In-Materio Extreme Learning Machines

Benedict. A. H. Jones[1 [0000−0002−1924−0560]], Noura Al Moubayed[2 [0000−0001−8942−355X]], Dagou A. Zeze[1 [0000−0002−6596−5490]], and Chris Groves[1 [0000−0003−2402−1618]]

[1] Department of Engineering, Durham University, Durham, DH1 3LE, UK
chris.groves@durham.ac.uk, benedict.jones@durham.ac.uk
[2] Department of Computer Science, Durham University, Durham, DH1 3LE, UK

Abstract. Nanomaterial networks have been presented as a building block for unconventional in-Materio processors. Evolution in-Materio (EiM) has previously presented a way to configure and exploit physical materials for computation, but their ability to scale as datasets get larger and more complex remains unclear. Extreme Learning Machines (ELMs) seek to exploit a randomly initialised single layer feed forward neural network by training the output layer only. An analogy for a physical ELM is produced by exploiting nanomaterial networks as material neurons within the hidden layer. Circuit simulations are used to efficiently investigate diode-resistor networks which act as our material neurons. These in-Materio ELMs (iM-ELMs) outperform common classification methods and traditional artificial ELMs of a similar hidden layer size. For iM-ELMs using the same number of hidden layer neurons, leveraging larger more complex material neuron topologies (with more nodes/electrodes) leads to better performance, showing that these larger materials have a better capability to process data. Finally, iM-ELMs using virtual material neurons, where a single material is re-used as several virtual neurons, were found to achieve comparable results to iM-ELMs which exploited several different materials. However, while these Virtual iM-ELMs provide significant flexibility, they sacrifice the highly parallelised nature of physically implemented iM-ELMs.

Keywords: Evolution in-Materio · Evolvable Processors · Extreme Learning Machines · Material Neurons · Virtual Neurons · Classification.

1 Introduction

The inevitable slowdown in traditional CMOS technology improvement [6] has led to a growing interest in alternative and unconventional computing techniques. Evolution in-Materio (EiM) is one such paradigm which attempts to leverage a material's inherent properties and exploit them for computation, these materials were initially referred to as Configurable Analogue Processors [29]. EiM uses an Evolutionary Algorithm (EA) to optimise external stimuli and other parameters such that the material can perform a target application. Materials which might have limited uses in conventional electronic devices, may in fact provide

sufficiently complex and interesting characteristics to be exploited as an EiM device. EiM processors have been used to achieve a range of applications such as logic gates [23,27,4,2] and for classification [5,28,40]. Notably, EiM attempts to exploit a system's intrinsic physics with only a few configurable parameters, which is important for the complexity engineering approach [15] and helps prevent an over parametrised system. If this relative computational simplicity is paired with low power materials, then in-Materio processors present a possible candidate for edge case computing [30]. Recent analysis of EiM systems has established fundamental good practices [21] and enhancements [20] to the EiM paradigm. However, even with these improvements, methods to scale in-Materio processing systems to larger and more complex datasets remains lacking. Without this, any real-world adoption remains unlikely.

Generally, within a 'material' or 'in-Materio' processor, some nanomaterial is placed on a microelectrode such that stimuli and data can be applied as a voltage. Some form of hardware interface is necessary to apply and read voltages to the material, often controlled from a PC (which in the case of EiM hosts the EA that optimises the system). Devices operating in such a manner include liquid crystals [17,18], metallic nanoparticles [2,16], single walled carbon nanotubes [27,28], and dopant networks [4,35]. However, in-Materio processors could be configured using any external stimuli such as light [39] or radio waves [25]. In fact, any medium with complex intrinsic characteristics which can be interfaced with and leveraged could be used as an in-Materio processor. Networks of common electronic devices (resistors, diodes, etc.) can provide interesting tunable dynamics [22] and can be realised physically or investigated using reliable and fast SPICE (Simulation Program with Integrated Circuit Emphasis) simulations [21,20].

Extreme Learning Machines (ELM) and Reservoir Computing (RC) present a good analogy for in-Materio processors since both involve the exploitation of random networks. These systems depend on the underlying assumption that the randomised network/reservoir will produce useful and often higher dimensional output states that are used to process the data more successfully. Notably, within these fields of research it is generally assumed that the network/reservoir remains fixed after its inception. However, previous work has shown that a small amount of stochastic optimisation can improve a systems performance [7,43,13]. RC was developed from recurrent neural networks and is generally employed to process temporal data. Physical RCs [38] could lead to low power, efficient and fast systems which can operate at 'the edge'. Examples include the use of circuit (anti-parallel diode) based non-linear neuron [22], memristive network [1,12] and magnetic spintronic [8] based reservoirs. ELMs were developed from single layer feed forward neural networks (SLFN) and are generally employed to process non-temporal data [19]. Examples of physical implementations of ELM remain sparse but include memristor based networks [1] and photonic systems [32,26]. Their remains significant opportunity to develop both classical and quantum substrates [31] for both RC and ELM.

Here we present a method of exploiting nanomaterial networks as 'material neurons', grouping them into a SLFN's hidden layer and training them as an

ELM. To enable efficient analysis, random diode-resistor networks are used as a proxy for physical nanomaterials, which are solved using fast, reliable SPICE simulations. The performance of these 'in-Materio ELMs' on several common machine learning datasets is examined for various hidden layer sizes and physical material network topologies. They are found to outperform other common classification techniques and traditional (artificial) ELMs of a similar size. Finally, drawing from the work showing EiM processors can be configured via external voltage stimuli and other parameters, we implement a material 're-use' system, whereby a single material neuron is used to create several virtual material neurons. These physical neuron based ELMs provide a scalable in-Materio unconventional computing method whereby the intrinsic properties of a material (or medium) can be exploited in a highly parallelisable way.

2 Background

2.1 Evolution in-Materio Processors

EiM exploits nanomaterials using an optimisation algorithm such that they can perform useful tasks. Since in-Materio processors are analogue and generally lack an analytical model to describe their electrical characteristics, derivative-free optimisation algorithms are used, rather than gradient based algorithms [28]. EAs are a subset of evolutionary computing [36], consisting of population-based metaheuristic search algorithms, making them ideal for EiM. Many types of EAs have been used for EiM such as Evolutionary Strategies [9], Genetic Algorithms [2], Differential Evolution [40,28]. In particular, Differential Evolution (DE) is an easily implemented and effective optimisation algorithm [37,36] which only requires a few robust control variables [33] and is attractive for real parameter optimisation [10].

Such a DE algorithm can be combined with a material simulation (developed in [21]) to allow for significantly faster testing and analysis of EiM processors than physical manufacturing and experimentations would allow. Full details are available elsewhere [37,10], but briefly, the DE algorithm uses the greedy criterion that involves evaluating the fitness of each member of a generation's population, with those members of the population with better fitness being more likely to proceed to the next generation. The characteristics of the population therefore change gradually over time due to the random mutation of characteristics and cross-over with other population members. Every member of the population is represented by a vector of decision variables $\mathbf{X}$. This decision vector contains configuration parameters which the EA optimises each generation. A basic EiM processor might commonly have a decision vector defined as:

$$\mathbf{X} = [\, V_{c1},\ V_{c2},\ ...,\ V_{cP},\ l_1,\ l_2,\ ...,\ l_R \,]^T\,, \tag{1}$$

where T is the vector transpose. Here, the included configuration parameters are as follows: Input "configuration" voltage stimuli $V_{cp} \in [-5, 5]\, V$ applied to a node p, where the total number of configuration nodes is P. These configuration

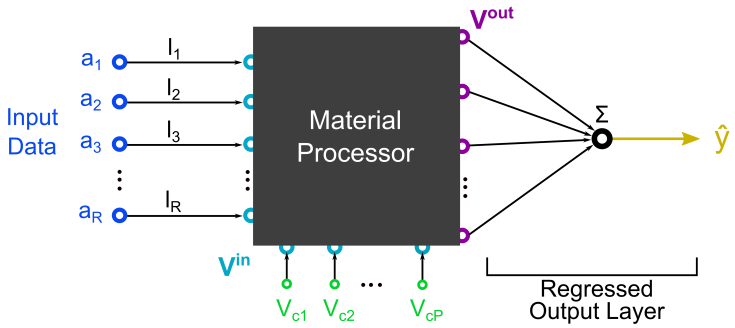

Fig. 1. Illustration of a typical monolithic EiM processor's structure. Input data is applied to the material as voltages. The output voltages (i.e., material processor output states) are regressed to produce an output layer which predicts the class ($\hat{y}$) of the processed data instances. Input weights (l_r) can be used to scale the input data voltages, and configurable voltage stimuli (V_c) can manipulate the processor's behaviour [21].

voltages have been shown to both introduce a bias but also alter the decision boundary of an EiM classifier [21]; therefore, these effect how the materials IV characteristic is exploited which could be analogous to altering the material's 'activation function'. Input weights $l_r \in [-1, 1]$, are used to scale the input voltages V_r^{in} applied at the data driven input electrodes r due to a corresponding input attribute a_r, such that:

$$V_r^{in}(k) = l_r \times a_r(k), \tag{2}$$

where k is a given data instance and the total number of data driven input electrodes is R. The structure of this type of typical EiM processor is shown in Figure 1, and illustrates how the configurable parameters effect the material processor's operation.

While an evolved output layer and threshold can be used to interpret the materials output voltages and assign a label to a particular data instance which has been processed [21], it has been found that a regressed output layer is more successful at evaluating and exploiting a material's output voltage states [20]. This regressed layer is generated when evaluating a population member on the training dataset and must be maintained and updated in tandem with the EA's population.

Generally, for classification, a dataset D, containing R attributes $a_1, a_2, \ldots, a_R$, is split into two subsets: a training set D^{train} and a test set D^{test}. During each generation, every member of the population is evaluated using the training data and an associated fitness is calculated using the EA's objective function. The objective function Φ has commonly been the classification error, but other types of fitnesses may be desirable, such as binary cross entropy [20]. The best population member p_{best} is tracked during training and the test set is used to evaluate the final best population member.

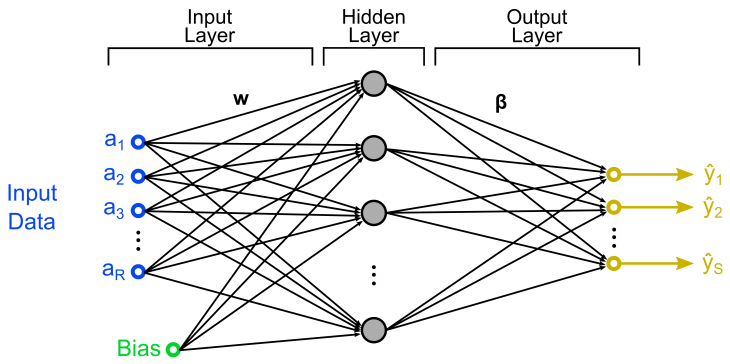

Fig. 2. Basic structure of an artificial single-hidden layer feed forward network used as an Extreme Learning Machine.

2.2 Extreme Learning Machines

ELMs generally consist of a SLFN, as seen in Figure 2. They operate by assigning random weights and biases to the input and hidden layers respectively [19]. These parameters are fixed and remain unchanged during training. Here, the hidden layer neurons use the sigmoid activation function. Whilst many activation functions exist [34], the sigmoid function is widely used [42] and can achieve good performance in most cases [3]. The only parameters learned are the weights (and sometimes biases) associated with the output layer, done during the training phase. Therefore, ELMs converge significantly faster than traditional artificial neural network algorithms, such as back propagation. ELMs have been shown to perform well and are more likely to reach a global optimum than systems with networks which have all parameters trained [42]. Specifically, ELM systems achieve fast training speeds with good generalisation capability.

Keeping our nomenclature consistent with §2.1, consider K data instances, where a particular data instance k is defined by its inputs $\mathbf{a}_k$ and its target outputs $\mathbf{y}_k$. Here, we define a particular instances' input containing R attributes as $\mathbf{a}_k = [a_{k1}, a_{k2}, ..., a_{kR}]^T$, and its corresponding target containing S outputs as $\mathbf{y}_k = [y_{k1}, y_{k2}, ..., y_{kS}]^T$. The predicted outputs $\hat{\mathbf{y}}$ from an ELM with N hidden neurons can be expressed as:

$$\hat{\mathbf{y}}_k = \sum_{n=1}^{N} \boldsymbol{\beta}_n g(\mathbf{w}_n \cdot \mathbf{a}_k + b_n) = \sum_{n=1}^{N} \boldsymbol{\beta}_n h_{kn} \ . \qquad k = 1, ..., K \tag{3}$$

where $\mathbf{w}_n = [w_{n1}, w_{n2}, ..., w_{nR}]^T$ is the weight vector connecting the n^{th} hidden neuron and the input neurons, $\boldsymbol{\beta}_n = [\beta_{n1}, \beta_{n2}, ..., \beta_{nS}]^T$ is the weight vector connecting the n^{th} hidden neuron and the output neurons, b_n is the bias of the n^{th} hidden neuron, $g(x)$ is the activation function of the hidden layer neurons, and h_{kn} is a hidden layer neurons' output. A SLFN with enough hidden neurons can approximate these K samples such that $\sum_{n=1}^{N} \|\hat{\mathbf{y}}_n - \mathbf{y}_k\| = 0$ (universal

approximation capability), hence a set of $\boldsymbol{\beta}_n$, $\mathbf{w}_n$ and b_n must exist so that [19]:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}\,, \tag{4}$$

where $\mathbf{H} = \{h_{kn}\}$ $(k = 1, ..., K$ and $n = 1, ..., N)$ is the hidden layer output matrix, $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_K]^T$ is the matrix of target outputs, and $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, ..., \boldsymbol{\beta}_N]^T$ is the matrix of output weights. Having randomised and fixed the input layer, the output layer is then learnt during training using the training data subset D^{train}. The output weights $\boldsymbol{\beta}$ are traditionally obtained by the Moore-Penrose inverse. Therefore, the smallest norm least-squares solution is:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{Y}\,, \tag{5}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose inverse of matrix $\mathbf{H}$. The final solution is then tested on the test set D^{test} to provide a uniform evaluation of the system.

Often, many randomly initialised networks are considered and the network size is incrementally increased. Various methods of calculating the output layer, adjusting network structure, and increasing convergence speed have been proposed [42]. Specifically, we highlight work producing an RR-ELM algorithm [24] which optimises the output layer using ridge regression rather than the Moore-Penrose method described above. The RR-ELM algorithm is shown to have good generalisation and stability, while also reducing adverse effects caused by perturbation or multicollinearity - properties likely to be useful in physical systems.

3 In-Materio Extreme Learning Machines

Traditionally, as described in §2.1, EiM processors have used one-to-one mapping. Here, we refer to this as a typical monolithic EiM processor, where each attribute is applied as a voltage to a corresponding input node. However, as a dataset becomes more complex, with more attributes, the size of the material network would need to physically grow. We postulate that in real microelectrode-based nanomaterial processors, larger networks might lead to fewer 'interactions' between distance electrodes, leading to poorer performance i.e., material processors may struggle to scale as the data does.

In order to overcome this problem we can take inspiration from SLFNs, as shown in Figure 2. Specifically, the Configurable Analogue Processors or 'material processors' used in previous EiM work can instead be viewed as a complex physical neuron. These 'material neurons' can then be grouped into a Hidden Layer (HL) to produce a typical SLFN like structure. The output voltages from these material neurons are the HL's output states; we note that a material generally projects the applied input data voltages to a higher dimensional number of output voltages. The remaining question then becomes how to translate the input data into usable voltage which can be fed into the material neurons' input data electrodes/nodes i.e., an input layer.

We propose a *directly connected* input layer network structure as shown in Figure 3. Consider a network with M material neurons, each of which contains

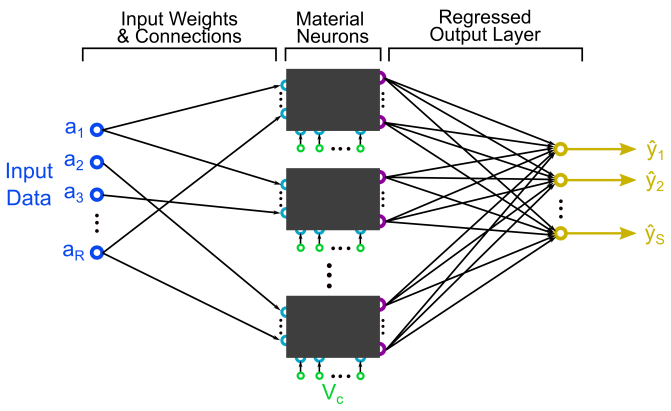

Fig. 3. Diagram of a structured network of material processors which are exploited as hidden layer neurons for an ELM using a *directly connected* input layer, where M material neurons make up the hidden layer.

J input data electrodes/nodes. Each data voltage input V^{in} is the product of a weight and a selected attribute. Therefore, the voltage applied to a particular material m and node j is defined as:

$$V^{in}_{mj} = \ l^m_j \times a_{C^m_j}\,, \tag{6}$$

where $C^m_j \in \{1, 2, ..., R\}$ defines which attribute a_r is being passed to a particular material's data input node, and $l^m_j \in [-1, 1]$ is that connection's associated weight. This is a relatively simple input layer scheme, not requiring the introduction of an activation function, ensuring that the computation within the system is carried out by the material neuron only and that the hardware voltage limits are note exceeded. The relatively few number of parameters helps the system comply with the complexity engineering approach [15] and potentially benefit from concepts such as weight agnostic and minimal neural network topologies which have been found to be beneficial in ANNs [14].

The system can be defined using a vector of decision variables $\mathbf{X}$ as discussed in Equation 1. Expanding this to include all the discussed adjustable parameters, the new structured network's decision vector can be defined as:

$$\mathbf{X} = [V^1_{c1}, ..., V^1_{cP}, l^1_1, ..., l^1_J, C^1_1, ..., C^1_J, \ ...., \ V^M_{c1}, ..., V^M_{cP}, l^M_1, ..., l^M_J, C^M_1, ..., C^M_J]^T. \tag{7}$$

Now, any single material neuron based SLFN or population p of material neuron based SLFNs (i.e., multiple initialisations of $\mathbf{X}$) can be randomly generated and trained as an ELM network using Algorithm 1. We refer to this method of combining a physical neuron based SLFN and ELM training as an in-Materio ELM (iM-ELM). In this paper, the output layer of an iM-ELM is trained using ridged regression rather then the Moore-Penrose inverse detailed in §2.2.

Finally, we highlight the possibility of re-using a single nanomaterial network as several 'virtual' neurons. The basis of this method stems from EiM processors

Algorithm 1: Method for in-Materio ELM.

Initialise random population of solutions p;
Train population p on D^{train};
Evaluate population using the test data D^{test};

whereby a wide variety of operations can be discovered by configuring the external stimuli of a single nanomaterial network. Therefore, by randomly initialising the different configurable parameters, but using only a single material, several virtual material neurons can be produced. Each of these will manifest their own unique internal IV characteristics which the ELM system will attempt to exploit. Here, we refer to such a network as a Virtual iM-ELM.

4 Problem Formulation

4.1 Simulated Material Networks

A circuit SPICE model is used to generate Diode Random Networks (DRNs) which behaves as a complex and exploitable network, and acts as a proxy for a non-linear nanomaterial. These networks contain: voltage driven input data nodes (*in-nodes*), voltage driven configuration stimuli (*c-nodes*), and measured output voltage nodes (*out-nodes*) calculated using a DC analysis. The DRN consists of randomly orientated diodes and series resistors between every node pair. The rapid changes in conductivity when a diode turns on allows for complex non-linear IV characteristics which can be exploited for classification. The DRN is physically realisable using discrete circuit components and its properties are common in nanomaterials. An example of a five node (i.e., electrode) material is given in Figure 4. We note that other IV characteristics or circuit components could be used to create a variety of 'material networks' for different types of analysis. 1N4148PH Signal Diodes are used, and the resistance of the interconnecting resistors are uniformly randomly selected between $\in [10, 100]\, k\Omega$.

4.2 Classification Tasks

The performance of the iM-ELM systems are compared against several common machine learning datasets which can be found on the UCI repository [11]. These include: the Pima Indians Diabetes Database (diabetes) dataset containing 8 attributes, 768 data instances and 2 classes. The wine (wine) dataset containing 13 attributes, 178 instances and 3 classes. The Australian Credit Approval (aca) dataset containing 14 attributes, 690 instances and 2 classes. The Wisconsin Diagnostic Breast Cancer (wdbc) dataset containing 30 attributes, 569 instances and 2 classes. These datasets are more complex (i.e., contain more attributes) than have been previously used, specifically within Evolution in-Materio based literature [20,41,27], but remain small enough (i.e., relatively few data instances) to ensure comprehensive analysis without excessive simulation times.

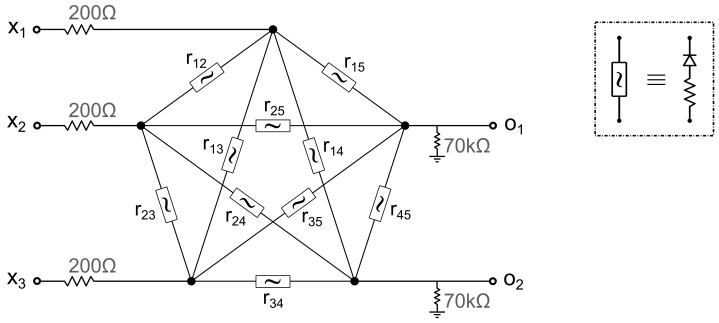

Fig. 4. An example five node DRN material, where each node is connected to every other node via a resistor and diode. In this example, two nodes are behaving as outputs (o_1, o_2) and three nodes as inputs (x_1, x_2, x_3) which could be allocated as either data driven or configuration stimuli voltages.

The datasets are randomly split into a training (D^{train}) and test (D^{test}) subset using a $70\% - 30\%$ split respectively. The datasets are normalised (using D^{train}) and then scaled to the max/min allowed hardware voltages $\in [-5, 5]\, V$. The performance of these datasets was considered using some simple (default) python sklearn classification methods and the results are shown in Table 1.

5 Results and Discussion

Considering the model developed in §3, iM-ELMs of incremental sizes were analysed. Specifically, the number of material neurons in the HL was increased from one to fifteen (beyond which the results plateau). For each HL size, thirty different random seeds were used to generate the material neurons within thirty iM-ELMs. The same thirty seeds are used for each network size incrementation, meaning that each system continues to include the same material neurons that were used in its corresponding previous smaller networks. Therefore, we can consider the change in performance of the iM-ELM networks as they are enlarged. For each iM-ELM, a 'population' of 100 randomly generated decision vectors are considered (i.e., randomly initialised input layer and configuration parameters),

Table 1. Test accuracy results for the datasets when using several common classification methods. The best accuracy achieved for each dataset is highlighted in bold.

Dataset	Classification Method				
	Ridge Regression	Logistic Regression	Random Forest	SVM (linear)	SVM (poly)
diabetes	0.7576	0.7619	0.7403	**0.7662**	0.6970
wine	0.9815	0.9815	**1.000**	0.9630	0.9815
aca	0.8357	0.8502	**0.8599**	0.8213	0.8261
wdbc	0.9357	**0.9591**	0.9298	0.9532	**0.9591**

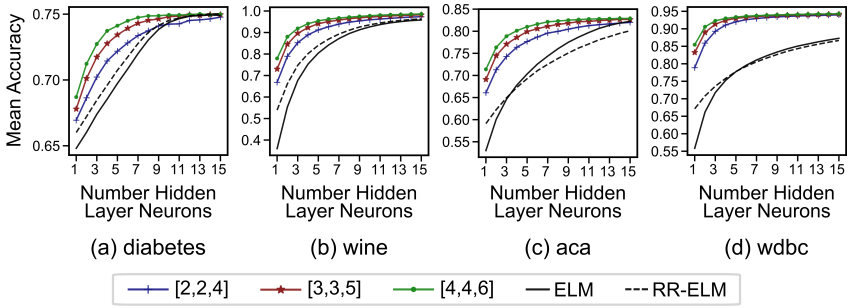

Fig. 5. Mean test accuracy of all the (30 systems, each with 100 parameter initialisations) in-Materio ELMs for each hidden layer size increment used to classify the (a) diabetes, (b) wine, (c) aca and (d) wdbc datasets. Three different material neuron topologies are considered ([No. *in-nodes*, No. *c-nodes*, No. *out-nodes*]), and these are compared to the mean accuracy of 3000 traditional artificial ELMs and RR-ELMs.

which was observed to provide a good insight into performance and maintain reasonable simulation times. The mean test accuracy from these thirty systems each with 100 parameter initialisation is shown for each HL size in Figure 5. Recall from §4.1 that these material neurons' consist of a fully interconnected network containing three main classes of nodes: input voltage nodes for data (*in-nodes*), input voltage nodes for configuration/stimuli altering the material neurons behaviour (*c-nodes*), and output voltage nodes (*out-nodes*). Notably, the *directly connected* input layer connects each data input node to only a single data attribute; so, if too few neurons are in use, then not all data attributes may be 'connected'. The experiment is performed with three increasingly larger material neuron network topologies: (i) materials containing two *in-nodes*, two *c-nodes* and four *out-nodes* denoted by [2,2,4], (ii) materials containing three *in-nodes*, three *c-nodes* and five *out-nodes* denoted by [3,3,5], (iii) materials containing four *in-nodes*, four *c-nodes* and six *out-nodes* denoted by [4,4,6]. This will provide some initial insight on the effect of scaling the size of (well connected) materials. The performance of these iM-ELMs is plotted against the mean accuracy of 3000 artificial (Moore-Penrose) ELMs and (Ridge Regression) RR-ELMs, selected because it matches the total 'computational expense' (i.e., total number of data instances) used over the 30 iM-ELMs systems.

On average, the iM-ELMs considered outperformed the artificial ELM and RR-ELM systems of equivalent network sizes. The simulated 'material' neurons are successfully generating useful, higher dimensional output states which the ELM algorithm can exploit, and these are out-performing their artificial neuron counterparts. As more material neurons are operated in parallel, the mean classification accuracy improves. Notably, the larger and more complex material neuron topologies (i.e., when using more input, configuration, and output nodes/electrodes) achieve higher mean accuracies for iM-ELMs with the same size of HL. This in turn means that fewer neurons are required within the SLFN HL to achieve comparable results with networks leveraging less capable neurons.

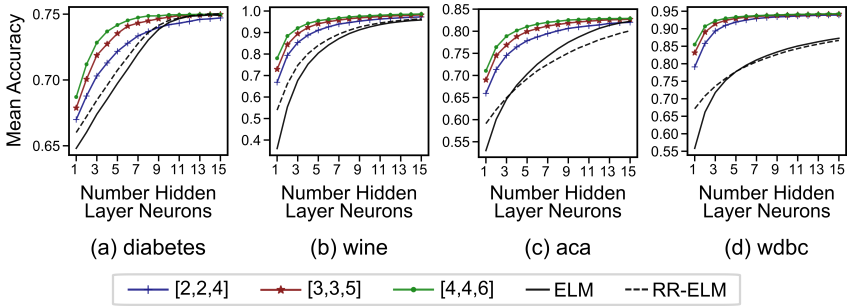

Fig. 6. Mean test accuracy of all the (30 systems, each with 100 parameter initialisations) Virtual in-Materio ELMs for each hidden layer size increment used to classify the (a) diabetes, (b) wine, (c) aca and (d) wdbc datasets. Three different material neuron topologies are considered ([No. *in-nodes*, No. *c-nodes*, No. *out-nodes*]), and these are compared to the mean accuracy of 3000 traditional artificial ELMs and RR-ELMs.

The best accuracy achieved, across all HL sizes, for the different material neuron topologies and datasets, is shown in Table 2. The iM-ELMs discussed can significantly outperform some of the common classification methods presented in Table 1. Indeed, the best iM-ELMs also compare favourably with the traditional artificial ELM networks, and in the case of the wdbc dataset the iM-ELMs can achieve a 1.76% increase in the best obtained accuracy.

As discussed in §3 any single material could be re-used as a virtual material neuron. Ideally, the different randomised parameters (input weights, connections and configuration voltages) enable the different virtual neurons to behave in a sufficiently independent manner. To investigate this, the previous analysis is repeated i.e., generating thirty SLFN for each HL size, each with 100 random initialisations. However, now each HL contains several virtual neurons which are generated from only a single material. The mean accuracy of these Virtual iM-ELMs is plotted against the size of the SLFN in Figure 6, and the best ever achieved accuracies are shown in Table 2. The Virtual iM-ELMs have a near identical average performance to the previously discussed iM-ELMs systems that exploited several different materials. This suggests that the type of material neuron used here (i.e., the simulated circuit based DRN non-linear network) can successfully produce several virtual instances, achieved by exploiting the

Table 2. Best accuracy achieved from the different systems, from across the different hidden layer sizes. The best accuracy for each dataset is highlighted in bold.

Dataset	iM-ELM			Virtual iM-ELM			ELM	RR-ELM
	[2,2,4]	[3,3,5]	[4,4,6]	[2,2,4]	[3,3,5]	[4,4,6]		
diabetes	0.7922	0.7922	**0.7965**	0.7879	0.7922	**0.7965**	**0.7965**	0.7922
wine	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
aca	0.8792	0.8792	**0.8889**	0.8841	0.8841	0.8792	**0.8889**	0.8744
wdbc	**0.9708**	**0.9708**	**0.9708**	**0.9708**	**0.9708**	0.9649	0.9532	0.9532

wide range of current-voltage characteristics which can be tuned and selected by the voltage stimuli and input layer respectively. These Virtual iM-ELMs are significantly more flexible, only requiring one material substrate to create an SLFN. However, by 're-using' a single material, the systems loses its ability to benefit from the highly parallelisable structure.

These results provide guidance on how to operate in-Materio processors in parallel to process much more complex datasets then would have previously been possible with only a single monolithic material. However, further work is needed to consider how well these systems can scale to the much larger datasets commonly found in state of the art machine learning problems.

6 Conclusion

In this paper, material networks are exploited as physical material neurons to implement a single hidden layer feed forward network (SLFN) which was trained as an Extreme Learning Machine (ELM). The input data was passed to the physical neurons using a *directly connected* input layer which ensured physical hardware limits were obeyed and that 'computation' within the system was carried out by the materials only. Complex diode-resistor networks were simulated to provide a convenient, fast and reliable proxy to nanomaterial based Configurable Analogue Processors, used as the physical neurons. This enabled the efficient investigation of these in-Materio ELMs (iM-ELMs) when classifying several datasets of varying complexity. It was found that these iM-ELMs could significantly outperform other common classification methods, as well as traditional (artificial) ELMs. The complex current-voltage characteristics of the materials are successfully being exploited to leverage them as physical neurons, which outperform traditional artificial neurons. As the size of the material topology increases (i.e., the number of nodes/electrodes used in the material network), the performance of iM-ELMs with similar hidden layer sizes improves; showing the capability of the material neurons to process data is increasing. As more material neurons are used in parallel, the average classification performance improves rapidly before plateauing.

Drawing from previous work with Evolution in-Materio processors, which show that a single nanomaterial network can be tuned for a range of operations, we present a method to re-use a single material as several 'virtual' physical neurons. These Virtual iM-ELMs which leveraged only a single physical material performed comparably to the iM-ELMs which used several different physical material neurons. This suggests that our circuit based 'materials' can achieve a wide range of physical properties which are successfully exploited as different virtual neurons. While this forgoes the benefits of parallelised operation, it grants more flexibility when creating larger SLFN and when designing the required physical hardware interface.

These physical analogue neurons have the potential to produce efficient in-Materio ELMs which can exploit the non-differentiable, complex characteristics presented by a nanomaterial. We anticipate that, when implemented using physical substrates, highly parallelisable and fast ELMs will be produced.

Acknowledgements This work was supported by the Engineering and Physical Sciences Research Council [EP/R513039/1].

References

1. Bennett, C., Querlioz, D., Klein, J.O.: Spatio-Temporal learning with arrays of analog nanosynapses. In: Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures, NANOARCH 2017. pp. 125–130 (2017). https://doi.org/10.1109/NANOARCH.2017.8053708
2. Bose, S.K., Lawrence, C.P., Liu, Z., Makarenko, K.S., van Damme, R.M.J., Broersma, H.J., van der Wiel, W.G.: Evolution of a designless nanoparticle network into reconfigurable Boolean logic. Nature Nanotechnology **10**(12), 1048–1052 (Dec 2015). https://doi.org/10.1038/nnano.2015.207
3. Cao, W., Gao, J., Ming, Z., Cai, S.: Some Tricks in Parameter Selection for Extreme Learning Machine. IOP Conference Series: Materials Science and Engineering **261**, 012002 (Oct 2017). https://doi.org/10.1088/1757-899X/261/1/012002
4. Chen, T., van Gelder, J., van de Ven, B., Amitonov, S.V., de Wilde, B., Euler, H.C.R., Broersma, H., Bobbert, P.A., Zwanenburg, F.A., van der Wiel, W.G.: Classification with a disordered dopant-atom network in silicon. Nature **577**(7790), 341–345 (Jan 2020). https://doi.org/10.1038/s41586-019-1901-0
5. Clegg, K.D., Miller, J.F., Massey, M.K., Petty, M.C.: Practical issues for configuring carbon nanotube composite materials for computation. In: 2014 IEEE International Conference on Evolvable Systems. pp. 61–68 (Dec 2014). https://doi.org/10.1109/ICES.2014.7008723
6. Conte, T.M., DeBenedictis, E.P., Gargini, P.A., Track, E.: Rebooting Computing: The Road Ahead. Computer **50**(1), 20–29 (Jan 2017). https://doi.org/10.1109/MC.2017.8
7. Dale, M., Stepney, S., Miller, J., Trefzer, M.: Reservoir computing in materio: An evaluation of configuration through evolution. 2016 IEEE Symposium Series on Computational Intelligence (SSCI) (2016). https://doi.org/10.1109/SSCI.2016.7850170
8. Dale, M., Evans, R.F.L., Jenkins, S., O'Keefe, S., Sebald, A., Stepney, S., Torre, F., Trefzer, M.: Reservoir Computing with Thin-film Ferromagnetic Devices. arXiv:2101.12700 [cond-mat] (Jan 2021)
9. Dale, M., Stepney, S., Miller, J.F., Trefzer, M.: Reservoir computing in materio: A computational framework for in materio computing. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 2178–2185 (May 2017). https://doi.org/10.1109/IJCNN.2017.7966119
10. Das, S., Suganthan, P.N.: Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation **15**(1), 4–31 (Feb 2011). https://doi.org/10.1109/TEVC.2010.2059031
11. Dheeru Dua, E.K.T.: UCI Machine Learning Repository. http://archive.ics.uci.edu/ml (2017)
12. Du, C., Cai, F., Zidan, M., Ma, W., Lee, S., Lu, W.: Reservoir computing using dynamic memristors for temporal information processing. Nature Communications **8**(1) (2017). https://doi.org/10.1038/s41467-017-02337-y
13. Eshtay, M., Faris, H., Obeid, N.: Metaheuristic-based extreme learning machines: A review of design formulations and applications. International Journal of Machine Learning and Cybernetics **10**(6), 1543–1561 (Jun 2019). https://doi.org/10.1007/s13042-018-0833-6

14. Gaier, A., Ha, D.: Weight Agnostic Neural Networks. arXiv:1906.04358 [cs, stat] (Sep 2019)
15. Ganesh, N.: Rebooting Neuromorphic Hardware Design – A Complexity Engineering Approach. arXiv:2005.00522 [cs] (Sep 2020)
16. Greff, K., van Damme, R.M.J., Koutnik, J., Broersma, H.J., Mikhal, J.O., Lawrence, C.P., van der Wiel, W.G., Schmidhuber, J.: Using neural networks to predict the functionality of reconfigurable nano-material networks. In: International Journal on Advances in Intelligent Systems. vol. 9, pp. 339–351. IARIA (Jan 2017)
17. Harding, S., Miller, J.: Evolution in materio: A tone discriminator in liquid crystal. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753). vol. 2, pp. 1800–1807 Vol.2 (Jun 2004). https://doi.org/10.1109/CEC.2004.1331114
18. Harding, S., Miller, J.F.: Evolution In Materio: Evolving Logic Gates in Liquid Crystal. International Journal of Unconventional Computing **3**, 243–257 (2007)
19. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing **70**(1), 489–501 (Dec 2006). https://doi.org/10.1016/j.neucom.2005.12.126
20. Jones, B.A.H., Al Moubayed, N., Zeze, D.A., Groves, C.: Enhanced methods for Evolution in-Materio Processors. In: 2021 International Conference on Rebooting Computing (ICRC). pp. 109–118 (Nov 2021). https://doi.org/10.1109/ICRC53822.2021.00026
21. Jones, B.A.H., Chouard, J.L.P., Branco, B.C.C., Vissol-Gaudin, E.G.B., Pearson, C., Petty, M.C., Al Moubayed, N., Zeze, D.A., Groves, C.: Towards Intelligently Designed Evolvable Processors. Evolutionary Computation pp. 1–23 (Mar 2022). https://doi.org/10.1162/evco_a_00309
22. Kan, S., Nakajima, K., Takeshima, Y., Asai, T., Kuwahara, Y., Akai-Kasaya, M.: Simple Reservoir Computing Capitalizing on the Nonlinear Response of Materials: Theory and Physical Implementations. Physical Review Applied **15**(2), 024030 (Feb 2021). https://doi.org/10.1103/PhysRevApplied.15.024030
23. Kotsialos, A., Massey, M.K., Qaiser, F., Zeze, D.A., Pearson, C., Petty, M.C.: Logic gate and circuit training on randomly dispersed carbon nanotubes. International journal of unconventional computing. **10**(5-6), 473–497 (Sep 2014)
24. Li, G., Niu, P.: An enhanced extreme learning machine based on ridge regression for regression. Neural Computing and Applications **22**(3), 803–810 (Mar 2013). https://doi.org/10.1007/s00521-011-0771-7
25. Linden, D.: A system for evolving antennas in-situ. In: Proceedings Third NASA/DoD Workshop on Evolvable Hardware. EH-2001. pp. 249–255 (Jul 2001). https://doi.org/10.1109/EH.2001.937968
26. Lupo, A., Butschek, L., Massar, S.: Photonic Extreme Learning Machine based on frequency multiplexing. Optics Express **29**(18), 28257 (Aug 2021). https://doi.org/10.1364/OE.433535
27. Massey, M.K., Kotsialos, A., Qaiser, F., Zeze, D.A., Pearson, C., Volpati, D., Bowen, L., Petty, M.C.: Computing with carbon nanotubes: Optimization of threshold logic gates using disordered nanotube/polymer composites. Journal of Applied Physics **117**(13), 134903 (Apr 2015). https://doi.org/10.1063/1.4915343
28. Massey, M.K., Kotsialos, A., Volpati, D., Vissol-Gaudin, E., Pearson, C., Bowen, L., Obara, B., Zeze, D.A., Groves, C., Petty, M.C.: Evolution of Electronic Circuits using Carbon Nanotube Composites. Scientific Reports **6**(1), 32197 (Oct 2016). https://doi.org/10.1038/srep32197

29. Miller, J., Downing, K.: Evolution in materio: Looking beyond the silicon box. In: Proceedings 2002 NASA/DoD Conference on Evolvable Hardware. pp. 167–176. IEEE Comput. Soc, Alexandria, VA, USA (2002). https://doi.org/10.1109/EH.2002.1029882
30. Morán, A., Canals, V., Galan-Prado, F., Frasser, C.F., Radhakrishnan, D., Safavi, S., Rosselló, J.L.: Hardware-Optimized Reservoir Computing System for Edge Intelligence Applications. Cognitive Computation (Feb 2021). https://doi.org/10.1007/s12559-020-09798-2
31. Mujal, P., Martínez-Peña, R., Nokkala, J., García-Beni, J., Giorgi, G.L., Soriano, M.C., Zambrini, R.: Opportunities in Quantum Reservoir Computing and Extreme Learning Machines. Advanced Quantum Technologies **4**(8), 2100027 (2021). https://doi.org/10.1002/qute.202100027
32. Ortín, S., Soriano, M.C., Pesquera, L., Brunner, D., San-Martín, D., Fischer, I., Mirasso, C.R., Gutiérrez, J.M.: A Unified Framework for Reservoir Computing and Extreme Learning Machines based on a Single Time-delayed Neuron. Scientific Reports **5**(1), 14945 (Oct 2015). https://doi.org/10.1038/srep14945
33. Pedersen, M.E.H.: Good Parameters for Differential Evolution (2010)
34. Ratnawati, D.E., Marjono, Widodo, Anam, S.: Comparison of activation function on extreme learning machine (ELM) performance for classifying the active compound. AIP Conference Proceedings **2264**(1), 140001 (Sep 2020). https://doi.org/10.1063/5.0023872
35. Ruiz-Euler, H.C., Alegre-Ibarra, U., van de Ven, B., Broersma, H., Bobbert, P.A., van der Wiel, W.G.: Dopant Network Processing Units: Towards Efficient Neural-network Emulators with High-capacity Nanoelectronic Nodes. arXiv:2007.12371 [cs, stat] (Jul 2020)
36. Sloss, A.N., Gustafson, S.: 2019 Evolutionary Algorithms Review. arXiv:1906.08870 [cs] (Jun 2019)
37. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization **11**(4), 341–359 (Dec 1997). https://doi.org/10.1023/A:1008202821328
38. Tanaka, G., Yamane, T., Héroux, J.B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., Hirose, A.: Recent advances in physical reservoir computing: A review. Neural Networks **115**, 100–123 (Jul 2019). https://doi.org/10.1016/j.neunet.2019.03.005
39. Viero, Y., Guérin, D., Vladyka, A., Alibart, F., Lenfant, S., Calame, M., Vuillaume, D.: Light-Stimulatable Molecules/Nanoparticles Networks for Switchable Logical Functions and Reservoir Computing. Advanced Functional Materials **28**(39), 1801506 (2018). https://doi.org/10.1002/adfm.201801506
40. Vissol-Gaudin, E., Kotsialos, A., Groves, C., Pearson, C., Zeze, D., Petty, M.: Computing Based on Material Training: Application to Binary Classification Problems. In: 2017 IEEE International Conference on Rebooting Computing (ICRC). pp. 1–8. IEEE, Washington, DC (Nov 2017). https://doi.org/10.1109/ICRC.2017.8123677
41. Vissol-Gaudin, E., Kotsialos, A., Groves, C., Pearson, C., Zeze, D., Petty, M., Al Moubayed, N.: Confidence Measures for Carbon-Nanotube / Liquid Crystals Classifiers. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8 (Jul 2018). https://doi.org/10.1109/CEC.2018.8477779
42. Wang, J., Lu, S., Wang, S.H., Zhang, Y.D.: A review on extreme learning machine. Multimedia Tools and Applications (May 2021). https://doi.org/10.1007/s11042-021-11007-7

43. Zhu, Q.Y., Qin, A.K., Suganthan, P.N., Huang, G.B.: Evolutionary extreme learning machine. Pattern Recognition **38**(10), 1759–1763 (Oct 2005). https://doi.org/10.1016/j.patcog.2005.03.028