridge: W. Heffer.

Uttley, A.M. 1954a. ‘Conditional Probability Machines and Conditioned Reflexes’. Radar Research Establishment Memorandum No. 1045.

Uttley, A.M. 1954b. ‘The Classification of Signals in the Nervous System’. R.R.E. Memorandum No. 1047.

Uttley, A.M. 1954c. ‘The Probability of Neural Connections’. R.R.E. Memorandum No. 1048.

Uttley, A.M. 1954d. ‘The Stability of a Uniform Population of Neurons’. R.R.E. Memorandum No. 1049.

Uttley, A.M. 1956a. ‘Conditional Probability Machines and Conditioned Reflexes’. In Shannon, C.E., McCarthy, J. (eds) 1956. *Automata Studies*. Princeton: Princeton University Press, pp.253-75.

Uttley, A.M. 1956b. ‘Temporal and Spatial Patterns in a Conditional Probability Machine’. In Shannon, C.E., McCarthy, J. (eds) 1956. *Automata Studies*. Princeton: Princeton University Press, pp.277-85.

Uttley, A.M. 1959. ‘Conditional Probability Computing in a Nervous System’. In *Mechanisation of Thought Processes*, Vol.1. London: H.M. Stationery Office, pp.121-47.

Von Neumann, J. 1945. 'First Draft of a Report on the EDVAC'. An edited version appears in Randell, B. (ed) 1982. *The Origins of Digital Computers*. Berlin: Springer-Verlag, pp.383-92.

Von Neumann, J. 1954. 'The NORC and Problems in High Speed Computing'. In von Neumann 1961, pp.238-47.

Von Neumann, J. 1961. *Collected Works*. Vol.5. Taub, A.H. (ed.). Oxford: Pergamon Press.

Wiener, N. 1948. *Cybernetics*. New York: John Wiley.

Wilkes, M.V. 1985. *Memoirs of a Computer Pioneer*. Cambridge, Mass.: MIT Press.