AN ANALOGUE NEURAL NETWORK ARCHITECTURE FOR IN-SITU RESOURCED COMPUTING HARDWARE ON THE MOON

Virtual Conference 19–23 October 2020

Vatsala Prasad[1], Alex Ellery[2]

[1]*now at Manipal Institute of Technology, MAHE, Manipal, India, E-mail: vatsala.prasad1998@gmail.com*
[2]*Carleton University, Mechanical and Aerospace, Ottawa, Canada, E-mail: AlexEllery@cunet.carleton.ca*

ABSTRACT

Future endeavours on Moon exploration would require exploitation of in-situ resources available on the lunar surface because of the high cost involved in the transportation of materials from Earth. The materials extracted from the available compounds on the Moon could be provided to a machine that has the capability of self-replication. This paper presents a trainable analogue neural network hardware that can act as the controlling unit of such a machine and demonstrates the possibility that on being trained and fed with the appropriate data and raw materials, the machine would possess the capability of printing any mechanical or electronics parts thereby exhibiting the properties of a universal constructor.

1 INTRODUCTION

In-situ resource utilization (ISRU) of the lunar surface involves the process of generating supplies using raw materials to ensure sustenance of humans, with the ongoing research concentrated on recovering water ice at the polar regions ostensibly to support a human presence [1,2].

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 colonization as independent of Earth as is feasible, i.e. to build an entire infrastructure from lunar resources. Our research focuses on the prospect of exploitation of accessible lunar raw materials for the purpose of complete lunar colonization, with negligible support from the resources of Earth. This would entail the creation of not just products for human use, but the construction of the machines involved in the process of manufacturing these products as well. A crucial backbone to this notion is the ability to construct machines of production rather than products from lunar resources. A step forward in this direction has been initiated by A. Ellery [3], who has demonstrated the possibility of realizing the concept of self-replication through a universal constructor. For a self-replicating machine, the product is indeed the machine of production. This idea has been spurred by the progress made in the field of 3D printing in recent times. This universal constructor carries the potential to manufacture fully functional parts, when provided with the set of necessary commands and raw materials, which in the current setting, would be the in-situ resources obtained from the lunar surface. In essence, this machine would be a programmed 3D printer. The validation to this theory would be acquired through the production of crucial mechanical parts such as actuators leading one to conclude that the physical assembly of the universal constructor is plausible. Further evidence in this context is the world's first partial self-replicating manufacturing device, the RepRap [4]. This machine, which started the revolution in the world of three-dimensional printing, possesses the capability of creating valuable parts with plastic as its raw material as well as 3-D print some of its own constituent components. The core idea therefore of producing any machine lies in assembling a set of strategically placed electric motors that run on a system which is capable of universal computation. Any kinematic machine – be it load-haul-dump vehicle, milling tool, 3D printer or assembly manipulator - comprises a specific configuration of electric motors supported by a control system capable of universal computation. It is the latter that is of concern here.

We wish 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. At first blush, it may appear that the production of semiconductor transistors is an achievable feat since the materials required for the process are accessible on the Moon using relatively simple (electro) chemical processing, that is, aluminum metal from anorthite for conducting electrodes, tracks and p-type dopant; silicon from silicate minerals for bulk semiconductors; and silica and alumina ceramics from silicates and anorthite for insulation layers. Dopants require a more complex and challenging chemical processing strategy for dealing with KREEP minerals to recover n-type

dopant phosphorus [5]. However, the implementation of solid state computers would be impossible from lunar resources due to: (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; (iv) extreme size and cost of electronics foundries. Another important consideration to bear in mind is the impossibility of transporting the entire supply chain units for this process from the Earth to the Moon, owing to its complexity and involved extravagant cost. Hence, the supply chain for the manufacturing process must be maximally lunar dependent, and the computational units, too, should be reliant on lunar resources. Vacuum tube technology is highly versatile that pre-dates solid-state technology. Vacuum tube technology seems the most desirable alternative as the materials to assemble one are readily available and relatively small in number - tungsten from NiFe meteorites for cathodes, calcium oxide from anorthite for cathode coatings, nickel from NiFe meteorites for anodes and gate electrodes, kovar from NiFe meteorites for high temperature wiring, and fused silica glass from silicates for vacuum tube enclosures to protect against dust. The problem with vacuum tubes is that there are limits to their miniaturization. The adoption of traditional CPU-based computational architectures with vacuum tubes will yield large-scale computers the size of a warehouse. A substitute to CPU-based computational architectures which presents the disadvantage of size, is the use of neural net architectures. Whereas computational footprint grows exponentially with CPU-based architectures with task complexity, it grows linearly with neural net architectures [6].

Neural networks have proved to be significantly more efficient than other computational algorithms developed to date, having the ability to adapt based on the requirement for which it is trained and implementing logic programs [7]. The recent trend is to examine the feasibility of having a hardware implementation of neural networks. Some of this research has been directed at studying the use of another crucial electrical memory element, the memristor [8,9,10]. The striking feature of this component is its ability to regulate current flow within itself, while simultaneously retaining the memory of its previous state without electrical power. CMOS-based architectures have also been explored in this context [11,12,13]. Additionally, feasibility of the implementation of neural networks [14] on Field Programmable Gated Arrays for real time applications [15] have also been studied.

Besides the traditional ability to act as control units, the use of analogue circuits primarily based on operational amplifiers to accomplish computations has also been examined. One of them is the Yamashita-Nakamura neuron model [16] which demonstrates how an operational amplifier could be modelled to perform the computations within a neuron. The proposed model is constructed using resistors, diodes and capacitors and exhibits a non-linear output function. However, the limitation to this model lies in its inability to be trained for different tasks, as the neuron weights implemented as resistors remain fixed. Additionally, the simulation results of a vacuum tube based neural network model demonstrated a backpropagation algorithm as well. The reported results claimed to be unable to efficiently replicate the process of learning [17].

This paper introduces a physical demonstration of an analogue neural net, a computational core, used to maneuver a rover. Furthermore, we have implemented a backpropogation algorithm in analogue circuitry, demonstrating that electronic hardware is malleable and therefore carries the potential to be used for computational purposes. The idea is to provide the rover with navigating capabilities such that it can demonstrate obstacle avoidance, through the training it receives by the analogue circuit, without any software programs. The neural net has a multi-layer structure and is primarily constructed using operational-amplifiers. The network's weights exhibit the property of being adaptable using potentiometers instead of fixed resistors as weights. The use of potentiometers as weight inputs in the form of voltages to the network aid the process of training, such that over multiple iterations and updates, the network can arrive at converged weight values that provides minimum error to the desired output. During the training phase, the circuit performing forward propagation is initiated with random voltage weights to adapt to the distance measuring sensor inputs fed to the circuit. The output is subsequently fed to the backpropagation circuitry comprising of a threshold activation sub-circuit as well as multipliers and summers. The potentiometers are varied according to this output thereby completing the process of training over the course of several iterations. The output obtained over several iterations is used to direct the robot, thereby helping it to demonstrate self-guidance inspired by the Braitenburg Vehicle 2B arrangement [18]. The organization of the work is as follows: Section 2 describes the architecture of the neural net, followed by the design and implementation of the circuit in section 3, 4 and 5. Next, the rover design is presented in section 6. Finally, the observed results are noted in section 7.

2 ARCHITECTURE

The architecture of a neural network is a particularly important aspect of the design process for it plays a central role in defining the working of the system. Neural networks comprise of an input layer which receives the input to be processed, the hidden layer(s) which is responsible for extracting relevant information from the input data and finally, the output layer which outputs the desired response.

Within our multilayer perceptron model, the inputs consisted of voltages received from the distance-based sensors attached to the rover based on whether an obstacle was near and a bias node. Next, there was a hidden layer with two nodes and a bias node, following which is an output layer with two output nodes. Each node of the output was attached to the wheels of the rover on either side guiding its movements. The weight updates of the network were achieved through the use of potentiometers which were varied in accordance to the output received from the backpropagation circuit. Fig. 1 depicts a representation of the architecture. The dataset which was used for training the network is listed in Tab. 1. A clear path should input a value of 1V and the indication of an obstacle in the rover's track would provide the value of 4V. A combination of these instances was used to train the network with the output values being greater than 2V to make the attached set of wheels to stop and less than 2V to continue moving forward.

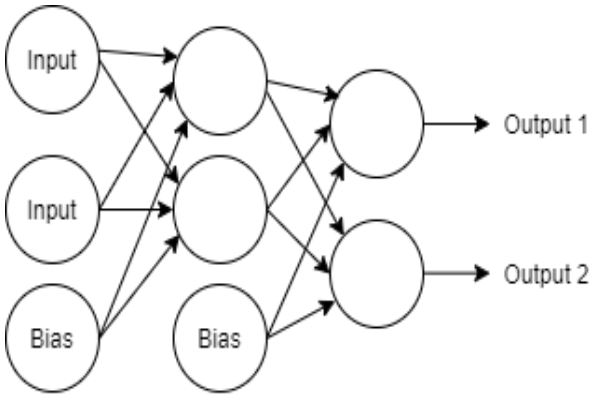

Figure 1: Architecture of Neural Net

Table 1: Dataset for Training Neural Net

$Input_1$ (V)	$Input_2$ (V)	$Output_1$	$Output_2$
1	1	0	0
4	4	1	1
1	4	0	1
4	1	1	0

3 DESIGN

At its very core, the computational ability of a neural network is based on a set of highly structured mathematical equations which process the dynamic inputs received through the input layer to obtain an output at the output layer. The two most crucial operations that work within these equations is summation and multiplication. It was thus necessary to have mechanisms within our proposed model that can perform the task of addition and multiplication.

One of the chief advantage of operational-amplifiers lies in its ability to be modelled as different computational units that can perform mathematical functionalities when used in combination with passive electrical components. The specifications of these components, particularly the resistor, is determined and can be varied according to needs. The following computational units were used throughout the circuitry of forward as well as the backward propagation as required.

3.1 Multiplication Unit

In order to perform the multiplication between the input received and the neuron weight, a four quadrant translinear multiplier, AD534, based on the gilbert cell was used. The inputs to the four quadrant multiplier in this case, can either be positive, negative or both. This is necessary in the implementation of the weight adjustable neural network as weights as well as inputs to the neuron unit can either be negative or positive. Therefore, the expectation was to have a functional unit which can accommodate the results of the combination of these signed inputs. The inputs to the multiplier V_x and V_y, are depicted in the Fig 2. During the network training phase, voltage V_x was varied using a potentiometer to update and obtain the optimum weight. The multiplied output V_{out} is expressed as:

$$(V_x * V_y)/V_{ref} = V_{out} \quad (1)$$

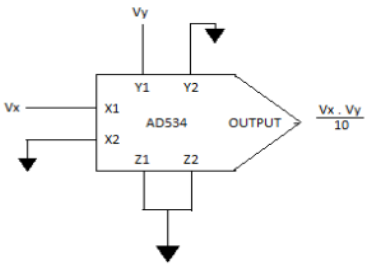

Figure 2: Translinear Multiplier

3.2 Summation Units

The summation operation requires a fairly uncomplicated implementation when an operational amplifier is used. The combination of resistors is used to provide scaled outputs and in our case, was chosen in a manner such that it compensated for the multiplication factor that is introduced through the multiplication unit. It can be mathematically expressed as:

$$R_4\left((V_{bias}/R_1) + (V_1/R_2) + (V_2/R_3)\right) = -V_{out} \qquad (2)$$

where V_{out} is the output, R_x are resistor values and V_{bias}, V_1 and V_2 are inputs.

Summation units provide outputs that are phase inverted with the expected result. Here, the second operational amplifier was applied so as to obtain the same phase at the output as anticipated. The resistors provide no scaling in this par. The implemented circuitry is represented in Fig. 3.

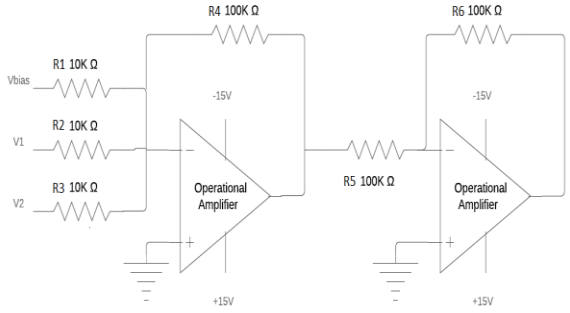

Figure 3: The Summation Circuit

4 FORWARD PROPAGATION CIRCUIT

The interconnection of neurons arranged within the network communicate information through the successive layers. Every neuron, on receiving an input from a previous layer, performs a set of operations on it following which a decision is made whether the neuron "fires" or not. The sequential functions that occur within each neuron is as follows:

- Pre Activation: It involves the summation of the different inputs to a neuron after each one of them individually undergoes a linear transformation. The computational formula of this is therefore described as:

$$X_0W_0 + X_1W_1 + .. = N_{out} \qquad (3)$$

  where N_{out} is the neuron output, X_i and W_i are the inputs and associated weights respectively.
- Activation: The primary purpose of an activation function remains that of adding non-linearity to the summation output obtained in pre-activation. The application of the activation function can be represented according to [19]:

$$F_{out} = F(N_{out}) \qquad (4)$$

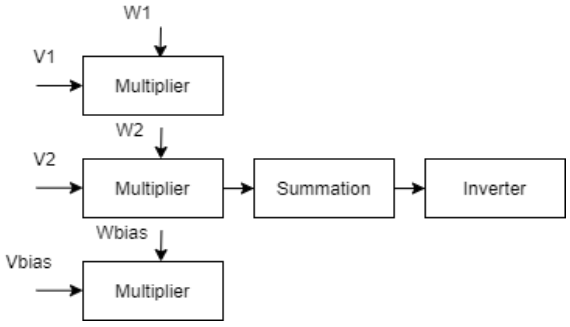

Figure 4: Neuron in the Forward Circuit

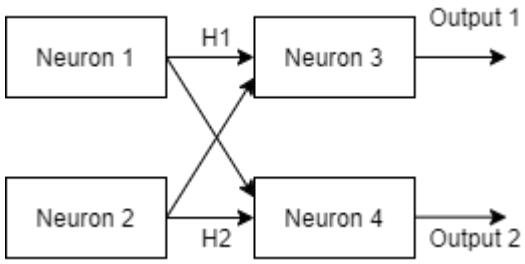

Figure 5: Neuron Arrangement on Circuit

The activation function used in our neural net model was a linear threshold activation. Instead of performing the activation step at each neuron, it was executed at the end of the forward pass during the training of data; on the circuit implementation of forward propagation, the need was to only carry out the pre activation which involved an arrangement of a multiplier which performs the stated operation on each input and its associated updatable weight, following which a summation is performed. A neuron thus can be represented as in Fig. 4. Fig. 5 shows the schematic of the neuron arrangement on the circuit. Here, H_i is the hidden layer output, and $Output_i$ is the final output. The prototype of the implementation of this arrangement is shown in Fig. 6.

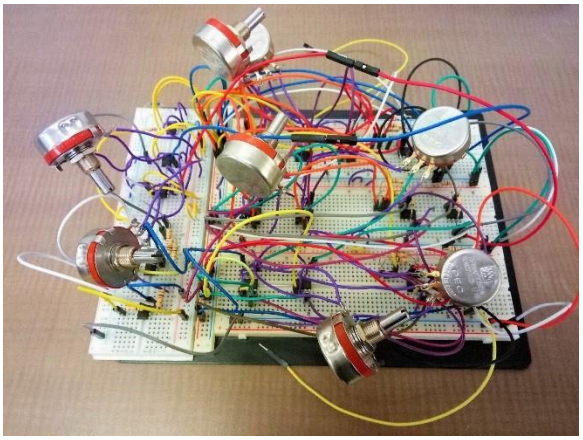

Figure 6: Prototype of Forward Propagation Circuit

5 BACKPROPAGATION CIRCUIT

The training of the neural net began by initializing the neurons weights with random values. The weight update was performed over each iteration until the output of the neural network was as desired – thereby minimizing error rates. The weight update equation is given as:

$$W_{ij} + \alpha * N_i * (o_d - o_{ob}) = W_{ij}(n+1) \quad (5)$$

where W_{ij}, α, N_i, O_{ob} and O_d are the weights, learning rate, neuron input, obtained output and the desired output respectively.

The choice of learning rate may vary from case to case; however, a smaller learning rate ensures better relibility although the process of optimization is slow. In our case, however, since our data set was simple, the choice of learning rate was quite straightforward. It was taken to be 0.1 (as a voltage) because this factor is already accounted for in the multiplier AD534.

The activation function chosen for our network was a threshold activation function for the purpose of ensuring stability in the circuit. The expected errors were as follows, $O_{desired} - O_{obtained}$:

- was -1 when the desired output was less than 2V, but the obtained output was greater than 2V
- was 1 when the desired output was greater than 2V, but the obtained output was less than 2V
- was 0 implying zero error

To obtain these values on the circuit in the form of voltage values of (-1V) and (+1V), the following subcircuits were used:

- Window Comparator: To detect whether an input was within a given range of voltage, a window comparator was as depicted in Fig. 7. It comprised of two voltage comparators which provided the upper and the lower reference voltages. The output was turned on if the input to the circuit was beyond this defined range and it is turned off when it was within the limits of the two reference voltages. For our application, the upper threshold voltage was chosen to be 5V and lower threshold was chosen to be 2V. This output was reversed using a voltage inverter thereby giving output value of 5V when the input was within the range and <1V when the output was beyond the range.
- Summation Circuit: A summation circuit can effectively carry out the operation of subtraction as well. The two inputs to the circuit was the desired output voltage (either 5V or <1V) and the output of the window comparator with a changed sign representing the obtained output voltage
- Voltage Comparator: Finally, to get the value of (-1V/+1V), the output of the summation was fed into the voltage comparator of Fig. 8. In voltage comparators, the output alternated between the negative and positive saturation

voltage of the op-amp based on the voltage at the non-inverting input terminal.

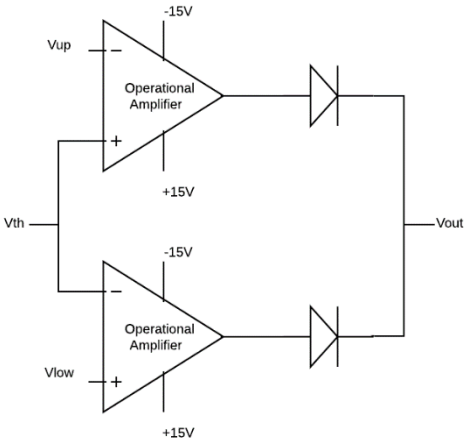

Figure 7: Window Comparator

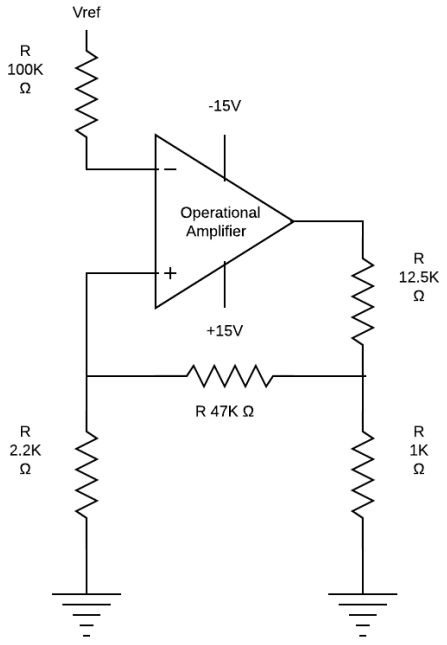

Figure 8: Voltage Comparator

The simulation was conducted using the source simulation software Proteus and the results obtained is plotted in Fig. 9. Since there were two outputs to our network, two error values were computed for $Output_1$ and $Output_2$ as $Error_1$ and $Error_2$ respectively. The representation of eq. 5 in its implementation is given in Fig. 11 and of the hidden layer in Fig. 10. Here again, multipliers and summers were employed to carry out the

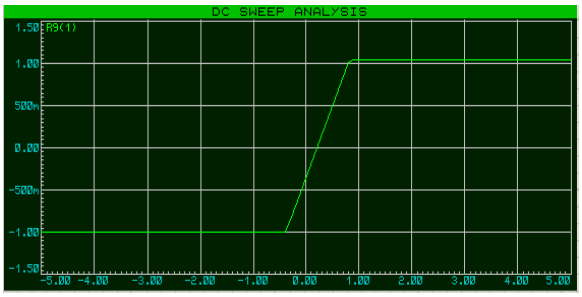

Figure 9: I/O Characteristics of Threshold Activation

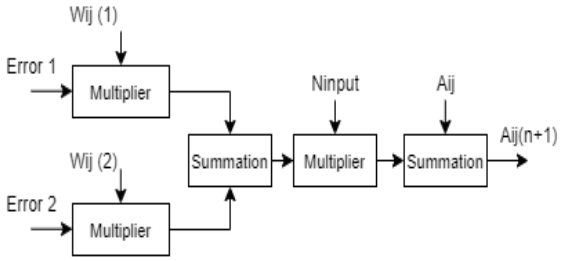

Figure 10: Hidden Layer Weight Update

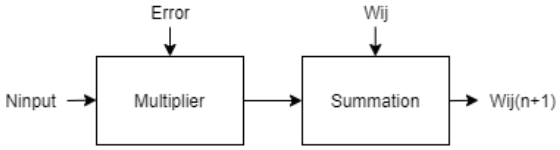

Figure 11: Output Layer Weight Update

operations between the input voltage quantities. These updates were carried out for each weight voltage associated with the neurons following which the output noted from the circuit was noted and the weight values were accordingly changed to obtain new outputs $Output_1$ and $Output_2$. The weight values converged to their minimal error points over successive iterations through the dataset. Fig. 12 shows the error convergence plot over each iteration cycle. The prototype of the backpropagation used is shown in Fig. 13.

6 THE ROVER

In order to provide a rover with obstacle avoidance capabilities, it is imperative that it has at least two object based sensor attached to it on either side so that not only does it senses the obstacle but also adapts its movement so as to move further away from the object

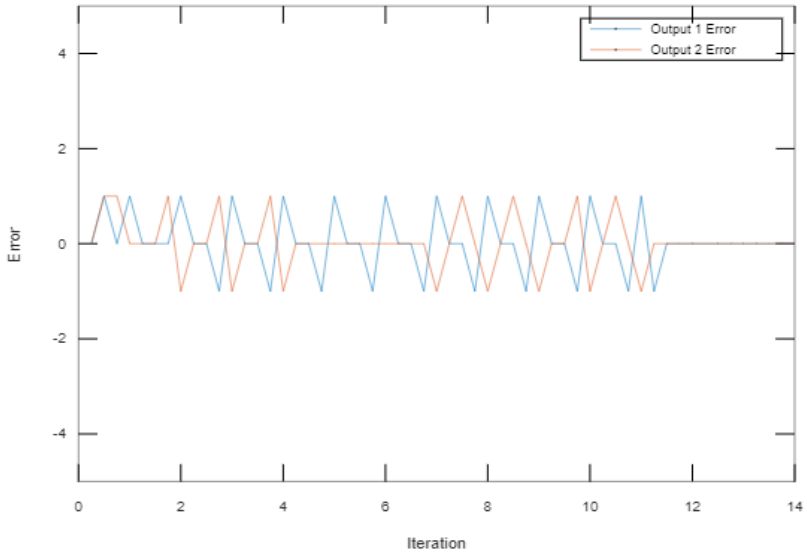

Figure 12: Output Error over Subsequent Iterations

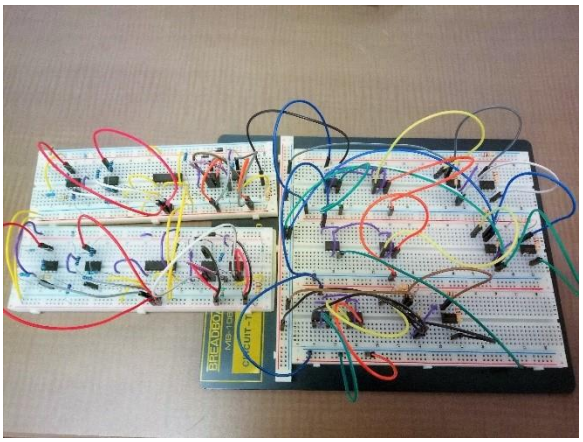

Figure 13: Prototype of Backpropagation Circuit

lying in its path. The rover was fitted with two digital IR sensor modules that transmitted and received IR signals thereby detecting obstacles. The distance of measurement for each of these sensors was set to 9 cm bearing in mind the processing time of signals through the circuit. The digital IR modules mentioned above provide binary outputs – 0 and 1 – to indicate a clear path or a hindrance respectively. For these outputs to be fed into the forward propagation circuit guiding the bot, analog voltages were required. Therefore, a 12-bit DAC converter MCP4725 was interfaced with an arduino UNO board to provide for the same. The output of this was 1V and 4V representing the digital values 0 and 1 respectively. These values were then fed into the forward propagation circuit which was mounted on the rover. The processed output from the circuit was fed into an arduino UNO which was interfaced with four servo motors, two for each side of the wheels of the rover to control it.

7 RESULTS

The robot was successfully tested and it displayed the desired obstacle avoidance capabilities as per its design. On sensing an obstacle directly in front of it at a maximum distance of 9 cm, it halted completely and continued moving when the path was clear. When an obstacle was detected on the left, the processed output from the sensors stopped the right wheels while the left wheels continued to move. The opposite was observed when the obstacle was detected on the right side of the robot. The robot therefore successfully demonstrated the ability of turning and stopping to avoid obstacles.

8 CONCLUSION

The goal of this project was to present a multi-neuron neural net on an analogue circuit which could be trained thereby affirming that circuits can be used as computational cores without employing software computations. There are several aspects to this project which may be addressed in subsequent research:

- In the presented model, the training of the neural net takes through a backpropagation circuitry. Evidently, the weight updates take place manually. Further research is required in order to define a method of automated updation of these weight values. The end-goal should be to make the process of neural computing completely mechanized as is the case in software computing. The model so designed should demonstrate the functionality learning from the physical environment, such that it can self-update to adapt to real-time conditions presented to it.
- The rover was trained for navigation through a path with several obstacles. Future work

should involve training the network of neural net with more complex tasks such as sensing the prevalent conditions in its immediate domain or making sense of raw data procured through sensors attached to it.

- The current model primarily utilizes operational amplifiers to perform computations. The internal circuitry of an op-amp consists of a structured arrangement of transistors which as mentioned previously may still be difficult to build through the self-replicating machine. Succeeding research should be directed towards the physical implementation of the multi-neuron neural network presented here on a vacuum tube-based circuit. Further, the possibility of realizing the concept of backpropagation should also be examined so as to train the model so obtained.

Acknowledgement

This project was carried out as a part of MITACS Globalink Research Internship Program, 2019.

References

[1] Crawford, I. A., Joy, K. H. and Anand, M. (2014). Lunar Exploration. *Encyclopedia of the Solar System* 555–579.

[2] Sanders, G. B. and Larson, W. E. (2012). Progress Made in Lunar In Situ Resource Utilization under NASA's Exploration Technology and Development Program. *Earth and Space* 2012.

[3] Ellery, A. (2016). Are Self-Replicating Machines Feasible? *Journal of Spacecraft and Rockets* 53(2), 317–327.

[4] Jones, R., Haufe, P., Sells, E., Iravani, P., Olliver, V., Palmer, C. and Bowyer, A. (2001) RepRap—The Replicating Rapid Prototype, *Robotica*, Vol. 29, No. 1, 2001, pp. 177–191

[5] Ellery A, Lowing P, Mellor I, Conti M, Wanjara P, Bernier F, Kirby M, Carpenter K, Dillon P, Dawes W, Sibille L, Mueller R (2018) Towards in-situ manufacture of magnetic devices from rare earth materials mined from asteroids *Proc Int Symp Artificial Intelligence Robotics & Automation in Space*, Madrid, Spain, paper no. 10c-1

[6] Parberry (1994) Circuit complexity and neural networks. *MIT Press*

[7] Ellery, A. (2015). Artificial intelligence through symbolic connectionism—A biomimetic rapprochement *Biomimetic Technologies*, 227–252

[8] L. O. Chua, (1971) Memristor: The Missing Circuit Element. *IEEE Trans. on Circuit Theory*, Vol 18, pp 507-519, 1971.

[9] Yonglei Zhao and Guoyong Shi, (2018) A Circuit Implementation Method for Memristor Crossbar with On-chip Training. *IEEE Asia Pacific Conference on Circuits and Systems*, 2018.

[10] A. Thomas, (2013) Memristor Based Neural Networks *Journal of Physics. D: Applied Physics*. 46 p.093001

[11] Ebong, I. E., and Mazumder, P. (2012). CMOS and Memristor-Based Neural Network Design for Position Detection *Proceedings of the IEEE*, 100(6), 2050–2060

[12] Larras, B., Chollet, P., Lahuec, C., Seguin, F. and Arzel, M. (2018) A Fully Flexible Circuit Implementation of Clique-Based Neural Networks in 65-nm CMOS. *IEEE Transactions on Circuits and Systems* I: Regular Papers, 1–12

[13] Yeo, I., Chu, M. and Lee, B.-G (2019). A Power and Area Efficient CMOS Stochastic Neuron for Neural Networks Employing Resistive Crossbar Array. *IEEE Transactions on Biomedical Circuits and Systems*, 1–1

[14] Omondi, A. R. and Rajapakse, J. C. (Eds.). (2006). FPGA Implementations of Neural Networks, *Springer*

[15] Krips, M., Lammert, T., & Kummert, A. (n.d.). (2002). FPGA implementation of a neural network for a real-time hand tracking system. *Proceedings First IEEE International Workshop on Electronic Design, Test and Applications*

[16] Yamashita Y and Nakamura Y (2007) Neuron Circuit Model with Smooth Output Functions *Proc Int Symp Non Linear Theory Its Applications*, Vancouver, Canada

[17] Samantha L and Ellery A, (2015). Trainable Analog Neural Network with Application to Lunar In-Situ Resource Utilization. *66th International Astronautical Congress*, 2015.

[18] V. Braitenburg (1984) Vehicles: Experiments in Synthetic Psychology *MIT Press*

[19] Nehmzow U. (2003). Robot Learning: Making Sense of Raw Data, *Mobile Robotics: A Practical Introduction*, 1st ed., Springer-Verlag Berlin Heidelburg, New York, pp 63-64