IAC-22-D4.5.x68581

NEURAL COMPUTATIONAL ARCHITECTURES FROM IN-SITU RESOURCES FOR PLANETARY EXPLORATION

Alex Ellery
Carleton University, Canada, aellery@mae.carleton.ca

We explore the prospect for leveraging analogue electronic circuitry from lunar resources and determine its utility for controlling production on the Moon. Since solid-state transistor-based electronics cannot be feasibly manufactured on the Moon in-situ, vacuum tube-based active electronics may be configured into neural network circuitry. Recent developments in Artificial Intelligence suggest an evolution of terrestrial computing towards special neural network hardware. We show that a wide range of algorithms can be implemented on analogue neural networks emphasising algorithms required to implement advanced robotics including bio-inspired approaches. The Nakamura-Yamashita analogue neuron circuit is one possibility that we have explored and show that two cross-strapped neurons can implement obstacle avoidance behaviours in rovers. We explore the use of RatSLAM for rover navigation and neural fields for path planning respectively that are uniquely suited to neural network implementation. Rovers will be crucial for production as mining and transport vehicles. For online learning, backpropagation is a simplification of the Kalman filter algorithm that has broad applications in optimal state estimation under stochastic conditions. Challenging process control required for electrochemical throughput may be implemented with neural networks. We finally examine neural network algorithms for complex manipulator controllers which constitute the final phase of production - assembly.

I. INTRODUCTION

The next stage for lunar exploration is consolidation through the establishment of a Moon Village. It would be advantageous to construct the Moon Village from lunar resources as far as is feasible. The crux is to construct the kinematic machines of production themselves from local resources, i.e. robotic machines. In constructing a lunar infrastructure leveraged from in-situ resources, it is apparent that traditional solid-state electronics such as transistors for constructing computing machines would be challenging to manufacture from lunar resources [1]. Although silicon is widely available, dopants are not (Table 1) nor are the reagents.

Element	Average concentration in lunar regolith
H	50 µg/g
C	124 µg/g
Au	9 ng/g
Fe	15% (mare basalt)
Ni	250 µg/g
Co	35 µg/g
W	370 ng/g
Ti	7% (mare basalt)
Al	18% (anorthite)
K	0.8% (KREEP)
P	0.6% (KREEP)

Table 1. Lunar element concentrations (adapted from [2])

However, lunar resources may be deployed to construct vacuum tube-based amplifiers/switches – lunar materials suffice to construct vacuum tubes [3]. Nevertheless, implementation of CPU-based computing would require very large computers that are impractical. An alternative is to employ analogue neural network architectures which are Turing complete (Appendix). We show that the physical neural network as a computational architecture is ideally suited to the range of applications required for robotic manufacturing systems on the Moon.

II. ARTIFICIAL NEURAL NETWORKS

We have adopted a simple neural network approach based on analogue electronics hardware. Neurons are finite state machines - the Elman simple recurrent neural net is a Moore machine with an output function of the form [4]:

$$y_i(x(t)) = f\left(\sum_{j=1}^{n} w_{ij} x_j(t) + w_i\right) \quad [1]$$

where y=output, x=input, w_{ij}=connection weight, w_i=threshold, f(.)=nonlinear activation function. Electrical representation of neural networks is based on the neuron as a weighted sum computation implemented as a multiply-and-accumulate operation. Hardware neural nets perform multiply (weights) and accumulate (summation) operations with forward propagation of voltages through a series of layers:

$$V_{out} = f\left(\sum_{i=0}^{N} V_{w(i)} V_{in(i)}\right) \quad [2]$$

where $f = \frac{1}{1+\exp(-I_{in}/I_{ref})}$=activation function implemented through piecewise linear approximation, $V_{in(0)}$=input threshold. Fixed weight neural networks cannot learn so the backpropagation algorithm requires computation of the mean squared error at the output layer:

$$E = \frac{1}{2}\sum_{i=0}^{N}\left(V_{out(i)} - V_{tgt(i)}\right)^2 \quad [3]$$

The chain rule provides the partial derivative of the error required for weight update at the output layer:

$$\frac{\partial E}{\partial V_w} = \frac{\partial E}{\partial V_{out}}.\frac{\partial V_{out}}{\partial\left(\sum_{i=0}^{N} V_{w(i)}V_{in(i)}\right)}.\frac{\partial\left(\sum_{i=0}^{N} V_{w(i)}V_{in(i)}\right)}{\partial V_w}$$
$$=\left(V_{out} - V_{tgt}\right).\frac{\partial V_{out}}{\partial\left(\sum_{i=0}^{N} V_{w(i)}V_{in(i)}\right)}.V_{in} \quad [4]$$

For the hidden layers, $\frac{\partial E}{\partial V_{out}}$ is a more complex expression derived from the chain rule. From this, backpropagation (generalised delta) circuitry may be implemented as a gradient descent on the error:

$$V_w(t) = V_w(t-1) - \eta\int_0^T \frac{\partial E(t)}{\partial V_w(t)}dt \quad [5]$$

Neural networks are limited in size by their speed of learning convergence and generalization capability. This suggests that multiple neural networks may be employed in ensembles to improve their performance [5]. Each expert network learns a subset of training cases and a separate gating network determines which network is adopted for each training pair according to the one that gives the smallest output error [6]. The final output constitutes a linear combination of the outputs of the local expert networks. Alternatively, the gating network may select local experts in a hierarchical architecture [7]. This suggests that neural networks have a scalability characteristic for dealing with more complex problems. One of the simplest biological neural networks in a non-aquatic free-living animal is that of the nematode worm *C elegans*: it comprises 959 cells in total as a hermaphrodite (of which 302 are neurons) or 1031 cells in total (of which 381 are neurons) as a male with approximately 5000 synapses. This potentially gives us a minimum neural network size for adaptive behaviour though an engineered version would be subdivided into subnetworks of more modest dimension. For example, the SpiNNaker project is based on combining a large number of digital ARM processors within a grid of switches to emulate a vast neural network representing a small brain of ~10^6 neurons [8] but SpiNNaker is not energy efficient.

III. HARDWARE NEURAL NETWORKS

Neural network computing offers an alternative to the von Neumann architecture [9]. Here, we consider hardware implementations of neural networks. Neuromorphic circuits are brain-inspired non-von Neumann architectures. In spiking neurons, a summation of action potentials from presynaptic inputs generates a postsynaptic pulse (action potential) [10]. Spiking neurons may employ rate coding in which the mean firing rate transmits information or time coding in which the time to first spike or interspike interval transmits information.

Hardware spiking neurons promise the temporal dynamics of biological neurons that combine memory with processing such as leaky integrate-and-fire neurons with the high capacitance of cell membranes:

$$C_i\frac{dV_i(t)}{dt} = -I_0 + \sum_{j=1}^{n} w_{ij}x_j(t) \quad [6]$$

with $\begin{cases} V_i < V_{th}\ for\ s_i = 0 \\ V_i \geq V_{th}\ for\ s_i = 0\ and\ V_i = 0 \end{cases}$

where V_i=internal potential for neuron i, C_i=membrane capacitance, V_{th}=neuron threshold, s_j=output of neuron j, I_0=leakage current, w_{ij}=synaptic weight between neuron j and neuron i. Integrate-and-fire neurons are dependent on membrane potential with leakage coefficients offer a more biologically plausible neuron model [11,12]. They are sensitive to specific input frequencies due to stochastic resonance whereby they output spikes with high signal-to-noise ratio [13]. Spiking neuron hardware is preferably analogue which offers much higher energy and areal footprint efficiency than digital forms for the same spiking rate [14].

The more complex Izhikevich spiking neuron balances biological emulation with computational complexity based on two differential equations [15]:

$$\frac{dV}{dt} = 0.04V^2 + 5V + 140 - U + I$$
$$\frac{dU}{dt} = a(bV - U) \quad [7]$$

Spike reset conditions are given by:

if V≥30 mV then $\begin{cases} v \leftarrow c \\ v \leftarrow u + d \end{cases}$

A spike-based leaky McCulloch-Pitts neural network with large layer sizes learned autonomous obstacle avoidance and navigation in a simulated grid environment using reinforcement learning [16]. A single optimised Izhikevich neuron model has been implemented on an FPGA [17]. Silicon spiking neurons as analogue/digital VLSI chips range from axon-hillock neurons to Izhikevich integrate-and-fire neurons to Hodgkin-Huxley conductance-based neurons with increasing hardware footprint and consequent richer dynamics [18]. Learning is challenging in spiking neural networks due to non-differentiable discontinuous spiking states. Spike timing dependent plasticity (STDP) is a Hebbian learning rule depending on relative timing of presynaptic and postsynaptic spikes. There are neuron models that can implement greater biological fidelity that may be explored [19]. This includes implementing a neural firing rate that is proportional to the product instead of the sum of weighted inputs – this constitutes a higher order neural net. These microtechnology-based hardware neurons are not

feasible on the Moon. Analogue spiking neurons however are feasible [20].

An all-optical neural network offers speed and energy advantages over electronic media. Optical neural networks offer rapid highly parallel processing but must implement matrix multiplication, summation, nonlinear activation, convolution and learning algorithms. Hardware negative weights may be implemented using inhibitory neurons into the hidden layer [21]. Crucial to optical neural nets is the representation of weight and matrix multiplication optically [22]. Spatial light modulator arrays representing a weight matrix sandwiched between LED and photodetector arrays offer inferior nonlinear processing performance and limits in weight array size. Holographic optical arrays and phase mask arrays are diffractive approaches that are immature. The most mature approach is to exploit light interference using arrays of Mach-Zehnder interferometers. Optical interconnects (lenses) can implement convolution and Fourier (inverse) transform optics in hardware. Optical fibres are doped with erbium to amplify light transmission near 1550 nm by up to 26 dB. Optical processing with optical switches can reduce power consumption where Fourier transforms exploit light's interference properties implemented as a 2D array of Mach-Zehnder interferometers but its N^2 matrix representation imposes scaling limits. Basic analogue optical components include an optical gate which may be implemented as a Mach-Zehnder interferometer amongst other options [23,24]. Optoelectronic approaches are necessary to implement nonlinear activation - integrated on-chip photonic deep neural networks require opto-electronic nonlinear processing through a pn junction micro-ring modulator [25]. In optical neural networks, error gradients may be computed from intensity measurements [26]. Implementation of backpropagation in optical neural networks is particularly challenging as it requires error derivatives [27]. An all-optical neural network has been implemented based on optical spiking neurons [28,29]. Micro-ring resonators can modulate optical waveguides through phase change materials that switch between crystalline states to implement adjustable synaptic weights. A photodiode imaging array sensor may be integrated with the neural network processor through tunable photoresponsive synaptic weights between the photodiodes [30]. A 3D printed optical diffractive deep neural network with a high neuron count can perform complex computations at light speed [31]. It comprised of multiple transmissive/reflective layers on which spatial light modulators represent neurons connected through the layers via optical diffraction. The network was trained as an imaging lens to implement a physical autoencoder network. In general, optical neural networks are immature, suffering from optical precision limits of error backpropagation and energy dissipation in nonlinear optics limiting their scalability. To compensate for limitations in all-optical neural networks, hybrid optoelectronics integrates electronic and photonic circuits onto silicon through lasers, photodetectors, optical waveguides, electro-optical modulators, etc [32]. However, these approaches require microtechnology fabrication which are not currently feasible on the Moon.

There have been several approaches to non-optical hardware implementation of neural networks [33]. An early approach to on-chip learning in VLSI implementations of neural networks was using the virtual targets algorithm with weights updated locally and commonly for all layers [34]. A transconductance multiplier generates a current pulse whose amplitude and duration encodes a neuron's weight and level of state activation respectively which is integrated, effectively computing $\int w_i x_i$. The sigmoid derivative is approximated by a triangular pulse generated by an integrator output from an EX-OR circuit square wave generator. Weights were held as voltages on capacitors with the midpoint voltage as zero weight to permit negative weights. The backward passing of error signals through the weight-encoding capacitors was not addressed. Neuron weights may be implemented as a MOSFET with output current proportional to the difference in the positive and negative input lines followed by nonlinear current-to-voltage conversion [35].

There are several dedicated ASIC hardware chip implementations of neural networks such as CNAPS (connected network of 64 adaptive processors) based on the N6400 neurochip connected by a broadcast bus in a SIMD configuration. Digital neurochips have traditionally been single instruction multiple data (SIMD) architectures in which multiple processors form systolic arrays, e.g. WARP. VLSI/ULSI implementation of neural networks such as associative memories and self-organising maps on CMOS circuitry is a costly approach [36]. Such VLSI approaches are inflexible but this can be overcome with the field programmable gate array (FPGA) hardware which offers programmable logic blocks with low power consumption. The implementation of neural networks [37] on FPGA for real time applications [38] have also been studied and reviewed [33]. To compensate for hardware-intensive multiplication operations in neural networks, time-multiplexing of forward and backward computations may be employed [39]. Fixed point representation in neural networks yields more hardware efficient and faster FPGA computation than floating point representations [40]. However, the Xilinx ZU9CG System on Chip FPGA can perform forward and backward computations for deep neural networks [41]. VHDL-implemented Boolean logic functions on FPGAs may be exploited to implement step activation

McCullough-Pitts neurons configured in neural networks [42]. Each neuron implements a Boolean function represented by its truth table. This involves a two-step procedure. Neural inputs are encoded into 2-complement binary representations with weights followed by conversion of the resulting binary neuron model into a logic gate structure. Replacement of negative weights is implemented through application of the modulus operator with reversal of affected inputs using a NOT gate. The binary neurons are then mapped onto the hardware structure through a decomposition process with an inversion process to inputs with former negative weights. The final step was to convert a netlist of the circuit into VHDL code using a C++ program. A fuzzy logic controller including fuzzifier, fuzzy learning base, fuzzy inference engine and defuzzifier may also be implemented on an FPGA [43] suggesting the feasibility of implementing fuzzy neural networks on FPGAs. The Tianjic chip is a multi-core, reconfigurable architecture that can accommodate brain-inspired circuits including spiking neurons and machine learning algorithms [44]. Deep learning neural networks (DNN) have been growing in capacity by an order of magnitude per year since 2010. Deep learning-based AI has induced a shift away from the CPU to the TPU (tensor processing unit), an ASIC/DSP that enables rapid parallel matrix multiplications required for neural network processing. Training and operating DNNs which require CPU, GPU or TPU-based platforms consume exponentially growing amounts of energy. As AI applications grow more pervasive, so energy demand will grow, potentially without limit. The advantage of hardware learning is considerable speed-up in training - in 2020, OpenAI created GPT-3, a DNN of 175×10^9 parameters.

IV. ANALOGUE NEURAL NETWORKS

A hardware integrate-and-fire neuron based on analogue components has been presented in which resistor-weighted input currents were summed and integrated through a simple parallel RC circuit by Kirchoff's laws [45]. This was fed into a transistor that acted as a thresholded pulse generator. The transistor may be substituted with a triode-based Schmitt trigger.

We consider fully analogue neural networks. There may be some interesting implications in extending analogue forward neural circuit model into a recurrent neural network circuit in the future. A finite analogue representation of a recurrent neural network with sigmoidal neurons represents a model of analogue computation and can simulate a universal Turing machine [46]. A recurrent neural network with rational weights is computationally equivalent to a Turing machine computer model. When the weights are rational numbers, the addition of stochastic properties (i.e. unreliable components) extends the power of their computation to that of a probabilistic Turing machine [47]. However, the computational abilities of analogue recurrent neural networks may be enhanced beyond that of the Turing machine using real-valued weights which offer the infinite precision of analogue information [48]. This allows them to compute non-recursive functions that cannot be computed by Turing machines. An example is Rado's construction $\Sigma(n)$ as a non-computable function of n [49]. Analogue recurrent neural networks thus constitute super-Turing machines. Theoretically therefore, analogue implementations of neural networks offer a more powerful methodology than traditional computing architectures. This is an interesting prospect that has yet to be explored.

Analogue recurrent neural networks are equivalent to oracle Turing machines that consults an oracle, a special tape [50]. The oracle introduces the possibility of inputting new non-deterministic or non-computable data to the machine. It may be regarded as a hardware upgrade through which the Turing machine gain computational advantages by interacting with the more powerful oracle. These interactive machines may be capable of super-Turing computations [51]. Reservoir computing shows that recurrent neural networks with large hidden layers can perform computation efficiently by exploiting the rich dynamics of nonlinear neurons with delayed feedback [52]. Analogue recurrent neural networks where the synaptic weights evolve over time (i.e. learning) are equivalent to interactive Turing machines with advice, i.e. they are capable of super-Turing computations – this is the case for both rational and real weights [53]. In rational-weighted recurrent neural networks, these super-Turing capabilities can only be realized if the synaptic weights evolve in a non-recursive manner – recursive learning limits such networks to Turing-computable functions. However, in real-weighted recurrent neural networks, there are no such requirements for learning to achieve super-Turing capacities. For example, two interconnected sub-threshold CMOS invertor circuits, each comprised of two triodes operating in tandem, can implement a Maxwell's demon-type of rectification and amplification of voltages generated by thermal fluctuations without contravening the second law [54]. Thus, learning from environmental cues imposes enhanced capacities to neural computation. This establishes the importance of physical interaction with the environment as a key element in the behavioural capacities of robotic vehicles [55,56].

Fully analogue neural circuits offer rapid computation with some biological fidelity in reducing specific energy consumption (energy/neuron) but at the cost of a fixed neural architecture (without learning). Neural networks per se are energy efficient as their processing and memory hardware are the same. Training neural networks however requires large

amounts of data processing while neural network inferencing relies on synaptic memory. Through the denial of online learning, it may be that we lose super-Turing capabilities but retain Turing machine capabilities. We consider this an acceptable cost. Fixed weights prevent runaway growth in computer footprint restricting the output of our Turing machine to an analogue neural network that encodes a specific algorithm in hardware form. The complexity of a neural network increases only as the logarithm of the task complexity unlike the exponential increase in circuit complexity of digital architectures [57]. Neural synaptic weights may be represented as a network of resistors which we assume are fixed in a pre-trained network. The simplest model of the neuron is a passive low-pass filter comprising a resistor in parallel with a capacitance.

Analogue circuits primarily based on operational amplifiers for general computation has been examined, e.g. the Yamashita-Nakamura neuron [58]. It comprises four operational amplifiers, two diodes, seven resistors and one capacitor in a three-stage circuit comprising a summing weighted input amplifier, a time delay circuit and an output amplifier with nonlinear diode feedback for the sigmoid response. Analogue implementation of the sigmoid function is difficult to achieve reliably however due to non-uniformity of components, so our version adopted a simplified McCulloch-Pitts signum function implemented with a comparator circuit (Fig 1).

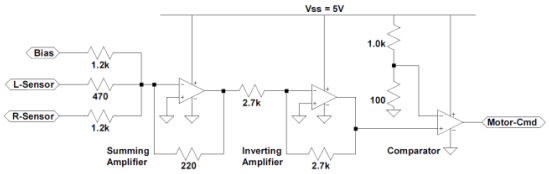

Fig. 1. Modified Yamashita-Nakamura neuron

The weights of each neuron are pre-trained offline to implement its desired behaviour.

To test the efficacy of fixed weight hardware neural networks in different tasks required for in-situ resource utilisation on the Moon, neural networks have been demonstrated for a wide range of sophisticated tasks. These include nervous nets for central pattern generation [59], Buffon's needle algorithm (for areal surveying) [60], autonomous online measurement of soil parameters (for geotechnic surveying) [61], autonomous drilling control (for subsurface mining) [62,63], symbolic artificial intelligence [64], etc. Indeed, some tasks are inherently suited to neural networks - a bio-inspired approach to image processing implements optic flow navigation which is implementable on neural networks [65]. A correlation algorithm to locate motion peaks between images can be implemented using AND-NOT gates [66]. There are three steps to the implementation of such a correlation algorithm in a neural network: (i) shift-and-comparison of velocity measurements; (ii) local summation that implements spatial averaging of an image region; (iii) winner-take-all scheme that suppresses all non-maximum matches selects the highest matching strength velocity. Of course, image processing has been the main application of deep learning neural networks exploiting in particular convolutional neural networks [67].

V. AUTONOMOUS NAVIGATION

The first generic task we examined was autonomous navigation. As well as surveying and mining, mobile rovers are used in manufacturing facilities to move material between stations. In all cases, simultaneous localisation and mapping (SLAM) is a fundamental capability. We have demonstrated a pre-trained two-neuron hardware circuit implementing a reactive Braitenburg control architecture of BV2/BV3 class [68] performing automatic obstacle avoidance on a desktop mobile robot (Fig 2):

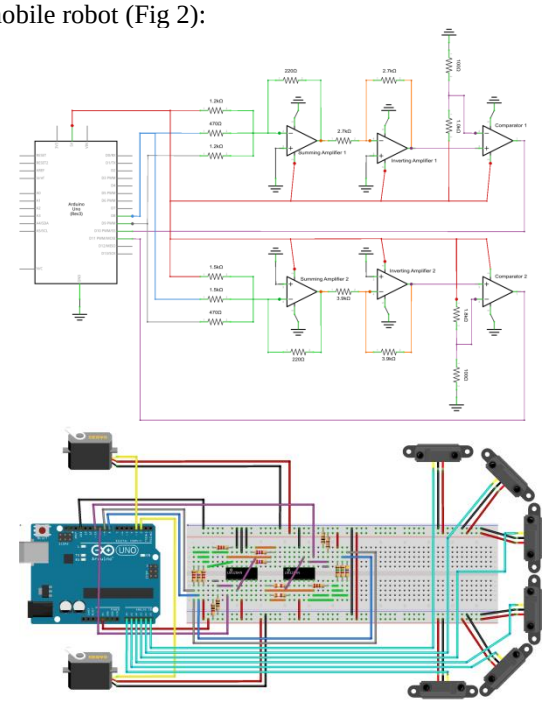

Fig. 2. (a) Two-neuron hardware controller circuit; (b) desktop rover representation with frontal sensors

The simplest Braitenburg architecture (BV1) of one sensor and one motor connected by an excitatory link exhibits wandering behaviour. BV2 comprises two sensors and two motors connected through excitatory links – they may be ipsilateral with avoidance (fearful) behaviour or contralateral with approach (aggressive) behaviour. BV3 is similar to BV2 but with inhibitory associations to implement approach (attraction) and avoidance (repulsion) with opposing connections. Our two-neuron circuit is of the BV2/BV3 type which successfully performed multiple obstacle avoidance tasks reactively along a curved obstacle field of posts. A

more sophisticated variant - BV3c – could be readily implemented with all the previous connections (BV2a,b and BV3a,b) linked the two motors to exhibit variable speeds. BV4 builds on BV3c and implements nonlinearities in the links which generates an approach to objects with increasing speed until it reaches a threshold beyond which it slows. BV5 implements thresholding allowing it to implement logic gates similar to McCulloch-Pitts neurons (our neurons are McCulloch-Pitts neurons) with reciprocal thresholding for memory storage. BV6 implements evolutionary learning.

Now, BV7-14 networks exhibit sophisticated neural network learning based on association with cognitive properties, object detection, movement detection, map-building and prediction through the introduction of wires with special properties. It is these capacities that we turned to next. To explore these more complex neural networks, we initially trained a software neural network to implement sophisticated tasks central to the realisation of robust in-situ realisation in mining rovers. Specifically, we implemented a multilayer perceptron with goal-directed navigation trained offline in software form to be subsequently constructed in hardware. We consider only the training of the software implementation here and used a rover model in Webots (Fig 3).

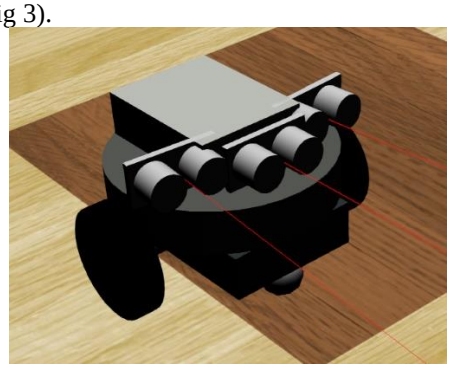

Fig. 3. Webot model of a simple rover

Labelled data pairs were input to the network and the weights adjusted through the backpropagation learning algorithm. The error function implemented was the softmax-loss function:

$$E(y, w(t)) = \log\left(\sum_{j=1}^{m} e^{w_j}\right) - w_{out} \quad [8]$$

The numbers of neurons in the input (sensors) and output (motors) layers is defined by the problem to be solved. The architecture of this network has a 5-4-4-4 topology with a sigmoidal activation function. The five-neuron input layer corresponds to three normalised distance sensor measurements to obstacles in the forward environment field with flanking sensors angled at $\pm 5^{o}$, a normalised distance measurement to goal, and a normalised polar angle measurement to goal. Normalisation to the maximum sensor range distance (100 cm) from the rover for the normalised obstacle distances, initial distance to goal for the normalised goal distance and π for the normalised polar angle to goal. Normalisation improved learning by eliminating zero input sensor readings. The input data is projected forward to the first hidden layer and then second hidden layer both of four neurons.

In any hardware neural network, the hidden layer(s) must be pre-defined. In a radial basis function, the number of hidden neurons n (neural complexity) equates to the number of training examples k (information complexity): $n(e) \leq k(e)$ to approximate a function f within an error e [69]. In reality, n<<k to limit the network size. Upper and lower bounds on the size N of a single hidden layer in a multilayer perceptron network have been defined as [70]: m+1≤N≤p(m.n) where $p(m.n) = \sum_{i=0}^{n} \binom{m}{i}$ = number of separate input-output pairs in the training set, n=number of input neurons, m=number of network layers. However, when the training set is much larger than the number of neurons, two or more hidden layers may require much fewer neurons collectively than a single hidden layer, even though a single hidden layer suffices for function approximation [71]. The number of hidden neurons encapsulates a trade-off between the complexity of the model and its approximation accuracy: this occurs at its maximum (optimal) generalization ability with the minimum predictive error [72]: $e =< h(u_0) - h(u_j) >$ where $u_j = y_n - f_j(x_n, w)$, h(.)=loss function, y_n=target outputs, j=number of hidden neurons, n=number of output neurons. A Bayesian information criterion (BIC) yields a test procedure defined by:

$$p_j = \frac{\sum_n h((u_0) - \sum_n h(u_j)}{\sum_n h(u_j)} \quad [9]$$

Hence, the number of hidden neurons can be determined only with substantial a priori knowledge of the learning environment. We adopted a simpler approach for our small network - the number of neurons in the hidden layers were selected to be less than the input layer (for compression) and at least as high as the output layer. Finally, the four-neuron output layer determines the rover movement of its front castor wheel and two differential wheels – forward, left, right and stop. The simulated world comprised a randomly-generated rock field of obstacles over 5 x 10 m (Fig 4) corresponding to approximately to the visual field of view of the ExoMars rover.

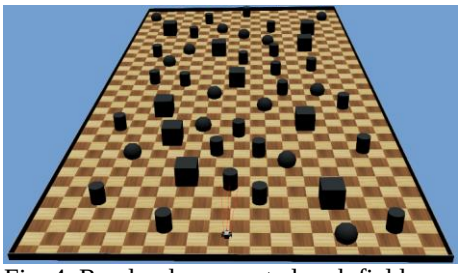

Fig. 4. Randomly generated rock field

Training data of goal position within 100 m and sensor distance values within 100 cm were randomly generated to prevent manual biasing. The human operator showed the direction required to reach the goal. 1000 different training scenarios were used during training. Although the environment was unchanged, the end goal position was varied (input). The trained network was tested with three different goals which it successfully achieved each time, though some goals took longer to reach due to the rover swaying during transit. We expect that the trained network can cope with any rockfield distribution as the process of varying the goals presents a different rockfield perspective [73]. We have yet to test this though.

The goal is to train the network offline such that it can be used to build an analogue neural circuit for the rover and provide it with obstacle avoidance and maneuvering capabilities without software computations and simulations. The circuit for the multilayer neural net based on the Yamashita-Nakamura neuron. However, given the simulation-reality gap, it is necessary that hardware neurons have the ability to be trained through weight adaptability [74]. 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 adapts to the distance measuring sensor inputs fed to the circuit. The output is subsequently fed to a 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.

VI. KALMAN FILTERING

Kalman filtering is the basis for families of SLAM algorithms as well as a host of other ubiquitous applications. Simple filters may be implemented directly in analogue op-amp circuitry. Self-tuning filters are a special case of analogue electronic filters comprising a voltage-controlled filter analogue filter (with fixed resistances replaced with voltage-controlled resistors) and an operational amplifier [75]. The Kalman filter is a ubiquitous state estimation algorithm that fuses noisy sensory data with a dynamic predictive model, i.e, a Bayesian algorithm. The chief problem with implementing the Kalman filter in a neural network are matrix computations. A two neuron-based Kalman filter can be based on a nonlinear autoregressive model to predict variables of the Kalman filter but it is complex in form [76]. Kalman filter is a model-based state estimator while neural networks are model-free. KalmanNet learns the Kalman filter gain $K_t = (AP_{t-1}A^T + Q)H^T(HP_{t-1}H^T + R)^{-1}$ from labelled datasets of Kalman gain to state estimation output [77]. The backpropagation algorithm can implement a Bayesian classifier [78] and a degenerate form of the extended Kalman filter [79]. This gradient descent approximation to the Kalman filter through the derivative of a loss function (error between predicted and measured output) was the basis of the neural Kalman filter [80]. A simplified EKF training algorithm may be deployed for gradient descent for LSTM for superior performance by [81]:

$$w_i(t) = w_i(t-1) + K_i(t)(y^d(t) - y(t)) \quad [10]$$

where $K_i(t) =$

$$\left[P_i(t-1)H_i^T(t)\left(\sum_i H_i(t)P_i(t-1)H_i^T(t) + R(t)\right)^{-1}\right]$$

Comparison between the multilayer perceptron and the Kalman filter indicate the superiority of the former in terms of computational requirements [82]. For example, artificial neural networks may substitute for Kalman filters for sensor fusion for vehicle navigation [83]. We have implemented the backpropagation algorithm as analogue circuitry so we have implemented a simplified form of Kalman filter circuitry. A neural attractor version of the Kalman filter can model head direction cells [84]. Neural networks may be employed to merge single EKF-generated maps from individual robots to form multirobot SLAM [85]. The neural networks perform map learning on Canny edge-processed images using the unsupervised self-organising map. Obstacles are learned through clustering occupied cells of the occupancy grid which are matched through cross correlation.

There are several intriguing possibilities for learning circuitry including the backpropagation algorithm [86]. We have also been exploring the potential for augmenting hardware neurons with online learning circuitry [87,88]. We have described in detail our analogue hardware backpropagation algorithm that effectively implements a degenerate Kalman filter state estimator [89].

VII. BIO-INSPIRED RATSLAM NAVIGATION

Biologically-inspired models of navigation in the hippocampus are primarily based on a priori “place” cells linked by synaptic weights which are learned during exploration forming a topological map [90]. A

cognitive map must encode two types of information – view-based place recognition (what information) and spatial relationships between them (where information). Cognitive maps may encode a temporal sequencing of views [91]. In insects, a sequence of view-based snapshots with vectors provide the basis for navigation through familiar terrain. Honeybees use snapshot images of landmarks adjacent to the target location to determine its steering [92]. Only the local region and the horizon are extracted from the images. Matching of the stored snapshot image in memory and the retinal image of the current view proceeds by pairing the snapshot with the closest match to the current view, yielding a vector to the target. Graded fusion on a local motor map gives bearings of obstacles which are to be avoided whilst pursuing the direction to the target. Obstacles are avoided through this snapshot map of obstacles in polar coordinates from the current heading. Their visual memory of landmark snapshots is versatile and is tolerant of mirror reversal and rotation of horizontal patterns but not vertical patterns as might be expected under natural conditions [93]. In insects such as flies, heading is represented by compass neurons that activate the ring-shaped ellipsoidal body forming a ring attractor network in its brain that activate specific visual snapshots [94]. Birds which require spatial memory of food caches possess hippocampal place cells similar in function to mammals [95].

RatSLAM is a biologically-inspired SLAM algorithm implemented as neural networks based on the function of the rat hippocampus (specifically CA1-CA3 regions) applicable to a changing rather than static world [96-98]. The hippocampus (of the rat) implements a topological (cognitive) map representation to support self-navigation - place cells in its CA3 and CA1 regions selectively fire when the rat is in specific absolute locations of its environment and head-direction cells that selectively fire only when the rat's head is oriented in specific absolute directions with respect to its environment. Firing of these cells encode the state (x,y,θ) of the rat within a 2D model of its environment. Place cells are not strictly cartesian or topological representations of the environment but they permit interpolation between place cells. In bats which have long ranges, place cells fire in different combinations for different locations, i.e. multiscale coding with a finite number of neurons [99]. There are, indeed, grid cells within the entorhinal cortex that encode multiple topographic rat locations arranged in a tessellated 2D hexagonal grid over the global environment supplemented by locally-ordered 3D grid cells [100]. The global lattice structure is crucial for odometric measurement of self-motion. The entorhinal cortex feeds into the hippocampus in which place cells are more precise – the hippocampus loops through CA1 and CA3 back into the entorhinal cortex. CA3 exhibits recurrent links suggestive of auto-associative memory implementing cognitive maps.

RatSLAM is a partial grid and topological representation of the physical environment similar to GraphSLAM – neighbouring locations are represented as closely adjacent "pose" neurons connected by excitatory links and inhibitory links to more distant neurons. RatSLAM uses proprioceptive (odometry) and external (landmark) sensors to create a semi-metric spatial map of its environment with a competitive attractor network to integrate its sensory data. A 3D neural attractor network is comprised of pose cells, each pose cell encoding an estimate of the robot's pose in (x,y,θ) space. Two representations are used – a global world-referenced experience map of accumulated odometry in conjunction with a local robot-centred obstacle map of visual landmarks [101]. Each local view is associated with a specific scene. Associations are learned between sensory cues V_{ijk} and pose estimates P_{ijk} with weights $w_{i+1} = w_i + \eta P_{ijk} V_{ijk}$. Pose cells are activated by local view cells along the weighted connections if a familiar scene is encountered. Pose cells are activated with a sensory firing rate that indicates the fit to the specific location with firing activity changing as $\Delta P_i = \sum_i w_i V_i$ where P_i=cell activity, w_i=connection weight, φ=global inhibition. This association with global pose and local views represents experience mapping [102]. Each pose cell is connected to proximal cells through fixed excitatory/inhibitory links. The distribution of excitatory and inhibitory connections is weighted as a 3D Gaussian function creating a Mexican hat function. RatSLAM increases the association (connection strength) between simultaneously active local view cells (encoding the visual scene) and pose cells. Active pose cells create the nodes of the global experience map thereby encoding the interconnectivity of different places. The experience map is a set of topologically linked locations (grid cells) in a grid with associated metric local views to allow navigation.

In biomimetic navigation, routes are defined as sequences of recognition-triggered actions to form a cognitive map [103]. Path planning between places is achieved through steepest descent from the goal to the current location. This bears some similarities to the PerAc (perception-action) neural network architecture that comprises a (reflex) action level and a (situation-recognition) perception level linked together through an associative reinforcement-based learning rule [104]. It is capable of generalisation from prior learned situations. The local view is compared with stored snapshot views of landmarks using similarity measures which serve to focus attention on the current view. Landmarks ("what" information) is correlated with (x,y) position ("where" information) represented by place cells. New place cells are generated by landmarks when place cell activity is

below threshold. Place cells may be linked using potential fields through gradient descent in which place cells constitute potential minima. The potential field constitutes the action vector as the derivative of the potential function. This equates to the appetitive and aversive stimulus response behaviours in which avoidance gradient is steeper than the approach gradient [105]. To prevent immobilisation in local minima, orientation cells provide cues through winner-takes-all competition for selecting directions to goals. Each place cell forms a basin of attraction that can be modulated by a motivation measure that encodes priority that selects the preferred direction of movement. Similarly, a motivational module (encoding curiosity for information, hunger for energy and fear of harm) was adopted to teach through reinforcement learning a cognitive map implemented as a time growing neural gas algorithm [106]. An extension of this is to include a gating network to act as a meta-controller to switch strategies based on context [107]. The linkage between place cells may be implemented through hypothetical transition cells to represent sensorimotor actions rather than place locations to form a cognitive map [108]. One possibility for such linkages is the potential field map. Reinforcement learning of situation-action associations (policy) pairs can incorporate a potential field gradient [109] through

$$w_{ij}(t+1) = w_{ij}(t) + \eta[z(t+1) + b_i(t) - b_j(t)]e_j(t) \quad [11]$$

where z(t+1)=reinforcement signal, b_i=estimated incremental cost, b_j=estimated cumulative cost, e_j=eligibility factor incorporating gradient descent. Indeed, it appears that the medial entorhinal cortex encodes vector operations using distance and direction from obstacles to determine self-location [110]. Deep belief networks with three layers (convolution layer – max-pooling unit – convolution layer) may be trained to recognise visual features in a stereo-image pair to classify and predict offroad terrainability for a hyperbolic polar map [111].

VIII. NEURAL FIELDS

Potential field methods are a powerful mechanism for path planning. Neural networks may be configured to implement neural fields which define agent motion with heading direction and velocity as control variables while the environment is characterized by multiple goal heading directions ψ_{tgt} and multiple obstacle heading directions ψ_{obs} relative to allocentric reference coordinates [112,113] extracted from vision sensors. The rate of change of heading direction is given by:

$$\dot{\phi} = f(\phi) = F_{tgt}(\phi) + \sum^{n} F_{obs} + n \quad [12]$$

These define a total vector field with fixed points defined by $\dot{\phi} = 0$. If the neighbourhood exhibits a negative slope then the fixed point is an attractor. Repellors may be defined similarly with positive slope. These may be regarded as dynamic plannjng behaviours ϕ(t) [114]. Behavioural dynamics are represented in phase space with behaviours representing attractor solutions. Attractors in the environment define motivations which diffuse through the neural network. The gradient of the slope around an attractor determines the relaxation time:

$$\tau_{rlx} = -\left(\frac{df(\phi)}{d\phi}\right)^{-1}_{\dot{\phi}=0} \quad [13]$$

At larger distances, different attractors may contribute to the motion of the robot due to fusion in the vector field. At a critical distance, bifurcations occur wherein one attractor becomes dominant. The heading direction φ may be represented through a neural activation function u(φ) which evolves over time: $\dot{u}(\phi,t) = f(u)$ defining the vector field u(φ,t). The stimulus input may be excitatory (from attractors) or inhibitory (from obstacles). The Amari neural field is governed by:

$$\tau\frac{du(\phi,t)}{dt} = -u(\phi,t) + I(\phi,t) + h + \int w(\Delta\phi)f\big(u(\phi',t)\big)d\phi' \quad [14]$$

where φ=heading, $\Delta\phi = \phi - \phi'$=polar difference between the two neurons, τ=dynamic timescale of system, u(φ,t)=neural activation field to encoding direction φ, I(ϕ,t)=external input stimulus encoding direction φ, h=global inhibition or excitation, w(.)=interaction kernel with Mexican hat-shaped response fields, f(u)=local activation function (step function):

$f(u) = 1$ for u≥0
= 0 for u<0

The integral term describes a weighted summation of activity including f(u) defining the neural firing rate. The Mexican hat function shapes the neural field with short-range excitatory connections and long-range inhibitory connections. The sigmoid function may be used rather than the Mexican hat function. The global inhibition term ensures that there is neural activity in the absence of inputs. If discretised, the synaptic weight between two connected neurons is given by a Gaussian function:

$$w(\phi,\phi') = kexp\left(-\frac{(\phi-\phi')^2}{2\sigma^2}\right) - w_{in} \quad [15]$$

where σ=excitation range, k=excitation amplitude, w_{in}=global inhibition factor which localises the peak. Localisation may be associated with sets of landmarks and their bearings (emulating place/head direction neurons of the mammalian hippocampus). Different locations are defined in terms of their topological

relations on a cognitive map. Path planning is implemented by gradient following as in the potential field but this potential field includes frictional terms. Coupled sets of neural fields representative of different neuronal populations can implement goal-directed behavior [115]. Recurrent interconnections amplify and stabilize neural activity patterns and neural fields exhibit complex dynamics including memory characteristics of the prefrontal cortex. Neural fields are essentially laterally coupled recurrent neural networks. Motor primitives may also be represented by Amari neural fields of neuronal populations characterized by location x and time t with Cartesian motion defined by integrating $\dot{x} = v\cos\phi(t)$ and $\dot{y} = v\sin\phi(t)$:

$$\tau\frac{du(x,t)}{dt} == -u(x,t) + h + \sum_i S_i(x,t) + f_1(u(x,t))\left(\int w(\mathrm{x}-\mathrm{x}')f_2(u(x',t))dx' - w_{inh}\int f_2(u(x',t))dx'\right) \quad [16]$$

where τ=neural relation time, u(x,t)=neural activity, h=neural resting field, S_i(x,t)=neural input, $f_i(u) = \frac{1}{1-exp(-\beta_i(u-\theta_i))}$=sigmoid function, θ_i=threshold, β_i=sigmoidal slope, w(x,x')=connection weights, w_{inh}=feedback inhibition weight. Saccade initiation and targeting can be implemented in neural fields emulating the superior colliculus [116]. Neural fields of a competitive dynamical neural network may be adopted to determine the next saccade target through a winner-take-all strategy [117]. Similarly, visual attention may be implemented in neural fields to target salient locations in a visual map [118].

IX. ROBOTIC MANIPULATOR CONTROL

Neural networks have a long history in manipulator robotics beginning with cerenellar model articulation controller (CMAC) [119] diversifying into multilayer perceptrons, Hopfield nets and Boltzmann machines for kinematics, dynamics, trajectory generation and control [120,121]. They have been adopted as nonlinear compensation (of inertia matrix, coriolis/centrifugal and unmodelled terms on the inverse dynamics) in the computed torque control law [122] and identification of parameters and model reference adaptive control [123] of manipulators. Most commonly, they have been adopted to compute the inverse kinematic transformations [124,125] or learn the inverse dynamics model [126] of manipulators. Neural net control generally gives smoother performance than traditional computed torque control with reduced oscillations. Indeed, neural networks may be regarded as a form of model reference adaptive control in which the neural net approximates the nonlinear function [127]. Nevertheless, the neural network approach is superior in terms of its robustness to noise and unmodelled nonlinearities than adaptive control approaches as they incorporate a form of memory [128]. Neural network approaches can model learning of both arm kinematics and dynamics by mapping joint torques to cartesian kinematics [129]. Neural net learning involves the building and refining of internal motor models as lookup table representations encoded as neural network weights of multilayer perceptrons [130]. Recurrent neural networks such as the Hopfield network have been applied to kinematic control by minimising the weighted norm of joint velocity $\frac{1}{2}\dot{\theta}^T W\dot{\theta}$ [131]. The inverse kinematics problem lends itself to feedforward neural network-based solutions [132] especially for kinematically redundant manipulators which are ill-conditioned. Two neural networks may be deployed in conjunction with each other – an emulator learns the manipulator dynamics while the controller learns to control the emulator while both interact through the actual manipulator [133]. Multiple neural networks can model multiple finger Jacobians of a multifingered robot hand for a hybrid position/force controller from visual feedback in the presence of uncertainties [134].

Neural nets can learn the end effector positions of a robotic manipulator from visual input from cameras using the neural gas network [135]. A large number of target objects were randomly chosen and the manipulator configuration visually observed in pixel coordinate. Data pairs of target position errors in pixel coordinates mapped to the joint motor torque outputs enabled learning using a Hebbian rule with neural gas adaptation. A modified topographic Kohonen self-organising neural network trained by a teacherless Widrow-Hoff learning rule can learn visuomotor mappings for a robotic manipulator from binocular cameras [136]. The self-organising map has been applied to self-calibration of space-based manipulators to provide adaptability to the traditional inverse kinematics approach in response to slow parameter changes [137]. The radial basis function has also been applied to space manipulators to approximate to an optimal controller [138]. Neural networks have been used for both identification of the plant and control of a plant [139]. The commonest approach is to employ a multilayer perceptron emulator to learn the system plant dynamics and a separate multilayer perceptron controller to learn to control the emulator [140]. The emulator is trained to match the input-output characteristics of the plant. The real plant cannot be used as the error cannot be backpropagated through it – hence the emulator. The emulator generates the error between the emulator and controller to be backpropagated through the controller to update the controller weights. Neural networks are suited to multiple robotic applications including task planning, path planning and trajectory control [141].

X. CHEMICAL & MANUFACTURING PROCESSING

Most traditional chemical process control systems are based on conservation of momentum laws and mass and energy conservation laws for which predictive control, model-based control and adaptive control. A continuously stirred tank reactor model of an exothermic reaction includes product concentration C(t) and mixture temperature T(t) [142]:

$$\dot{C}_{prod} = \frac{q}{V}\Big(C_{prod}(0) - C_{prod}(t)\Big) - k_{prod}C_{prod}(t)e^{-E/RT(t)}$$

$$\dot{T}(t) = \frac{q}{V}\big(T(0) - T(t)\big) - k_1 C_{prod}(t)e^{-\frac{E}{RT(t)}} + k_2 q_c(t)(1 - e^{-\frac{k_3}{q_c(t)}})(T_c - T(t)) \quad [17]$$

where q=process flow rate, V=reactor volume, q_c=coolant flow rate, C(0)=inlet concentration, T(0)=inlet temperature, T_c=coolant temperature. The six reaction model parameters may be monitored for progress of the chemical reaction [143]. Neural networks learn empirical models of final outputs correlated with input parameters without knowledge of complex nonlinear physical and chemical laws. Neural networks can substitute for specific aspects of inverse model and predictive control [144], e.g. model predictive control may utilise a predictive model implemented as a feedforward neural network trained using an online Levenberg-Marquardt algorithm incorporated into the backpropagation algorithm such that $\Delta w = \frac{J^T(w)E(w)}{J^T(w)J(w)+\eta I}$ where J(w)=Jacobian matrix, E(w)=error, η=adaptation parameter [142]. Artificial neural networks may be applied for entire control of chemical engineering applications especially multilayer perceptron, radial basis function and support vector machine regression for sensor processing, fault detection, nonlinear control and process identification [145,146]. Neural networks may implement feedforward models trained by genetic algorithm for nonlinear predictive control of a chemical process yielding superior performance to a standard PI controller [147]. A feedforward model may be implemented with a neural network for controlling a chemical reaction using an inverse neural model controller yielding a more robust controller than a direct neural network inverse model [148]. A neural network nonlinear reference model trained online from input/output data permits adaptive control of a chemical process by a second neural network controller [149]. Chemical reaction optimisation neural networks may also be trained using evolutionary algorithms to simulate molecular populations and their interactions in a chemical reaction towards a global minimum potential energy state (fitness function) [150]. The knowledge-based artificial neural network (KBANN) exploits neural networks to refine propositional rules to control a plant [151]. The neural network is initially trained to implement a PID controller with tuned Ziegler-Nichols parameters. Randomised weights are added to the network which then trained by backpropagation to improve its control performance yielding faster learning.

Neural networks may be employed for autonomous control of general manufacturing tasks [152,153]. The potential field approach may be adapted for the control of movement of material by attractors to specific manufacturing machines [154]. Traditional automated manufacturing involved subtractive processes and assembly using expert systems [155] but more recent approaches include neural network scheduling [156]. An analogue network has been developed for job-shop scheduling [157] in which the number of neurons is mk with mk+mk(k-k/n) interconnection weights which grow linearly with the number of jobs. All manufacturing machines comprise of tooling that physically alter the shape of the product using rotating machinery. All are characterised by vibration for which turning is typical. Neural network models for process identification of a self-tuning controller during turning operations can measure self-excited vibration measurements by accelerometer as a proxy for surface finish friction to influence feed rate, tool speed and cut depth [158]. Neural networks such as LSTM are ideal for controlling additive manufacturing (AM) processes due to uncertainties in modelling parameters such as laser power, scan speed, hatch spacing and layer thickness (energy density, $E = \frac{P}{vhd}$ where P=laser power, v=scan speed, h=hatch spacing, d=layer thickness) for selective laser melting [159]. Thermomechanical (finite element) analysis of the AM process output and/or in-situ measured visual features (melt pool, pluming and spattering) output from the CAD model input is required to train the neural network model which is subsequently converted into an STL file. The chief challenges in neural network modelling are small data sets (though generative models such as autoencoders can synthetically augment data by generating a Gaussian distribution for sampling), lack of labelled data, lack of empirical guidelines for neural network hyperparameter selection and lack of knowledge of metallurgical quality to parameter combinations.

XI. CONCLUSIONS

In implementing computational electronics built from in-situ resources, there are major constraints:

(i) avoidance of the use of solid-state technology as solid state electronics manufacturing requires enormous infrastructure and complex processes

(ii) limitation on lunar material resources

(iii) emphasis on multi-use devices, i.e. vacuum tubes.

Our rationale inevitably leads to the adoption of neural network controllers as our general computational architecture. We have shown that the neural network is thus a versatile approach to autonomous control systems

encountered in-situ resource utilisation applications – surveying, mining, chemical processing and manufacturing tasks – with a special emphasis on autonomous rover navigation which is required for mining and transport. The robustness of our resultant neural network for goal-directed navigation must be tested more fully. If successful, the neural network architecture will be constructed in hardware and tested in reality rather than simulation (subject to the transition from simulation to reality requiring fine tuning), first in transistor-based op-amp form then in vacuum tube-based op-amp form. Thence, 3D printing of the circuitry can be undertaken.

APPENDIX: TURING MACHINES

Computation involves transforming an input data string into an output string using a finite set of steps – this finite set of steps constitutes an algorithm. An algorithm is thus a finite sequence of well-defined procedures that transform an input into an output. Equivalently, an algorithm is computable by a Turing machine. A Turing machine is a general computational model based on an automaton – it comprises an infinitely long tape divided into discrete cells, each cell containing one input symbol from a finite alphabet. More formally, the Turing machine is a finite-state machine comprising a read/write head mounted onto an infinitely long tape divided into discrete squares. Turing machine may be codified as a finite state automaton on the form: TM=(Q,X,A,D,q_0, B,F) where Q=finite set of states, X=input alphabet, A=tape alphabet, $D: Q \times A \rightarrow Q \times A \times \{L, R\}$=output function, $q_0 \in Q$ =initial state, $B \in A (B \notin X)$=blank symbol, $F \subset Q$=finite set of final states. The Turing machine sequentially reads an infinitely long digital tape of cells. Symbols from a finite alphabet may be inscribed on the tape which are read in sequence by the read/write head. The initial tape encodes a set of input data. The read/write head scans the tape, reading each symbol and over-writing with an output symbol. It can move left or right, reading and writing a symbol at each move. The read/write head incorporates a finite memory of internal state transitions constituting the specific program of the Turing machine. The motion of the read/write head – the behaviour of the Turing machine - is determined by the symbol inscribed on each cell of the tape and the internal state of the machine. The Turing machine's behaviour is determined by the current state of the machine, the input symbol being read on the tape, and a table of instructions (program). The program determines the output symbol to be written, the movement direction of the head to read the next input, and the next machine state. The tape's symbol is overwritten with a new output symbol and the read/write head moves to the next scanning position left or right according to the Turing machine's state transition function. The resulting tape encodes a set of output data. The unproven Turing-Church thesis asserts that the mechanical computations of a Turing machine define an algorithm (program). Different Turing machines are specified by different state transition functions. This simple machine implements a mathematical function that converts its input into an output – the Turing machine's mechanical procedure encapsulates the algorithm concept as a finite sequence of simple operations. Any specific Turing machine may be encoded as an input tape so a universal Turing machine can emulate any specific Turing machine, i.e. a universal Turing machine can compute any computable function given the appropriate algorithm:

UTM={Q,{0,1},{0,1,B},D,q_0,B,{q'}).

A magnetic tape is one physical instantiation of the Turing machine tape comprising a polymer tape coated with a thin magnetic ferrite forming narrow tracks. It uses an electromagnetic write head to magnetise the tape to encode binary data. The key to increasing the data density is shrinking the magnetic grains and the use of signal processing/servocontrol algorithms for the write head. Magnetic tape as a data storage medium has progressed enormously from a density of 225 bits/cm^2 in the first IBM tape drive (1952) and has reached 30 Gb/cm^2 today. Tape storage, although slow to retrieve, is highly reliable (error rate ~10^{-5} that of hard drives).

AKNOWLEDGEMENTS

The author would like to thank his student Liam Gritters for the Webot modelling.

REFERENCES

[1] Ellery A (2022) "Is electronics fabrication feasible on the Moon?" in press with *Proc ASCE Earth & Space Conf*, Colorado School of Mines, Denver

[2] Taylor G & Martel L (2003) "Lunar prospecting" *Advances in Space Research* **31** (11), 2403-2412

[3] Ellery A (2020) "Sustainable in-situ resource utilisation on the Moon" *Planetary & Space Science* **184**, 104870

[4] Kock C (1997) "Computation and the single neuron" *Nature* **385**, 207-210

[5] Hansen L & Salamon P (1990) "Neural network ensembles" *IEEE Trans Pattern Analysis & Machine Intelligence* **12** (10), 993-1001

[6] Jacobs R, Jordan M, Nowlan S, Hinton G (1991) "Adaptive mixtures of local experts" *Neural Computation* **3**, 79-87

[7] Jordan M & Jacobs R (1993) "Hierarchical mixtures of experts and the EM algorithm" MIT AI Memo No 1440

[8] Furber S, Galluppi F, Temple S, Plana L (2014) "Spinnaker project" *Proc IEEE* **102**(5), 652-665

[9] Rhodes O (2020) "Brain-inspired computing becomes complete" *Nature* **586** (Oct), 364-366
[10] Bouvier M, Valentian A, Mesquida T, Rummens F, Reyboz M, Vianello E, Beigne E (2019) "Spiking neural networks hardware implementations and challenges: a survey" *ACM J Emerging Technologies in Computing Systems* **15** (2), article no 22
[11] Feng J (2001) "Is the integrate-and-fire model good enough? – a review" *Neural Networks* **14**, 955-975
[12] Feng J & Brown D (2000) "Integrate-and-fire models with nonlinear leakage" *Bulletin Mathematical Biology* **62**, 467-481
[13] Plesser H & Geisel T (2001) "Signal processing by means of noise" *Neurocomputing* **38-40**, 307-312
[14] Joubert A, Belhadj B, Temam O, Heliot R (2012) "Hardware spiking neuron design: analogue or digital?" *Proc IEEE World Congress on Computational Intelligence*, Brisbane, Australia
[15] Izhikevich E (2000) "Simple model of spiking neurons" *IEEE Trans Neural Networks* **14** (6), 1569-1572
[16] Aitkenhead M, McDonald A (2002) "Neural network-based obstacle navigation animat in a virtual environment" *Engineering Applications of Artificial Intelligence* **15**, 229-239
[17] Sapounaki M, Kakarountas A (2019) "High performance neuron for artificial neural network based on Izhikevich model" *Proc 29th Int Symp on Power & Timing Modelling, Optimisation & Simulation*, Rhodes, Greece, 29-34
[18] Indiveri G, Linares-Barranco B, Hamilton T, van Schaik A, Etienne-Cummings R, Delbruck T, Liu S-C. Dudek P, Hafliger P, Renaud S, Schemmel J, Cauwenberghs G, Arthur J, Hynna K, Folowosele F, Saighi S, Serrano-Gotarredona T, Wijekoon J, Wang Y, Boahen K (2011) "Neuromorphic silicon neuron circuits" *Fronters in Neuroscience* **5**, article 73
[19] Siegelmann H (1999) "Stochastic analog networks and computational complexity" *J Complexity* **15**, 451-475
[20] Langlois N, Miche P, Benshair A (2000) "Analogue circuits of a learning spiking neuron model" *Proc IEEE Int Joint Conf Neural Networks* **4**, 485-489
[21] Bradley W, Mears R (1996) "Backpropagation learning using positive weights for multilayer optoelectronic neural networks" *Proc 9th Annual Meeting IEEE Lasers & Electro-Optics Society*, 294-295
[22] Xu R, Lv P, Xu F, Shi Y (2021) "Survey of approaches for implementing optical neural networks" *Optics & Laser Technology* **136**, 106787
[23] Bogaerts W, Perez D, Capmany J, Miller D, Poon J, Englund D, Morichetti F, Melloni A (2020) "Programmable photonic circuits" *Nature* **586** (Oct), 207-216
[24] Wertstein G, Ozcan A, Gigan S, Fan S, Englund D, Soljacic M, Denz C, Miller D, Psaltis D (2020) "Inference in artificial intelligence with deep optics and photonics" *Nature* **588** (Dec), 39-47
[25] Ashtiani F, Geers A. Aflarouni F (2022) "On-chip photonic deep neural network for image classification" *Nature* **606** (Jun), 501-506
[26] Hughes T, Minkov M, Shi Y, Fan S (2018) "Training of photonic neural networks through in situ backpropagation and gradient measurement" *Optica* **5** (7), 864-871
[27] Neftci E (2018) "Data and power efficient intelligence with neuromorphic learning machines" *iScience* **5** (Jul), 52-68
[28] Burr G (2019) "Role for optics in AI hardware" *Nature* **569** (May), 199-200
[29] Feldmann J, Youngblood N, Wright C, Bhaskaran H, Pernicke W (2019) "All-optical spiking neurosynaptic networks with self-learning capabilities" *Nature* **569** (May), 208-214
[30] Mennel L, Symonowicz J, Wachter S, Polyushkin D, Molina-Mendoza A, Mueller T (2020) "Ultrafast machine vision with 2D material neural network image sensors" *Nature* **579** (Mar), 62-66
[31] Lin X, Rivenson Y, Yardimci N, Veli M, Jarrahi M, Ozcan A (2018) "All optical machine learning using diffractive deep neural networks" *Science* **361**, 1004-1008
[32] Nashanovich D (2018) "Electronics and photonics combined" *Nature* **536** (Apr), 316-317
[33] Misra J & Saha I (2010) "Artificial neural networks in hardware: a survey of two decades of progress" *Neurocomputing* **74**, 239-250
[34] Woodburn R, Reekie M, Murray A (1994) "Pulse stream circuits for on-chip learning in analogue VLSI neural networks" *Proc IEEE Int Symp Circuits & Systems*, 103-106
[35] Soda K, Pack D (2004) "Simple hardware implementation of neural networks for instruction in analogue electronics" *Proc American Society for Engineering Education Annual Conf & Exposition*, paper 1096
[36] Ruckert U (2002) "ULSI architectures for artificial neural networks" *IEEE Micro* (May/Jun), 10-18
[37] Omondi A, Rajapakse J (eds) (2006) "FPGA Implementations of Neural Networks" *Springer Publishers*
[38] Krips M, Lammert T, Kummert A (n.d.) (2002) "FPGA implementation of a neural network for a real-time hand tracking system" *Proc 1st IEEE Int Workshop Electronic Design, Test and Applications*
[39] Zhu J, Sutton P (2003) "FPGA implementations of neural networks – a survey of a decade of progress" *Lecture Notes in Computer Science* **2778**, 1062-1066
[40] Savich A, Moussa M, Areibi S (2007) "Impact of arithmetic representation on implementing MLP-BP on

FPGAs: a study" *IEEE Trans Neural Networks* **18**(1), 240-252
[41] Hao V (2017) "General neural network hardware architecture on FPGA" arXiv:1711.05850 [cs.CV]
[42] Dinu A, Cirstea M, Cirstea S (2009) "Direct neural networks hardware implementation algorithm" *IEEE Trans Industrial Electronics* **57**(5), 1845-1848
[43] Kim D (2000) "Implementation of fuzzy logic controller on the reconfigurable system" *IEEE Trans Industrial Electronics* **47**(3), 703-715
[44] Pei J, Deng L, Song S, Zhao M, Zhang Y, Wu S, Wang G, Zou Z, Wu Z, He W, Chen F, Seng N, Wu S, Wang Y, Wu Y, Yang Z, Ma C, Li G, Han W, Li H. Wu H, Zhao R, Xie Y, Shi L (2019) "Towards artificial general intelligence with hybrid Tianjic chip architecture" *Nature* **572** (Aug), 106-111
[45] Siegelmann H & Sontag E (1991) "Turing computability with neural nets" *Applied Mathematical Letters* **4**, 77-80
[46] Siegelmann H & Sontag E (1993) "Analogue computation via neural networks" *Proc 2nd Israel Symp on Theory & Computing Systems*, 98-107
[47] Balcazar J, Gavalda R, Siegelmann H (1997) "Computational power of neural networks: a characterization in terms of Kolmogorov complexity" *IEEE Trans Information Theory* **43** (4), 1175-1183
[48] Siegelmann H (1995) "Computation beyond the Turing limit" *Science* **268**, 545-458
[49] Rado T (1962) "On non-computable functions" *Bell Systems Technical J* **41**, 877-884
[50] Van Leeuwen J & Wiedermann J (2000) "Turing machine paradigm in contemporary computing" in *Mathematics Unlimited – 2001 & Beyond*, Lecture Notes in Computer Science, 1139-1159
[51] Wegner P, Eberbach E & Burgin M (2012) "Computational completeness of interaction machines and Turing machines" in *Turing-100* (ed. Voronkov A), EPiC series, EasyChair Publishing, **10**, 405-414
[52] Appeltant L, Soriano M, van der Sande G, Danckaert J, Massar S, Dambre J, Schrauwen B, Mirasso C, Fischer I (2011) "Information processing using a single dynamical node as complex system" *Nature Communications* **2**, 468
[53] Cabessa J (2014) "Interactive evolving recurrent neural networks are super-Turing" *Lecture Notes in Computer Science* **8681** (ed. Wermter S et al), 57-64
[54] Freitas N, Esposito M (2022) "Maxwell demon that can work at macroroscopic scales" *arXiv:2204.09466v1 [cond-mat.stat-mech] 20 Apr 2022*
[55] Brooks R (1991) "Intelligence without representation" *Artificial Intelligence* **47** (1-3), 139-159
[56] Ellery A (2005) "Robot-environment interaction – the basis for mobility in planetary micro-rovers" *Robotics & Autonomous Systems* **51** (1), 29-39
[57] Parberry I (1994) *Circuit Complexity and Neural Networks*, MIT Press Foundations of Computing, Cambridge, MA
[58] Yamashita Y, Nakamura Y (2007) "Neuron circuit model with smooth nonlinear output function" *Proc Int Symp Nonlinear Theory & its Applications*, Vancouver, pp. 11-14
[59] Hasslacher B & Tilden M (1995) "Living machines" *Robotics & Autonomous Systems* **15**, 143-169
[60] Sahin E & Franks N (2002) "Measurement of space: from ants to robots" *EPSRC/BBRC Proc Int Workshop Biologically Inspired Robotics: Legacy of W Grey Walter*, Bristol, 241-247
[61] Cross M, Ellery A, Qadi A (2013) "Estimating terrain parameters for a rigid wheeled rover using neural networks" *J Terramechanics* **50** (3), 165-174
[62] Gharbi R & Mansoori G (2005) "Introduction to artificial intelligence applications in petroleum exploration and production" *J Petroleum Science & Engineering* **49**, 93-96
[63] Mohaghegh S (2005) "Recent developments in the application of artificial intelligence in petroleum engineering" *J Petroleum Technology* **57** (4), SPE-89033
[64] Ellery A (2014) "Artificial intelligence through symbolic connectionism – a biomimetic rapprochement" in *Biomimetic Technologies Vol II: Actuators, Robotics & Integrated Systems* (ed. Ngo D), Woodhead Publishing
[65] Srinivasan M (1992) "How bees exploit optic flow: behavioural experiments and neural network models" *Philosophical Transactions Royal Society* **B337**, 253-258
[66] Mallot H, Bulthoff H, Little J, Bohrer S (1991) "Inverse perspective mapping simplifies optical flow computation and obstacle detection" *Biological Cybernetics* **64**, 177-185
[67] Schmidhuber J (2015) "Deep learning in neural networks: an overview" *Neural Networks* **61**, 85-117
[68] Braitenburg V (1984) "Vehicles: Experiments in Synthetic Psychology" *MIT Press*
[69] Kon M & Plaskota L (2000) "Information complexity of neural networks" *Neural Networks* **13** (3), 365-375
[70] Huang S-C & Huang Y-F (1991) "Bounds on the number of hidden neurons in multilayer perceptrons" *IEEE Trans Neural Networks* **2** (1), 47-55
[71] Andonie R (2012) "Psychological limits of neural computation" in *Dealing with Complexity: A Neural Network Approach*, Springer Publishers, 252-263
[72] La Rocca M & Perna C (2014) "Designing neural networks for modelling biological data: a statistical perspective" *Mathematical Biosciences & Engineering* **11** (2), 331-342

[73] Golombek M and Rapp D (1997) "Size-frequency distributions of rocks on Mars and Earth analogue sites: implications for future landed missions," *J Geophysical Research* **102** (E2), 4117-4129
[74] Ellery A (2022) "Bootstrapping neural electronics from lunar resources for in-situ artificial intelligence applications" submitted to *British Computer Society Special Group on Artificial intelligence Conf*, Cambridge, UK
[75] Sasikala K, Ganesh E, Kumar P (2019) "Design of adaptive analogue self-tuning filters" *J Architecture & Technology* **11** (6), 74-81
[76] Bai Y-T, Wang X-Y, Jin X-B, Zhao Z-Y, Zhao B-H (2020) "Neuron-based Kalman filter with nonlinear autoregressive model" *Sensors* **20**, paper no 299
[77] Revach G, Shlezinger N, Ni X, Escoriza L, van Sloun R, Eldar Y (2022) "KalmanNet: neural network aided Kalman filtering for partially known dynamics" *IEEE Trans Signal Processing* **70** (Mar), 1532-1547
[78] Ruck D, Rogers S, Kabrisky M, Oxley M, Suter B (1990) "Multilayer perceptron as an approximation to a Bayes optimal discriminant function" *IEEE Trans Neural Networks* **1** (4), 296-298
[79] Ruck D, Rogers S, Kabrisky M, Maybeck P, Oxley M (1992) "Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons" *IEEE Trans Pattern Analysis & Machine Intelligence* **14** (6), 686-691
[80] Millidge B, Tschankz A, Seth A, Buckley C (2021) "Neural Kalman filtering" arXiv:2102.10021c2 [cs.NE] 29 Apr 2021
[81] Perez-Ortiz A, Gere F, Eck D, Schmidhuber J (2003) "Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets" *Neural Networks* **16** (2), 241-250
[82] Shareef A, Zhu Y, Musavi M, Shen B (2007) "Comparison of MLP neural networks and Kalman filter for localisation in wireless sensor networks" *Proc 19th IASTED Int Conf Parallel & Distributed Computing & Systems*, 323-330
[83] St-Pierre M, Gingras D (2004) "Neural network-based data fusion for vehicle positioning in land navigation system" *SAE Technical paper 2004-01-0752*
[84] Wilson R, Finkel L (2009) "Neural implementation of the Kalman filter" *Proc 22nd Int Conf on Neural Information Processing Systems*, Vancouver, 2062-2070
[85] Saeedi S, Paull L, Trentini M, Li H (2011) "Neural network-based multiple robot simultaneous localization and mapping" *IEEE Trans Neural Networks* **22**(12), 2376-2387
[86] Martinelli G & Perfetti R (1991) "Circuit theoretic approach to the backpropagation learning algorithm" *IEEE Int Symp Circuits & Systems* **3**, 1481-1484
[87] Larson S & Ellery A (2015) "Trainable analogue neural network with application to lunar in-situ resource utilisation" *Proc Int Astronautics Federation Congress*, Jerusalem, IAC-15-D3.3.6
[88] 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
[89] Ellery A (2022) "Bootstrapping neural electronics from lunar resources for in-situ artificial intelligence applications" submitted to *British Computer Society Special Group on Artificial intelligence Conf*, Cambridge, UK
[90] Trullier O, Meyer J-A (1997) "Biomimetic navigation models and strategies in animats" *AI Communications* **10** (2), 79-92
[91] Scholkopf B, Mallot H (1995) "View-based cognitive mapping and path planning" *Adaptive Behaviour* **3** (3), 311-348
[92] Moller R, Maris M, Lambrinos D (1999) "Neural model of landmark navigation in insects" *Neurocomputing* **26** (2), 801-808
[93] Gould J (1990) "Honeybee cognition" *Cognition* **37**, 83-103
[94] Kim S, Hermundstad A, Romani S, Abbott L, Jayaraman V (2018) "Generation of stable heading representations in diverse visual scenes" *Nature* **576** (Dec), 126-131
[95] Payne H, Lynch G, Aronov D (2021) "Neural representations of space in the hippocampus of a food-caching bird" *Science* **373** (Jul), 343-347
[96] Milford M & Wyeth G (2009) "Persistent navigation and mapping using a biologically inspired SLAM system" *Int J Robotics Research* **29** (9), 1131-1153
[97] Wyeth G & Milford M (2009) "Spatial cognition for robots" *IEEE Trans Robotics & Automation Magazine* (Sep), 24-32
[98] Milford M, Wyeth G, Prasser D (2004) "RatSLAM: a hippocampal model for simultaneous localisation and mapping" *Proc IEEE Int Conf Robotics & Automation*, 403-408
[99] Wood E, Dudchenko P (2021) "Navigating space in the mammalian brain" *Science* **372** (May), 913-914
[100] Ginosar G, Aljadeff J, Burak Y, Sompolinsky H, Las L, Ulanovsky N (2021) "Locally ordered representation of 3D space in the entorhinal cortex" *Nature* **596** (Aug), 404-409
[101] Milford M, Wyeth G (2010) "Hybrid robot control and SLAM for persistent navigation and mapping" *Robotics & Autonomous Systems* **58**, 1096-1104
[102] Milford M, Schulz R, Prasser D, Wyeth J, Wiles J (2007) "Learning spatial concepts from RatSLAM representations" *Robotics & Autonomous Systems* **55**, 403-410

[103] Franz M, Mallot H (2000) "Biomimetic robot navigation" *Robotics & Autonomous Systems* **30**, 133-153
[104] Gaussier P, Joulain C, Banquet J, Lepretre S, Revel A (2000) "Visual homing problem: an example of robotics/biology cross fertilisation" *Robotics & Autonomous Systems* **30**, 155-180
[105] Schmajuk B, Blair H (1993) "Place learning and the dynamics of spatial navigation: a neural network approach" *Adaptive Behaviour* **1** (5), 353-385
[106] Butz M, Shirinov E, Reif K (2010) "Self-organising sensorimotor maps plus internal motivations yield animal-like behaviour" *Adaptive Behaviour* **18** (3-4), 315-337
[107] Caluwaerts K, Staffa M, N'Guyen S, Grand C, Dolle L, Facre-Felix A, Girard B, Khamassi M (2012) "Biologically inspired meta-control navigation system for the Psikharpax rat robot" *Bioinspiration & Biomimetics* **7**, 025009
[108] Cuperlier N, Quoy M, Gaussier P (2007) "Neurobiologically inspired mobile robot navigation and planning" *Frontiers in Neurorobotics* **1** (3), 1-15
[109] Millan J (1995) "Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot" *Robotics & Autonomous Systems* **15**, 275-299
[110] Hoydal O, Skytoen R, Andersson O, Moser M-B, Moser E (2019) "Object-vector coding in the medial entorhinal cortex" *Nature* **568** (Apr), 400-404
[111] Hadsell R, Erkin A, Sermanet P, Scoffier M, Muller U, LeCun Y (2008) "Deep belief net learning in a long-range vision system for autonomous off-road driving" *Proc IEEE/RSJ Int Conf Intelligent Robots & Systems*, paper no 4651217
[112] Schoner G, Dose M, Engels C (1995) "Dynamics of behaviour: theory and applications for autonomous robot architectures" *Robotics & Autonomous Systems* **16**, 213-245
[113] Quoy M, Moga S, Gaussier P (2003) "Dynamical neural networks for planning and low-level robot control" *IEEE Trans Systems Man & Cybernetics A: Systems & Humans* **33** (4), 523-532
[114] Schoner G, Dose M (1992) "Dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion" *Robotics & Autonomous Systems* **10**, 253-267
[115] Erlhagen W & Bicho E (2006) "Dynamic neural field approach to cognitive robotics" *J Neural Engineering* **3**, R36-R54
[116] Wilimzig C, Schneider S, Schoner G (2006) "Time course of saccadic decision-making: dynamic field theory" *Neural Networks* **19**, 1059-1074
[117] Shibata T, Vijayakumar S, Conradt J, Schaal S (2001) "Biomimetic oculomotor control" *Adaptive Behaviour* **9** (3/4), 189-207
[118] Vitay J, Rougier N (2005) "Using neural dynamics to switch attention" *IEEE Int Conf Neural Networks*, paper no 1556384
[119] Albus J (1975) "New approach to manipulator control: the cerebellar model articulation controller (CMAC)" *Trans AMS J Dynamic Systems, Measurement & Control* **97**, 220-233
[120] Horne B, Jamshidi M, Vaduee N (1990) "Neural networks in robotics: a survey" *J Intelligent & Robotic Systems* **3**, 51-66
[121] Kung S-Y, Hwang J-N (1989) "Neural network architectures for robotic applications" *IEEE Trans Robotics & Automation* **5** (3), 641-657
[122] Ozaki T, Suzuki T, Furuhashi T, Okuma S (1991) "Trajectory control of robotic manipulators using neural networks" *IEEE Trans Industrial Electronics* **38** (3), 195-202
[123] Narendra K, Parthasarathy K (1990) "Identification and control of dynamical systems using neural networks" *IEEE Trans Neural Networks* **1** (1), 4-27
[124] Rao D (1995) "Neural networks in robotics and control: some perspectives" *Proc IEEE/IAS Int Conf Industrial Automation & Control*, Hyderabad, India, 451-456
[125] Alsina P & Gehlot N (1996) "Robot inverse kinematics: a modular neural network approach" *Proc IEEE Circuits & Systems*, 631-634
[126] Miller T, Glanz F, Kraft G (1987) "Application of a general learning algorithm to the control of robotic manipulators" *Int J Robotics Research* **6** (2), 84-97
[127] Patino H & Liu D (2000) "Neural network-based model reference adaptive control system" *IEEE Trans Systems Man & Cybernetics B: Cybernetics* **30** (1), 198-204
[128] Kraft L & Campagna D (1990) "Comparison between CMAC naural network control and two traditional adaptive control systems" *IEEE Control Systems Magazine* (April), 36-43
[129] Atkeson C (1989) "Learning arm kinematics and dynamics" *Annual Reviews Neuroscience* **12**, 157-183
[130] Morris A & Mansor A (1997) "Finding the inverse kinematics of manipulator arm using artificial neural network with lookup table" *Robotica* **15**, 617-625
[131] Wang J, Hu Q, Jiang D (1999) "Lagrangian network for kinematic control of redundant robot manipulators" *IEEE Trans Neural Networks* **10** (5), 1123-1131
[132] Dermatas E, Nearchou A, Aspragathos N (1996) "Error-back-propagation solution to the inverse kinematic problem of redundant manipulators" *Robotics & CIM* **12** (4), 303-310
[133] Nguyen D, Widrow B (1990) "Neural networks for self-learning control systems" *IEEE Control Systems Magazine* (Apr), 18-23

[134] Zhao Y, Cheah C (2009) "Neural network control of multifingered robot hands using visual feedback" *IEEE Trans Neural Networks* **20** (5), 758-767
[135] Martinetz T & Schulten K (1993) "Neural network with Hebbian-like adaptation rules learning visuomotor coordination of a PUMA robot" *Proc IEEE Int Conf Neural Networks* **2**, 820-822
[136] Martinetz T, Ritter H, Schulten K (1990) "Three-dimensional neural net for learning visuomotor coordination of a robotic arm" *IEEE Trans Neural Networks* **1** (1), 131-136
[137] De Angulo V & Torras C (1997) "Self-calibration of a space robot" *IEEE Trans Neural Networks* **8** (4), 951-963
[138] Gorinevsky A, Kapitanovsky A, Goldenberg A (1996) "Radial basis function network architecture for nonholonomic motion planning and control of free-flying manipulators" *IEEE Trans Robotics & Automation* **12** (3), 491-496
[139] Narendra K & Parthasarathy K (1990) "Identification and control of dynamical systems using neural networks" *IEEE Trans Neural Networks* **1** (1), 4-27
[140] Nguyen D & Widrow B (1990) "Neural networks for self-learning control systems" *IEEE Control Systems Magazine* (Apr), 18-23
[141] Kung A-Y & Hwang J-N (1989) "Neural network architectures for robotic applications" *IEEE Trans Robotics & Automation* **5** (5), 641-657
[142] Yu H, Zhang Z (2006) "Predictive control based on neural networks of the chemical process" *Proc 25th Chinese Control Conf*, Harbin, 1143-1147
[143] Gasteiger J, Zupan J (1993) "Neural networks in chemistry" *Angewandte Chemie International Edition in English* **32**, 503-527
[144] Hussain A (1999) "Review of the applications of neural networks in chemical process control – simulation and online implementation" *Artificial Intelligence in Engineering* **13**, 55-68
[145] Himmelbrau D (2000) "Applications of artificial neural networks in chemical engineering" *Korean J Chemical Engineering* **17** (4), 373-392
[146] Pirdashti M, Curteanu S, Kamangar H, Hassim M, Khatami A (2013) "Artificial neural networks: applications in chemical engineering" *Reviews in Chemical Engineering* **29** (4), 205-239
[147] Draeger A, Engell S, Ranke H (1995) "Model predictive control using neural networks" *IEEE Control Systems* (Oct), 61-66
[148] Hussain A, Kittisupakorn P, Daosud W (2001) "Implementation of neural network-based inverse model control strategies on an exothermic reactor" *ScienceAsia* **27**, 41-50
[149] Douratsos I, Gomm B (2006) "Neural network based model reference adaptive control for processes with time delay" *Int J Information & Systems Sciences* **3** (1), 161-179
[150] Yu J, Lam A, Li V (2015) "Evolutionary artificial neural network based on chemical reaction optimisation" *arXiv: 1502.00193v1 [cs.NE] 1 Feb 2015*
[151] Scott G, Shavlik J, Ray H (1992) "Refining PID controllers using neural networks" *Neural Computation* **4**, 746-757
[152] Huang S & Zhang H-C (1994) "Artificial neural networks in manufacturing: concepts, applications and perspectives" *IEEE Trans Components, Packaging & Manufacturing Technology – Part A* **17** (2), 212-228
[153] Abdelhameed M & Tolbah F (2002) "Recurrent neural network-based sequential controller for manufacturing automated systems" *Mechatronics* **12**, 617-633
[154] Ueda K, Hatono I, Fujii N, Vaario J (2000) "Reinforcement learning approaches to biological manufacturing systems" *CIRP Annals – Manufacturing Technology* **49** (1), 343-346
[155] Gupta S, Regli W, Das D, Nau D (1997) "Automated manufacturability analysis: a survey" *Research in Engineering & Design* **9**, 168-190
[156] Arzi Y, Iaroslavitz L (1999) "Neural network-based adaptive production control system for a flexible manufacturing cell under a random environment" *IIE Trans* **31**, 217-230
[157] Zhou D, Cherkassky V, Baldwin T, Olson D (1991) "Neural network approach to job-shop scheduling" *IEEE Trans Neural Networks* **2** (1), 175-179
[158] Kazeem I, Zangana N (2007) "Neural network based real time controller for turning process" *Jordan J Mechanical & Industrial Engineering* **1** (1), 43-55
[159] Qi X, Chen G, Li Y, Cheng X, Li C (2019) "Applying neural network-based machine learning to additive manufacturing: current applications, challenges and future perspectives" *Engineering* **5**, 721-729