Bootstrapping Neural Electronics from Lunar Resources for In-Situ Artificial Intelligence Applications

Alex Ellery(✉)

Carleton University, Ottawa, ON K1S 5B6, Canada
aellery@mae.carleton.ca

Abstract. Artificial intelligence and robotics are leveraging technologies for lunar exploration. However, future lunar surface exploration will require exploitation of in-situ resources to reduce (and ultimately eliminate) the costs imposed by the transport of materiel from Earth. Solid-state manufacturing of electronics assets from lunar resources to eliminate its supply from Earth is impractical. We propose the in-situ manufacture of vacuum tube-based computational electronics which requires only a handful of materials that are available on the Moon. To offset the problem of exponential growth in physical footprint in CPU-based electronics, we propose the implementation of analogue neural network hardware which has Turing machine capabilities. We suggest that the artificial intelligence requirements for lunar industrialisation ecology can be demonstrated in principle by analogue neural networks controlling a small rover. We pay particular attention to online learning circuitry as the key to adaptability of analogue neural networks.

Keywords: Analogue neural network hardware · Turing machine models · Lunar in-situ resource utilisation · Neural network rover navigation

1 Introduction

Everyone is choosing to go the Moon, not because it is easy, but because it has potentially useful resources. Current interest in in-situ resource utilisation (ISRU) on the Moon is based on recovering water ice at the polar regions ostensibly to support a human presence. The advent of commercial, government and other interests towards lunar resources leads to a loose confederation of private and government installations on the Moon, the "Moon Village". However, rather than adopting ISRU as an addendum to human bases on the Moon, we are interested in leveraging lunar resources as a *modus operandi* for total lunar industrialisation as independent of Earth as is feasible, i.e. to bootstrap an entire infrastructure from lunar resources. A crucial backbone to this notion is the ability to construct machines of production rather than products *per se* from lunar resources Any kinematic machine – be they load-haul-dump rovers, CNC milling machines, 3D printers or assembly manipulators as the machines of production - comprises specific configurations of electric motors supported by control systems capable of universal computation.

M. Bramer and F. Stahl (Eds.): SGAI-AI 2022, LNAI 13652, pp. 83–97, 2022.
https://doi.org/10.1007/978-3-031-21441-7_6

It is the latter that is of concern here. A general approach to exploiting in-situ resources on the Moon involves extracting around 10 materials sufficient to realise all basic functional subsystems of a spacecraft – a demandite list extracted through a lunar industrial ecology [1, 2] which feeds into a generalised metal extraction system [3]. Two examples of utilities that can be leveraged from such in-situ resources are an energy infrastructure [4] and lunar bases [5]. We wish specifically to construct computational resources from in-situ resources on the Moon. An important consideration is that we cannot import Earth technology with all its complex supply chains to the extraterrestrial environment. Any technology we employ on the Moon must have a robust lunar supply chain with minimal reliance on Earth. So it must be with lunar computers which must function in a thermally hostile and radiation-saturated environment. We cannot simply shoehorn terrestrial technologies wholesale to the Moon it is premised on the entire globally-interconnected industrial complex on Earth. A mobile phone for instance comprises over 30 different materials for different functional components – each material must be mined, extracted, refined, mixed, formed and assembled. The manufacturing process assumes a bevy of reagents and high precision terrestrial facilities. The in-situ lunar environment imposes severe constraints on the availability of material and infrastructure resources. Here, we focus on using lunar resources to construct neural electronics capabilities in-situ as a step towards full self-sufficiency of computer technology necessary to realise artificial intelligence functions. The implementation of solid-state computers would be impossible from lunar resources due to [6]: (i) paucity of common reagents used in solid-state manufacture; (ii) stringently controlled environmental conditions required in solid-state manufacture; (iii) extreme temperatures required for solid-state manufacture; (vi) unsuitability of solid-state electronics to severe thermal and radiation environments extant on the Moon; (v) extreme cost of electronics foundries. Furthermore, given the paucity of plastic ingredients – carbon in particular – and exotic metals on the Moon suggest that approaches using organic polymers, nanoparticles or exotic metals cannot be substituted on the Moon. We must adopt a new engineering philosophy that exploits technologies derived from the available materials.

2 Modes of Computation

We propose thermionic vacuum tubes as an alternative to transistors because they comprise of only a small number of materials configured into a relatively simple construction, all of which can be sourced from the Moon. A vacuum tube is simple in construction – a tungsten cathode that emits electrons attracted to a nickel anode controlled by a third nickel grid electrode encased in an evacuated glass or ceramic tube and linked by silicone or ceramic-insulated kovar wiring. Fused silica glass encapsulation may be derived from lunar anorthite), aluminum wire from lunar anorthite, high temperature kovar wiring from nickel-iron meteorites, tungsten filament cathodes from nickel-iron meteorites coated with calcium oxide and alumina from lunar anorthite, and nickel anodes and control grids from nickel-iron meteorites. Only a small number of materials are required and are readily extracted from lunar resources. The chief problem resides in that vacuum tube-based computers are cumbersome – ENIAC used almost 17,500 vacuum tubes, over 7000 diodes, 1500 relays, 70,000 resistors and 10,000 capacitors

with a mass of over 25 tonnes covering an area of almost 170 m^2 and consumed 150 kW of power. Clearly, the enormity of such a computer renders this approach infeasible for a lunar infrastructure. This is a direct result of the von Neumann central processing unit (CPU)-based architecture. The core of the CPU is one or more arithmetic logic units (ALU), each a combinatorial logic circuit for performing arithmetic operations (addition, subtraction, negate, increment/decrement and sign) and bitwise logical operations (AND, OR, EX-OR and NOT) on 4-bit, 8-bit, 16-bit, 32-bit or 64-bit data widths. For example, the modest embedded 8051 CPU comprises 2,200 logic gates but modern computers comprise ~500 million logic gates. The von Neumann architecture stores data in a variety of different memory locations which must be fetched as input data to the CPU and the results of which must be pushed back into memory. The basic operation of the von Neumann architecture is the fetch-decode-execute cycle which is wasteful in hardware footprint and energy. There are alternative Turing machine-equivalent approaches to computation. We have adopted a neural network architecture implemented as electronics of such a machine. This was driven by several constraints: (1) limited material availability on the Moon for electronics manufacture, (2) superiority over the traditional von Neumann architecture in terms of Turing computability, (3) superiority of neural architectures over von Neumann architectures in terms of physical footprint.

3 3D Printer-Based Turing Machine

To address the problem of 3D printing computing machines, we revert to the original Turing machine model of a computer. Our implementation of a Turing machine comprises an input tape represented by magnetic core memory rather than a magnetic tape, an output tape represented by an analogue neural network circuit and a read/write head represented by a generic 3D printer (Fig. 1).

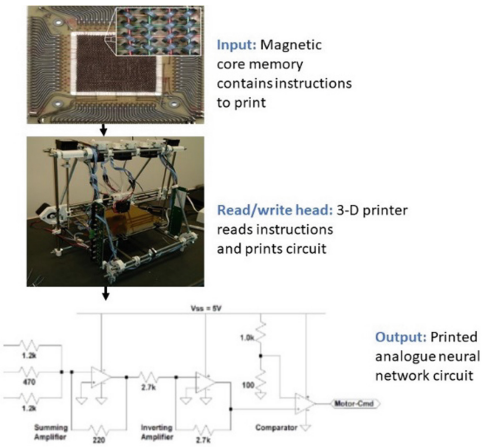

Fig. 1. Turing machine model with magnetic core memory (input tape), neural net circuit (output tape) and 3D printer (read/write head).

Magnetic core memory comprises the same basic components as a DC electric motor – 3D printing of such has been demonstrated [7]. Magnetic core memory uses ferrite magnetic cores (toroids) through which wires are passed to convey read and write signals. Each core stores 1 bit of information non-volatilely as zero or one depending on the direction of the core's magnetisation. The coincident current system enables a small number of wires to control a large number of cores in 3D stacks. A large number of small ferrite toroidal cores are held on layers of XY grids of wires through the toroidal centres. Only where the combined magnetic field from X and Y lines cross exceeds the threshold will the magnetic polarity reverse. Magnetic core memory offers high reliability and was used for the Apollo Guidance Computer and Space Shuttle Flight Computers. An automated assembler machine based on a four-axis cartesian (x,y,z,R) gantry that positions two tooling heads with respect to a worktable constitutes a possible model of the 3D printer [8]. The tooling heads position standard parts for assembly with compliance to accommodate small positioning errors. A punch-press was used for fabricating the modular boards of electrical insulator FR4 fibreglass-epoxy composite. A wire-electric discharge machine was used for fabricating electrically conducting tin-plated phosphor-bronze alloy wire. Together, these two materials constituted the modules for the assembly of electrical circuits. In our Turing machine model, the 3D printer thus becomes a central component of universal computation that operates in much the same way as a Jacquard loom – it prints out analogue neural network hardware circuitry according to the program stored in magnetic core memory as its output.

We must now consider the 3D printed output circuitry. For our output tape, we have adopted the artificial neural network whose architectural complexity grows only logarithmically with space and time resource requirements compared with the exponential growth in von Neumann architectures – any function that can be computed by logic gates of size z and depth d can be computed by a neural network of depth d/elog(z) and size $O(z^{1+e})$ with error $e > 0$ [9]. Neural networks are capable of implementing logic and arithmetic functions directly: (i) a finite neural network of discrete neurons characterised by Heaviside squashing functions (McCulloch-Pitts neurons) is equivalent to a finite sequential automaton implementing Boolean logic gates [10]; (ii) the half-adder can be realised as two oscillator output neurons cross-strapped to their inputs with excitatory connections and a direct bidirectional inhibitory link between the two neurons. Recurrent neural networks (for example, the Elman simple recurrent neural net is a Moore machine) are Turing machines so any computation that can be performed by a von Neumann architecture-based CPU can be performed with a recurrent neural network. A universal Turing machine was implemented on a recurrent neural network comprising 886 neurons [11] though this was subsequently reduced to 96 then 25 and finally to just 9 neurons [12]. A neural network with $r = m + n + 2$ layers of neurons and two sets of second order connection weights can simulate in real-time a Turing machine with m-symbols and n-states [13]. The neural Turing machine is based on a recurrent neural network computer augmented by external addressable memory [14]. The recurrent neural network acts as a controller network that performs processing of input data and data stored in memory, then storing it and outputting the results. Content-based addressing through partial similarity matching of vectors simplifies data retrieval

from the external memory. Like recurrent neural networks in general, the neural Turing machine can be trained through gradient descent learning.

There is a bio-inspired rationale to the adoption of analogue neurons – neocortical neurons exhibit the multistability of digital flipflop circuits with the graded response of analogue circuits modulated by positive feedback gain in its recurrent connections [15]. These neuromorphic computing architectures favour analogue electronics with digital encoding implemented as integrate-and-fire (spiking) neurons rather than McCulloch-Pitts neurons [16]. However, we do not address spiking neurons here. The physical circuit footprint of analogue electronics is often more efficient *ceteris paribus* than digital architectures. Analogue neural networks implement asynchronous event-driven computing eliminating the processor clock and reducing power consumption. They are more tolerant of low-accuracy components than conventional logic-based computation. Although analogue circuits suffer from noise and parameter variations, they offer high speed and low energy consumption. The use of physical neural networks through its synaptic weights co-locating data memory and data processing in massively parallel architectures such as SpiNNaker (spiking neural network architecture) eliminate the von Neumann bottleneck of CPUs and GPUs which waste energy [17]. In-memory computing may be based on crossbar arrays of memory cells that conduct analogue multiply-accumulate operations typical of a neural network of neurons [18].

4 Analogue Neural Network Learning Circuitry

Online learning is a highly desirable capability – indeed, it is essential for compensating for unreliable components, a trait of analogue circuitry. In general, neural learning involves weight update $w_{ij}(t+1) = w_{ij}(t) + \eta\delta_{ij}x_{ij}$ performed at each iteration until the output of the neural network minimised the error from the desired outputs $\delta_{ij}^{out} = (x_{ij}^{d} - x_{ij})x_{ij}(1 - x_{ij})$. Implementing neural learning through weight update with analogue circuitry represents a challenge. A simple learning algorithm is the Madaline rule II (MRII) which implements feedforward neural network training so that weight changes are minimized [19] – MRII extends the MR I to multilayer networks. This is the minimum disturbance principle. All weights are set to small random values and training patterns are then presented in random order. In response to an output error in the network, the MRII algorithm selects first layer neuron with the lowest output value and reverses it. The network output error is checked again: if the network output error is reduced, the weight change is accepted, otherwise the original weight is restored. This process continues with first layer neurons with increasing output values until all the output errors are corrected. If all the output errors are not zero, the procedure is then applied to pairs of first layer neurons beginning with those with outputs close to zero. Then it may be applied to trios of neurons, etc. After the first layer has been exhaustively tested, the second layer may be tested, etc. depending on the number of layers. Once completed, the previous training patterns must be re-applied in random order to prevent oscillations. Although a painstaking process in comparison with backpropagation, it is nevertheless readily implemented in analogue electronics. These suffer from nonlinear distortions which motivated the MR III algorithm which does not require the derivative of the error

function. The weight update is given by a least square algorithm based on mean square error as the performance index:

$$w_{i+1} = w_i + \Delta w_i = w_i - \eta\left(\frac{\Delta e_i^2}{\Delta s}\right) = w_i - 2\eta e_i\left(\frac{\partial e_i}{\partial s}\right) = w_i + 2\eta e_i\left(\frac{\partial y_i}{\partial s_i}\right) \quad (1)$$

where $\Delta e_i = -\Delta y_i$, $e_i = y_i^d - y_i$ = output error. This is the Widrow-Hoff rule which may be simplified to eliminate the need for linear multipliers at the cost of slower convergence:

$$w_{i+1} = w_i + 2\eta sgn(e_i)sgn(\nabla y_i) \quad (2)$$

In this case, DC offsets may appear which require adaptive cancellation using a number of analogue approaches at the cost of physical footprint [20]. An analogue implementation of neural learning through gradient descent was presented composed of integrators, summers, comparators and multipliers [21]. A simple Hebbian learning rule has been implemented as voltage activations on a series of circuits – V_{ij} multiplier - $w_{ij}V_j$ multiplier - summer - V_i neuron multiplier circuits [22]:

$$V_i = f\left(\sum_j w_{ij}V_j\right) \quad (3)$$

$$\frac{dw_{ij}}{dt} \propto \frac{dV_{ij}}{dt} = aV_iV_j - bw_{ij} \quad (4)$$

where k, a, b = small constants, f(.) = sigmoidal activation function, V_i = voltage on post-synaptic neuron i, V_j = voltage on pre-synaptic neurons j synapsing on neuron i. The second term represents weight decay that may be set to zero as b = 0. Wide-range Gilbert multipliers output a current that is proportional to the product of the input voltages: $I_0 = a\left(\frac{V_1 - V_2}{V_3 - V_4}\right)$. The summer outputs a current that is proportional to the sum of inputs based on Kirchoff's laws. The synaptic weight w_{ij} may be represented as a voltage V_{ij} on a capacitor: $V_{ij} = \frac{I_c}{C} \propto V_iV_j$. . The non-linear function f(.) exploits the saturation behaviour of the neuron multiplier. An analogue circuit block that implements backpropagation without clock synchronisation has been devised in which synaptic weights are stored as voltages in capacitors $V_w(t) = V_w(0) + \frac{1}{C}\int_o^T I(t)dt = V_w(0) - \frac{K}{C}\int_o^T \frac{\partial E(t)}{\partial V_w(t)}dt$ where K = conductance coefficient, the charging of which imposes time delays [23]. Learning was implemented with analogue circuit feedback. Analogue multiple voltage input lines with sample-and-hold circuits may implement input voltages as modifiable synaptic weights which may be adjusted more easily than resistance weights [24]. During learning, input voltage is varied but during operations, connection weights are held in the sample-and sold circuit. Online backpropagation offers the potential for continuous online learning [25]. An electrical circuit representation of the more complex backpropagation learning algorithm has been derived [26]. The backpropagation algorithm of the multilayer perceptron adjusts its weights (conductances) by gradient descent:

$$w_{ih}(t+1) = w_{ij}(t) - \eta\frac{\partial e}{\partial w_{ij}} \quad (5)$$

where $e = \frac{1}{2}(y^d - y)^2$, $\frac{\partial e}{\partial w_{ij}} = \frac{\partial e}{\partial y}\frac{\partial y}{\partial w_{ij}} = (y^d - y)\frac{\partial t}{\partial w_{ij}}$, , f'(.).) = sigmoid function, y = network output, V_i^k = neuron voltage output from ith neuron of kth layer. The output sensitivity of the multilayer perceptron is given by:

$$\frac{\partial y}{\partial w_{ij}(k)} = V_i^{k-1}\Delta V_j^k = V_i^{k-1} f'(V_j^k) I_j^k = V_i^{k-1} f'(V_j^k) \sum\nolimits_m w_{mj}^{k_1} y_m^{k+1}(1 - y_m^{k+1}) \quad (6)$$

The incremental connection weight may be represented by:

$$\Delta w_{ij}(t) = -\eta \frac{\partial e}{\partial w_{ij}(t)} \quad (7)$$

where η = RC = settling time constant. The implementation of the backpropagation algorithm on physical neural networks offers both speed of learning and high energy efficiencies [27]. To avoid complex computations of derivative of the error function required of the delta rule, the Madaline III rule implements gradient estimation based on node perturbation:

$$\Delta w_{ij} = \frac{\partial E}{\partial \mathrm{f}\left(\sum_j w_{ij} x_j\right)} x_j \quad (8)$$

where f(.) = nonlinear squashing function. This requires wire routing to each neuron, multiplication hardware and addressing logic. Rather than node perturbation, weight perturbation may be employed with less hardware. Analogue VLSI implementation of feedforward and recurrent neural networks may implement on-chip learning through gradient estimation through finite differences as an approximation to backpropagation [28]. The weight update rule involves a constant weight perturbation $\mathrm{pert_{ij}}$ and is given by the finite difference which can be mapped onto an analogue implementation:

$$\Delta w_{ij} = \frac{E(w_{ij} + pert_{ij}) - E(w_{ij})}{pert_{ij}} \quad (9)$$

where E = total mean square error. Analogue implementation measures the errors with unperturbed and perturbed weights, subtracts them and multiplies by a constant. No backpropagation flow is necessary. A similar version applies simultaneous weight perturbations to compute central difference approximations to the differential of the error in parallel [29]:

$$\Delta w_{ij} = \frac{E\left(w_{ij} + \frac{1}{2}pert_{ij}\right) - E\left(w_{ij} - \frac{1}{2}pert_{ij}\right)}{pert_{ij}} \quad (10)$$

The learning rule thence becomes: $w_{ij}(t+1) = w_{ij}(t) - \eta \Delta w_{ij}$ (11)

Our goal is to explore analogue neural networks with a focus on neural learning to determine its plausibility as an approach to artificial intelligence on the Moon.

5 Analogue Neural Network Circuit Simulations

Neural networks comprise of an input layer of input data to be processed, hidden layer(s) responsible for extracting information from the input data and an output layer outputting the desired response. Fixed weight neural networks are inflexible due to the inability to be trained for different tasks as the neuron weights implemented as resistors remain fixed. We have examined two approaches to analogue neural networks with learning circuitry. The first was a simulated approach and the second was a simulated and physical construction approach. In both cases, we implemented two analogue circuit modules – a forward network that propagates signals from the input to the output layer and a backpropagation circuit that propagates the error backwards through the network from the output to compute the weight changes.

In our first simulated analogue neural network design using LTSPICE [30], we adopted a two input neuron – three hidden neuron – one output neuron configuration for the forward circuit. The forward network comprised several subcircuits – an op-amp-based summation circuit, an activation function circuit, a voltage-controlled resistor, an op-amp multiplier and a general inverter op-amp circuit. The neuron itself is constructed from a neuron summation sub-circuit and an activation function sub-circuit. The combination of weighted resistors is used to provide summed outputs. The summer amplifier is well understood (see Fig. 6(b) for an example) such as that adopted in the Yamashita-Nakamura neuron model [31]. Voltage-controlled resistors (VCR) were adopted to implement synaptic weights similar to [32] – they were based on a surgeless electronic variable resistor and attenuator design [33]. The VCR design was based on two vacuum tubes with cross-connected cathodes and grids to operate as a voltage divider (Fig. 2(a)). The weights represented by the VCR are updated from the backpropagation network. Every neuron performs a set of operations for which a decision is made whether to fire a neuron or not based on the activation function. The activation function typically applies a nonlinear squashing function to the weighted summation. We explored two activation functions – an op-amp-based linear activation function (Fig. 2(b)) and a vacuum tube-based differential amplifier (Fig. 2(c)) [34]. The threshold may be adjusted through the activation function but use of resistors requires positive weights. Inhibitory links may be implemented through the activation function as negative state [35]. An analogue op-amp multiplier exploits the nonlinear properties of class AB op-amps using three interconnected op-amps with a collection of resistors to output an approximation of the product of two inputs [36] (Fig. 3). There is a vacuum tube amplifier implementation with class-AB operation characteristics [37].

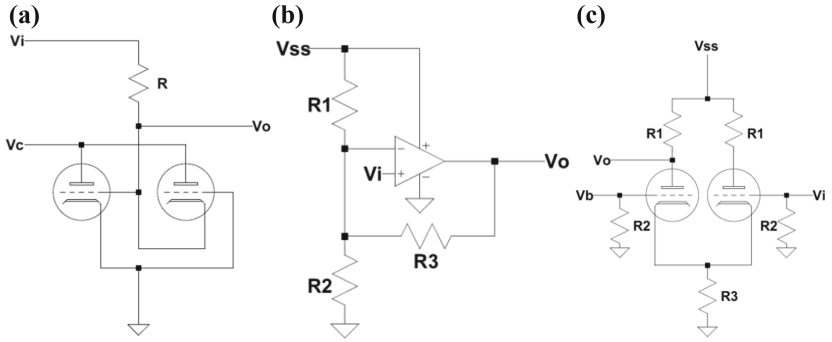

Fig. 2. (a) Voltage-controlled resistor; (b) linear activation function; (c) vacuum tube-based differential amplifier.

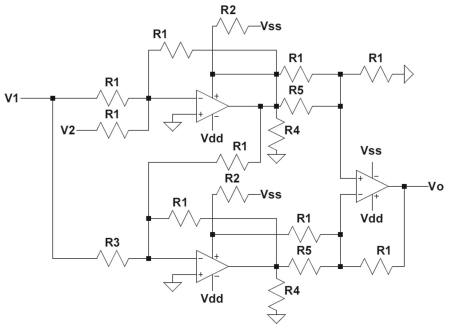

Fig. 3. Op-amp multiplier.

The backpropagation circuit comprised three sub-circuits: (i) a subtraction sub-circuit to calculate the error between the forward network output and the target value, (ii) VCRs and (iii) the back propagation multiplication block that calculates the neuron errors. The subtraction sub-circuit comprised an op-amp inverting amplifier to subtract voltages as well as for adding voltage biases (Fig. 4).

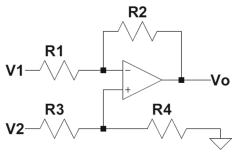

Fig. 4. Subtracting op-amp inverting amplifier.

The backpropagation multiplication sub-circuit contains a subtraction op-amp and cascaded op-amp multipliers. The resultant error outputs of the backpropagation circuit are multiplied with the neuron outputs using op-amp multiplication circuits. Each of the sub-circuits were functionally validated but once assembled into the neural network, the

training simulations failed to update the weights of our vacuum tube-based analogue neural network successfully. We suspected the problem may be associated with the VCRs.

6 Analogue Neural Network Circuit Hardware

We built a different trainable analogue neural network circuit to demonstrate the principle of online training on a rover vehicle [38]. The forward interconnection of neurons were arranged into successive layers – the analogue neural network configuration comprised two input neurons plus a bias neuron followed by a two neuron hidden layer plus a bias neuron to a two neuron output layer. As before, the neural network is primarily constructed using (transistor-based) op-amps. The circuits were also simulated using the Proteus PCB simulator software generating two outputs to our network with weight values changed from output error to generate new outputs. Rather than VCRs for the adjustable network weights, we adopted variable potentiometers which were updated according to the voltage outputs from the backpropagation circuit during training. As before, the analogue circuit implementation of the neuron requires a weighted summation and a multiplication. Multiplication of the input and the weight was implemented by a four quadrant translinear multiplier (AD534) based on the Gilbert cell. The inputs to the multiplier can either be positive, negative or both yielding a combination of the signed inputs. During the training phase, voltage *Vx* was varied using a potentiometer to update and obtain the optimum weight yielding the output *Vout*. The weighted summation of inputs to a neuron was followed a linear threshold activation function. It was executed at the end of the forward pass during the training of data rather than at each neuron. The forward circuit of the analogue neural network is prototype of the implementation of this arrangement is shown in Fig. 5.

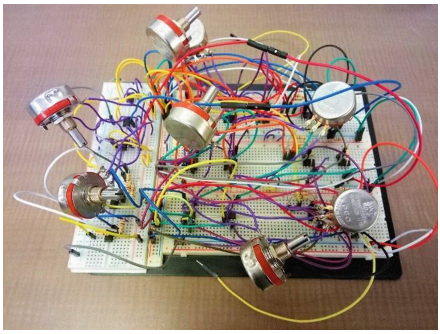

Fig. 5. Forward propagation neural network circuit.

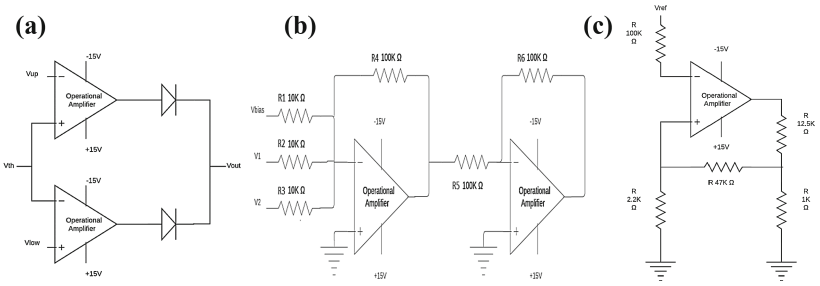

Fig. 6. (a) Window comparator; (b) summer amplifier (c) voltage comparator circuits.

The backpropagation circuit required several subcircuits:

(i) Window comparator (Fig. 6(a)) to detect input voltage range – two voltage comparators provide upper (5 V) and lower (2 V) reference voltages. The output fires if the input is outside this range but not of it is within the range, i.e. a bandstop filter. A voltage inverter inverts this to give an output of 5 V within the range and < 1 V outside the range.

(II) Summation circuit (Fig. 6(b)) to output subtraction of the two inputs (either 5 V or <1 V) and the output of the window comparator.

(III) (iii) Voltage comparator (Fig. 6(c)) outputs (-1 V/$+1$ V) the input summation which alternates between the negative and positive saturation voltage based on the voltage at the non-inverting input of the op-amp.

Smaller learning rate ensures better stability over convergence speed but our data set was simple so it was set at 0.1 V. The error output was defined thus:

Error $= -1$ when the desired output was less than 2 V but the actual output was greater than 2 V

$= 1$ when the desired output was greater than 2 V but actual output was less than 2 V.

$= 0$ for zero error

The errors were output as voltage values of -1 V and $+1$ V. The backpropagation circuit is shown in Fig. 7.

The forward and backpropagation circuits were mounted on a small rover over an obstacle course. The rover was fitted with two front corner digital IR sensors to detect obstacles within 9 cm connected directly to the input nodes of the forward neural network through a 12-bit DAC converter MCP4725. An Arduino UNO board controlled four servo motors for the four wheels, two on each side. Two voltage inputs to the neural network are from the distance sensors that detect obstacles and from a bias node while the two-node output layer is connected to the respective wheel motors of the rover on either side. A simple dataset was used for training the network: zero obstacle path yielded inputs of 1 V but a left/right obstacle yielded an input of 4 V. A total motor output of 2 V yielded all-stop while under 2 V yielded appropriate turns and forward movement. During the

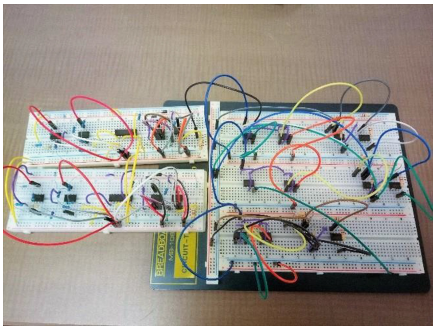

Fig. 7. Backpropagation circuit.

training phase, the circuit performing forward propagation was initiated with random voltage weights to map distance sensor inputs to the output – this is subsequently fed to the backpropagation circuitry comprising a threshold activation sub-circuit as well as multipliers and summers. Over multiple iterations, the network successfully converge to weight values that minimised the error from the desired output. The potentiometers were varied according to the output over several iterations to drive robot to realise a BV2B behaviour [39]. The robot successfully demonstrated the desired obstacle avoidance behaviour through a path with several obstacles – a left obstacle stopped the right wheels while driving the left wheels, and vice versa.

7 Conclusions

We have implemented a backpropagation algorithm in analogue circuitry demonstrating that electronic hardware is malleable and therefore carries the potential to be used for computational purposes as demonstrated on a simple rover. The neural network received training through the analogue circuit without any software programs. However, there are several issues that deserve further investigation. Although the training of the neural net is implemented through the backpropagation circuitry, the weight updates were applied manually which would preferably be implemented automatically clearly demonstrating learning directly from the physical environment. The crux is to implement weight memory updates directly electronically. There are several options for non-volatile memory cells including resistive RAM and magnetoresistive RAM in which each memory cell encodes a synaptic weight as analogue conductance. Non-volatile memories may be manufactured through microtechnology as magnetoresistive RAM comprising current-controlled magnetic tunnel junctions collectively implementing resistance summation [40]. This offers much lower energy consumption. Hardware implementation of neural networks is of interest as an application of the electrical memory element, the memristor for its ability to regulate current flow within itself, while simultaneously retaining the memory of its previous state without electrical power [41–43], including CMOS-based memristors [44–46]. A ferroelectric memristor with a voltage-controlled resistance employed as a variable synaptic weight suffer from conductance hysteresis from historical voltages [47]. However, memristor micro-manufacturing (as for solid-state) does

not appear feasible for the near future. Future work should involve training the network of neural net with more complex tasks including sensing in its immediate domain and with predicted expectations. This introduces the issue of analogue neural network scalability. Although a neural design by trial-and-error was implemented because of the circuit's simplicity, genetic algorithms offer the possibility of evolving neural network designs for more complex problems offline. However, incremental adaptation rather than global optimization offers ease of implementation and time-efficiency [48]. Such learning methods have yet to be tested, but they can plausibly be implemented through electronic hardware. Finally, the internal circuitry of the op-amps consisted of transistor configurations which should be replaced with vacuum tube-based circuitry to enhance its relevancy to manufacturability on the Moon.

References

1. Ellery, A.: Sustainable in-situ resource utilisation on the Moon. Planet. Space Sci. **184**(4), 104870 (2020)
2. Ellery, A.: Are there biomimetic lessons from genetic regulatory networks for developing a lunar industrial ecology? Biomimetics J **6**(3), 50 (2021)
3. Ellery, A., Mellor, I., Wanjara, P., Conti, M.: Metalysis FFC process as a strategic lunar in-situ resource utilisation technology. New Space J **10**(2), 224–238 (2022)
4. Ellery, A.: Generating and storing power on the Moon using in-situ resources. Proc. IMechE J. Aerosp. Eng **236**(6), 1045–1063 (2021)
5. Ellery, A.: Leveraging in-situ resources for lunar base construction. Can. J. Civ. Eng. **49**(5), 657–674 (2022)
6. Ellery, A.: Is electronics fabrication feasible on the Moon? In: Proceedings ASCE Earth & Space Conference Colorado School of Mines, Denver (2022)
7. Ellery, A.: Universal construction based on 3D printing electric motors: steps towards self-replicating robots to transform space exploration. In: IEEE International Symposium Robotics & Intelligent Sensors (IRIS), pp. 81–85. Ottawa, Canada (2017)
8. Lagsford, W., Ghassaei, A., Gershenfeld, N.: Automated assembly of electronic digital materials. In: Proceedings of the Manufacturing Science and Engineering Conference, paper no. MSEC2016-8627 (2016)
9. Parberry, I.: Circuit Complexity and Neural Networks. MIT Press Foundations of Computing, Cambridge, MA (1994)
10. Hopfield, J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc Natl. Acad. Sci. **81**, 3088–3092 (1984)
11. Siegelmann, H., Sontag, E.: On the computational power of neural nets. J. Comput. Syst. Sci. **50**, 132–150 (1995)
12. Siegelmann, H., Margenstern, M.: Nine switch-affine neurons suffice for Turing universality. Neural Netw. **12**, 593–600 (1999)
13. Sun, G.-Z., Chen, H.-H., Lee, Y.-C., Giles, C.: Turing equivalence of neural networks with second order connection weights. Proc. Int. Joint Conf. Neural Networks **2**, 357–362 (1991)
14. Graves, A., Wayne, G., Danihelka, I.: Neural Turing machines. arXiv-1410.5401 (2014)
15. Hahnloser, R., Sarpeshkar, R., Mahowald, M., Douglas, R., Seung, S.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405**, 947–951 (2000)
16. Roy, K., Jaiswai, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**, 607–617 (2019)
17. Mehonic, A., Kenyon, A.J.: Brain-inspired computing needs a master plan. Nature **604**, 255–260 (2022)

18. Burr, G.W., Sebastian, A., Ando, T., Haensch, W.: Ohm's Law + Kirchhoff's Current Law = Better AI: neural-network processing done in memory with analog circuits will save energy. In: IEEE Spectrum, vol. 58, no. 12, pp. 44–49 (2021)
19. Winter, R., Widrow, B.: MADALINE RULE II: a training algorithm for neural networks. In: IEEE 1988 International Conference on Neural Networks, vol. 1, pp. 401–408 (1988)
20. Carusone, A., Johns, D.: Analogue adaptive filters: past and present. IEE Proc. Circuits Devices Syst. **147**(1), 82–90 (2000)
21. Wang, T., Zhuang, X., Xing, X., Xiao, X.: Neuron-weighted learning algorithm and its hardware implementation in associative memories. IEEE Trans. Comput. **42**(5), 636–640 (1993)
22. Schneider, C., Card, H.: CMOS implementation of analog Hebbian synaptic learning circuits. In: IJCNN-91-Seattle International Joint Conference on Neural Networks, vol. 1, pp. 437–442 (1991)
23. Paulu, F., Hospodka, J.: Design of fully analogue artificial neural network with learning based on backpropagation. Radioengineering **30**(2), 357–363 (2021)
24. Kawaguchi, M., Ishii, N., Umeno, M.: Analogue neural circuit and hardware design of deep learning model. Procedia Comput. Sci. **60**, 976–985 (2015)
25. Wang, Y., Lee, D.: Online backpropagation learning for a human-following mobile robot. Preprint (2007)
26. Martinelli, G., Perfetti, R.: Circuit theoretic approach to the backpropagation learning algorithm. IEEE Int. Symp. Circuits Syst. **3**, 1481–1484 (1991)
27. Wright, L., et al.: Deep physical neural networks trained with backpropagation. Nature **601**, 549–555 (2022)
28. Jabri, M., Flower, B.: Weight perturbation: an optimal architecture and learning technique for analogue VLSI feedforward and recurrent multilayer networks. IEEE Trans. Neural Netw. **3**(1), 154–157 (1992)
29. Maeda, Y., Hiano, H., Kanata, Y.: Learning rule of neural networks via simultaneous perturbation and its hardware implementation. Neural Netw. **8**(2), 251–259 (1995)
30. Larson, S., Ellery, A.: Trainable analogue neural network with application to lunar in-situ resource utilization. In: Proceedings of the International Astronautical Congress, Jerusalem, IAC-15-D3.3.6 (2015)
31. Yamashita, Y., Nakamura, Y.: Neuron circuit model with smooth nonlinear output function. In: Proceedings of the International Symposium Nonlinear Theory & its Applications, Vancouver, pp. 11–14 (2007)
32. Martinelli, G., Perfetti, R.: Circuit theoretic approach to the backpropagation learning algorithm. In: IEEE Symposium on Circuits and Systems, vol. 3, pp. 1481–1484 (1991)
33. Meier, E.: Surgeless electronic variable resistor and attenuator. U.S. Patent 2 726 290 (1955)
34. Gray, T.: Direct-Coupled Amplifiers Applied Electronics, 2nd edn., pp. 499–508. John Wiley & Sons Inc, New York (1954)
35. Bradley, W., Mears, R.: Backpropagation learning using positive weights for multilayer opto-electronic neural networks. In: IEEE Lasers and Electro-Optics Society Annual Meeting, pp. 294–295 (1996)
36. Riewruja, V., Rerkratn, A.: Analog multiplier using operational amplifiers. Indian J. Pure Appl. Phys. **48**, 67–70 (2010)
37. Gray, T.: Amplifiers with operation extending beyond the linear range of the tube characteristic curves: class AB, class B and class C amplifiers Applied Electronics, 2nd edn., pp. 609–652. John Wiley and Sons Inc., New York (1954)
38. Prasad V, Ellery A (2020) "Analogue neural network architecture for in-situ resourced computing hardware on the Moon" *Proc Int Symp Artificial Intelligence, Robotics and Automation in Space* (iSAIRAS), paper no 5005

39. Braitenburg, V.: Vehicles: Experiments in Synthetic Psychology. MIT Press (1984)
40. Jung, S., et al.: Crossbar array of magnetoresistive memory devices for in-memory computing. Nature **601**, 211–217 (2022)
41. Chua, L.: Memristor: missing circuit element. IEEE Trans Circuit Theory **18**, 507–519 (1971)
42. Zhao, Y, Shi, G.: Circuit implementation method for memristor crossbar with on-chip training. In: IEEE Asia Pacific Conference on Circuits and Systems (2018)
43. Thomas, A.: Memristor based neural networks. J. Phys. D: Appl. Phys. **46**, 093001 (2013)
44. Ebong, I., Mazumder, P.: CMOS and memristor-based neural network design for position detection. Proc IEEE **100**(6), 2050–2060 (2012)
45. Larras, B., Chollet, P., Lahuec, C., Seguin, F., Arzel, M.: Fully flexible circuit implementation of clique-based neural networks in 65-nm CMOS. IEEE Trans. Circ. Syst. **I**, 1–12 (2018)
46. Yeo, I., Chu, M., Lee, B.-G.: A power and area efficient cmos stochastic neuron for neural networks employing resistive crossbar array. IEEE Trans. Biomed. Circuits Syst. **13**(6), 1678–1689 (2019). https://doi.org/10.1109/TBCAS.2019.2945559
47. Ueda, M., Nishitani, Y., Kaneko, Y., Omote, A.: Backpropagation operation for analogue neural network hardware with synapse components having hysteresis characteristics. PLoS ONE **9**(11), e112659 (2014)
48. Harvey, I.: Cognition is not computation; evolution is not optimisation. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) Artificial Neural Networks — ICANN'97. LNCS, vol. 1327, pp. 685–690. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0020233