Towards Intelligently Designed Evolvable Processors

Benedict A. H. Jones benedict.jones@durham.ac.uk
Department of Engineering, Durham University, Durham, DH1 3LE, UK

John L. P. Chouard
Department of Engineering, Durham University, Durham, DH1 3LE, UK

Bianca C. C. Branco
Department of Engineering, Durham University, Durham, DH1 3LE, UK

Eléonore G. B. Vissol-Gaudin
Department of Engineering, Durham University, Durham, DH1 3LE, UK

Christopher Pearson
Department of Engineering, Durham University, Durham, DH1 3LE, UK

Michael C. Petty m.c.petty@durham.ac.uk
Department of Engineering, Durham University, Durham, DH1 3LE, UK

Noura Al Moubayed noura.al-moubayed@durham.ac.uk
Department of Computer Science, Durham University, Durham, DH1 3LE, UK

Dagou A. Zeze d.a.zeze@durham.ac.uk
Department of Engineering, Durham University, Durham, DH1 3LE, UK

Chris Groves chris.groves@durham.ac.uk
Department of Engineering, Durham University, Durham, DH1 3LE, UK

Abstract

Evolution-in-Materio is a computational paradigm in which an algorithm reconfigures a material's properties to achieve a specific computational function. This paper addresses the question of how successful and well performing Evolution-in-Materio processors can be designed through the selection of nanomaterials and an evolutionary algorithm for a target application. A physical model of a nanomaterial network is developed which allows for both randomness, and the possibility of Ohmic and non-Ohmic conduction, that are characteristic of such materials. These differing networks are then exploited by differential evolution, which optimises several configuration parameters (e.g., configuration voltages, weights, etc.), to solve different classification problems. We show that ideal nanomaterial choice depends upon problem complexity, with more complex problems being favoured by complex voltage dependence of conductivity and vice versa. Furthermore, we highlight how intrinsic nanomaterial electrical properties can be exploited by differing configuration parameters, clarifying the role and limitations of these techniques. These findings provide guidance for the rational design of nanomaterials and algorithms for future Evolution-in-Materio processors.

Keywords

Differential evolution, evolution in-materio, evolutionary materials, evolvable processors, material networks, nano computing.

1 Introduction

The ever-increasing computational demands posed by big data problems, coupled with the challenge of further improving silicon CMOS technology, has led to an urgent need to develop new computing technologies (Conte et al., 2017). Evolution-in-Materio (EiM) is one such paradigm, a flexible unconventional computing method which can exploit a variety of physical substrates, not relying on increasingly limited silicon technology. Proposed by Miller and Downing (2002), EiM is inspired by the remarkably complex and varied functions that simple nucleotides can perform when configured by evolution into a genome. They proposed a form of evolutionary exploitable device as a kind of configurable analogue processor, which in this work we refer to as an EiM processor. So, while a material system could be optimised for a particular purpose, such as optimising the morphology of a soft robot's gripper (Howard et al., 2021), EiM processors consist of a system within which a material's physical properties are themselves exploited and leveraged towards the desired computational task. EiM processors thus comprise of a material (or medium) whose characteristics are determined by its configuration, and programming (or re-programming) is achieved by an Evolutionary Algorithm (EA) which optimises the material's configuration for a target application. While in-Materio processors could be configured using any external stimuli such as light (Viero et al., 2018) or radio waves (Linden, 2001), research often focuses on materials which can be interacted with via the application and reading of voltages. The electronic functionality of these EiM processors is not designed by the assembling of discrete components, rather an optimal material configuration is sought after and evolved through a supervised learning process. The human element of EiM processor design is the selection of a configurable material, and an appropriate algorithm to efficiently exploit its properties for the target application (Vissol-Gaudin et al., 2016a,b; Lawson and Wolpert, 2006; Dale et al., 2016a). The importance of material selection in this process is underscored by the significant differences in computing function observed when different configurable materials are optimised by the same algorithms to solve the same problems (Dale et al., 2016a; Massey et al., 2015b; Dale et al., 2016b). The fundamental question of which configurable material properties lead to better performing EiM processors, and how they can best be exploited, is largely unanswered and motivates the current paper. This is complicated by the many and varied styles of EiM processor design, leading us to establish a simple EiM classifier structure. Different materials and more complex configurable parameters were slowly introduced, allowing foundational questions about their possible benefits to be answered. We develop a simulation of an EiM processor which allows us to directly relate the choice of nanomaterial used, via the current-voltage characteristic, and algorithm configuration parameters to the classification performance. We demonstrate that problem complexity influences the most successful nanomaterial choice, and furthermore, that design of the algorithm can lead to better exploitation of intrinsic properties. Together these findings display how EiM processors can be designed to give better performance for classification problems.

Evolution pursues its goal of increased fitness for a task by exploiting any aspect of the system upon which it is operating. As such, Miller and Downing (2002) suggested that good evolutionary material candidates are those with complex physics underpinning their properties. This has encouraged the use of configurable nanomaterials (hereafter referred to as nanomaterials) with rich electronic and morphological complexity, such as liquid crystals (Harding and Miller, 2004, 2007), metallic nanoparticles (Bose et al., 2015; Greff et al., 2017) and single-walled carbon nanotube (SWCNT) compos-

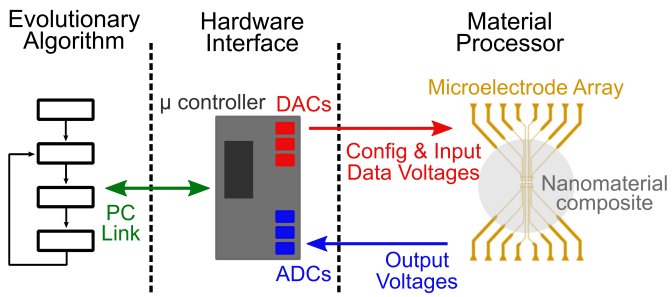

Figure 1: High level diagram of a generic physical EiM processor showing how an EA can interact with some custom hardware which can apply and read voltage signals from a material processor, generally with the use of precision Digital-to-Analogue Converters (DACs) and Analogue-to-Digital Converters (ADCs).

ites (Massey et al., 2015b, 2016), all of which can be configured through the application of a voltage. EiM processors can be fabricated by depositing the nanomaterial onto a microelectrode array which allows input data and configuration voltages to be applied, and output voltages to be measured, as shown schematically in Figure 1. EiM processors therefore utilise a nanoscale electronic network whilst only requiring basic photolithographic processes. An EA is used to configure the electronic transfer function of the nanomaterial network to suit a target computational problem. In this paper we focus on classifying non-temporal datasets, requiring the material processor's output voltages to settle before being read. Therefore, nanomaterials without memory (traps, defects, etc.) will achieve larger bandwidths for data processing. In contrast, if such a system was to process temporal data, then it is likely that nanomaterials with memory might be desirable. The nanomaterial should produce useful output voltages which can be exploited for the target application; although, advice on the selection of conduction characteristics is lacking. As previously mentioned, different nanomaterials can be selected for a wide range of different properties, potentially allowing for fast, power efficient unconventional computing. While some nanomaterials can alter their internal connections (e.g., memristive materials (Strukov et al., 2008), and nanomaterials suspended in solution (Vissol-Gaudin et al., 2017)), these EiM processors lie outside the scope of this article. Instead, we focus on 'fixed' material network EiM processors, enabling their foundational operating principles to be established. This then might inform future research on these more complex systems based on more 'dynamic' materials.

The EA is an iterative process that seeks to improve a population of candidate solutions using biologically inspired methods such as mutation, recombination, and natural selection. Here, each member of the population represents a different set of configuration parameters, which in the case of basic nanomaterial EiM processors are evolvable configuration voltages. First, a training phase is used to evolve an initially random population into a more favourable set of configuration parameters. This involves sending a series of input data instances from a training dataset to the configured nanomaterial in the form of voltages, and determining a corresponding error between the actual and desired output voltages. Following training, a test phase is then used to examine the performance of the best configuration (i.e., population member) using

previously unseen data.

Although EAs are a powerful and flexible method to optimise a system for a task, there are limits to what EiM processors can achieve. It has been demonstrated that optimising different nanomaterials with the same EA for the same function has both varying degrees of success and training time (Dale et al., 2016a; Massey et al., 2015b; Dale et al., 2016b; Clegg et al., 2014). Nanomaterial 'processors' are analogue and often have underlying physical properties that are difficult to model. This leads them to be treated as black boxes, making investigation into which nanomaterial properties are beneficial difficult. Even processors fabricated from nominally the same nanomaterial and optimised by the same EA for the same computational problem vary in quality of solution due to the inherent randomness of nanomaterial morphology (Clegg et al., 2014) and EA convergence. Collating statistical data to further investigate this issue is challenging due to the fabrication and training processes that are required for each EiM processor (Greff et al., 2017). Whilst the impact of nanomaterial formulation on computing performance has been examined for some composites (Dale et al., 2016a; Massey et al., 2015b; Dale et al., 2016b), the general question of what nanomaterial properties lead to better exploited performance remains an open question. This, in turn, precludes the possibility of rational selection or design of nanomaterials to optimise future EiM processors.

In this paper, our contributions to resolving these issues are as follows. We have developed a model of an EiM processor which allows quantitative analysis of how nanomaterial selection on the one hand, and EA design on the other, impact the quality of solutions obtained. The framework developed enables significantly faster investigation into EiM systems then would otherwise be possible with physical construction and experimentation. The physical model of the nanomaterial network uses equivalent circuits to allow examination of differing conduction mechanisms and randomness in conductivity, whilst the allied EA is modified to examine the impact of evolving configuration voltages, connections to the nanomaterial network, input weightings and output weightings. To the best of our knowledge, this is the first time such a study has been carried out. It is shown that the complexity of the classification problem posed influences the best choice of nanomaterial, with more complex problems being favoured by more complex (non-linear) conductivity. Further, it is shown that evolving different aspects of the material configuration have differing physical impacts upon how the nanomaterial behaves and is exploited, in turn providing distinct (if limited) methods to tune EiM performance to a target application. Hence, the presented model, and findings discussed, provide actionable guidance as to how EiM materials and algorithms can be co-designed in the future.

2 Problem Formulation

2.1 Physical Model

To examine the impact of nanomaterial selection and conductivity randomness upon EiM performance, a verified electrical model was required to act as a proxy for the physical nanomaterial. In this paper, the physical model was based on circuit networks which could be solved within a SPICE (Simulation Program with Integrated Circuit Emphasis) simulator. All code used to generate the data in the paper is available on GitHub at `https://github.com/benedictjones/EiM_MaterialProxy`. The material models developed in this section are either physically realisable circuits or are based on measurements of real nanomaterials.

Many EiM processors utilise nanomaterials that provide complex morphology

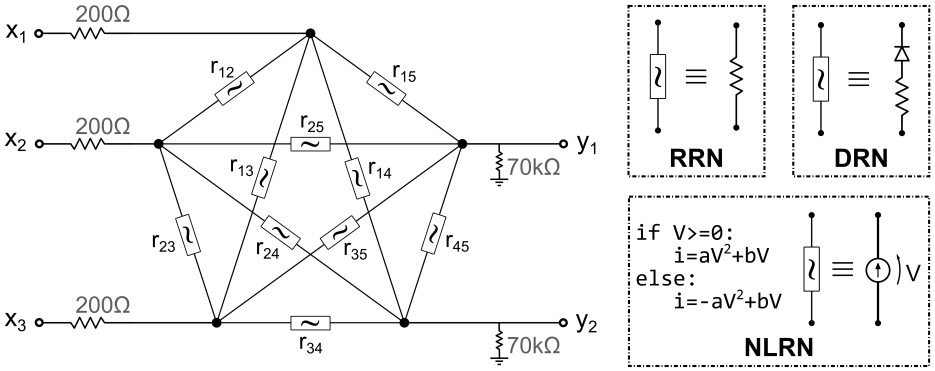

Figure 2: **Electrical model of an example EiM processor.** Example of a 5 node fully connected network used to model EiM devices, with three input nodes (x_1, x_2 and x_3) and two output nodes (y_1 and y_2). Input nodes can be allocated as data driven voltages or configuration voltage stimuli. Between every pair of nodes is a component modelling the material properties: a resistor for the RRN, a resistor in series with a diode (of random direction) for the DRN, or a current source of non-linear characteristic for the NLRN.

with random interconnections between electrodes. Thus, within the model, networks are generated in which all nodes are interconnected via an equivalent circuit which have IV (current-voltage) characteristics selected from a distribution similar to that of a real nanomaterial. Each node, n represents an electrode that is either an input node driven at a voltage (x_n) or a measured output node (y_n), the voltage of which is calculated using an operating point analysis. The input nodes can be used to input data or as evolvable configuration voltage stimuli. An individual processor is thus characterised by the equivalent circuits between nodes, and the number of physical connections required for the inputs and outputs. Figure 2 shows an example of a fully connected network that represents a physical EiM processor with three input nodes and two output nodes. The equivalent circuits between nodes are selected to replicate different functional forms of conductivity and randomness therein. Thus, this model describes the IV characteristics of an ensemble of nanoparticles between electrodes of the EiM processor, rather than individual elements (Lawson and Wolpert, 2006) or junctions (Greff et al., 2017). The motivation for this approach is that individual nanoparticles are usually at least an order of magnitude smaller than the electrode array upon which they are deposited (Massey et al., 2015b; Bose et al., 2015; Massey et al., 2015a) and so it is generally only possible to experimentally characterise a network of nanoparticles, rather than the nanoparticles themselves. In this paper, three material models are considered: (1) a Resistive Random Network (RRN), in which a randomly selected resistor is between every node pair; (2) a Non-Linear Random Network (NLRN) in which a current source of non-linear characteristic is between every node pair; and (3) a Diode Random Network (DRN), in which a diode of random orientation is in series with a randomly selected resistor between every node pair, as shown in Figure 2. Note that as our focus is on the foundational issues of randomness and conductivity, the impact of negative differential resistance offered by memristive materials (Strukov et al., 2008), and physically reconfigurable materials such as nanomaterials suspended in solution

(Vissol-Gaudin et al., 2017), are outside the scope of this article.

The non-linearity of the conduction within the materials increases from the RRN, to NLRN, to DRN models. These models were chosen to represent a range of conduction mechanisms that could be realised with nanomaterials. RRN networks have been reported in the literature (Dale et al., 2016a; Lykkebø et al., 2015). The NLRNs are modelled after the behaviour of on SWCNT/PBMA composites. Assuming the IV characteristic was symmetrical, Experiments (Supplementary Section I) found it could be fitted well using current source equations defined as,

$$I = \begin{cases} aV^2 + bV, & \text{if } V \geq 0 \\ -aV^2 + bV, & \text{if } V < 0 \end{cases} \tag{1}$$

where a & b are material properties, I is the current and V is the voltage between nodes. Testing of the real SWCNT/PBMA material was carried out (Supplementary Section I) to determine that $a \in$ [33n, 170n] and $b \in$ [280n, 960n]. These limits were used to generate uniformly distributed random a & b values for the simulated NL materials. The DRN is a natural extension to the modelled NLRN, representing a highly non-linear nanomaterials with a non-symmetrical IV characteristic. Further details regarding circuit properties and the limits used when uniformly randomly generating new material networks, are given in Supplementary Table II. Finally, we note that physical manifestations of the RRN and DRN were constructed using discrete circuit components and tested using a custom test bench to verify the model's behaviour (Supplementary Section IV).

When a new material is generated the SPICE netlist is saved, and so can be reloaded and re-used for different simulated experiments. These saved netlists could be used to re-create the RRNs and DRNs physically. However, the NLRNs are theoretical, representing the expected behaviour of the SWCNT composites described above.

EiM processors have generally used smaller microelectrode arrays to contact the material: often with sixty four (Miller et al., 2014), sixteen (Vissol-Gaudin et al., 2018; Massey et al., 2016) or fewer electrodes (Kotsialos et al., 2014; Chen et al., 2020; Bose et al., 2015). Smaller networks using only eight electrodes have shown promising results as physical realisations of high-capacity neurons (Ruiz-Euler et al., 2020). The benefits of larger or smaller networks (i.e., the number of electrodes) is of interest. However, this paper focuses on the foundational issues associated with EiM and only considers networks containing ten nodes (i.e., electrodes) or fewer.

2.2 Basic Algorithm Implementation

Different types of EAs have been used for EiM such as Evolutionary Strategies (Dale et al., 2017), Genetic Algorithms (Bose et al., 2015; Chen et al., 2020), Differential Evolution (Vissol-Gaudin et al., 2017; Massey et al., 2016) or Particle Swarm Optimisation Vissol-Gaudin et al. (2016a,b). Here, the physical model is combined with a Differential Evolution (DE) (Storn and Price, 1997; Sloss and Gustafson, 2019) EA. DE is a derivative-free, stochastic, population-base, heuristic direct search method which only requires a few robust control variables (Pedersen, 2010). The DE algorithm uses the greedy criterion and 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 thus change gradually over time due to the random mutation of characteristics and crossover with other population members. Each member of the population is defined by a

d-dimensional vector of decision variables **X** (sometimes known as the genome) that contains the d number of configuration parameters which the DE algorithm optimises. The basic set of decision variables considered are configuration voltages as shown in Equation (2),

$$\mathbf{X} = [V_{c1}\ V_{c2}\ ...\ V_{cP}]^T, \tag{2}$$

where T is the transpose, V_{cp} is the voltage at a configuration node p, and the total number of configuration nodes is P. Extensions to this set of decision variables are discussed later in the paper.

In this work, we considered binary classification problems of varying difficulty. In each problem, an input instance to be classified k contains a series of attributes $a_r(k)$. These input attributes are converted to input data voltages $V_r^{in}(k)$ which are applied to one of the input electrodes as follows,

$$V_r^{in}(k) = a_r(k), \tag{3}$$

where r is a data driven input electrode corresponding to an input attribute, and a 1:1 conversion between the attribute unit and Volts is assumed. Thus, for each instance of the dataset, input voltages are applied, and an operating point analysis is used to determine output voltages.

The material outputs require some interpretation in order to classify a processed input data instance. Here, the output voltages are collected using an output layer which produces an overall network response Y, defined as the sum of the voltages $V_q^{out}(k)$ at all the output nodes q,

$$Y(k) = \sum_{q=1}^{Q} V_q^{out}(k), \tag{4}$$

where Q is the total number of output nodes. The data instance is then designated a class label using a simple threshold,

$$class(k) = \begin{cases} 2, & \text{if } Y(k) \geq 0 \\ 1, & \text{if } Y(k) < 0 \end{cases}. \tag{5}$$

The structure of the proposed EiM processor is shown in Figure 3. Output signals of physical EiM systems have been interpreted in many ways (e.g., a single output-threshold comparison (Bose et al., 2015), output-output comparisons (Massey et al., 2015b; Vissol-Gaudin et al., 2017), with or without evolvable thresholds and penalty terms, etc.). This lack of consistency makes it difficult to compare techniques and isolate which changes lead to more (or less) successful EiM systems. The interpretation scheme used in this paper, defined by (4) and (5), was chosen to be conceptually simple, with the aim of clarifying the role of material properties upon performance. The choice of interpretation scheme can impact the performance of an EiM processor, and this is discussed in further detail in §3.2. We note, however, that the framework developed and used here could be modified to examine other arbitrary network topologies or interpretation schemes.

During each generation, every member of the population is evaluated using some training data, and an associated fitness is calculated using the EAs objective function Φ. We define the objective function as the mean classification error of the processed dataset,

$$\Phi = \frac{1}{K} \sum_{k=1}^{K} e(k), \tag{6}$$

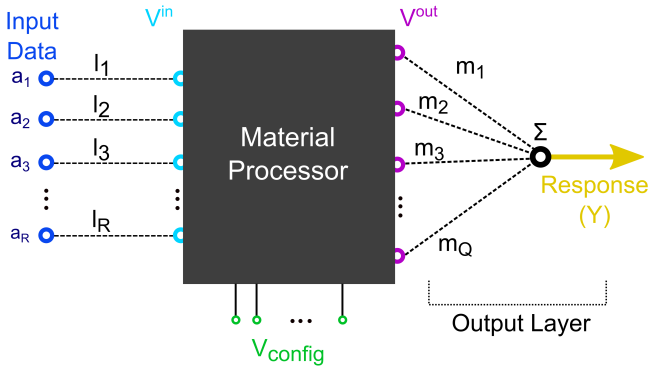

Figure 3: **Illustration of the proposed EiM processor structure.** Input data is applied to the material as voltages. Configuration voltages are applied to the material as evolvable stimuli. The output voltages are summed to generate an overall response (Y) which is used to determine the class. If enabled input weights (l_r) and output weights (m_q) are applied, where R is the total number of data driven input electrodes and Q is the total number of output nodes. A shuffle gene can re-arrange the applied location of the inputs (both input data and configuration nodes).

where k is an instance within the dataset which has a total length of K. Each data input instance produces an error value $e(k)$ of 0 or 1 for correct or incorrect classification respectively. The objective of the EA is thus to minimise the objective function.

A DE/best/1/bin algorithm is used with a mutation factor of $F = 0.8$, crossover rate of $CR = 0.8$ and population of $NP = 20$ (further details given in Supplementary Table II). Selecting the optimum F and CR can prove challenging (Das and Suganthan, 2011), we have selected values which Vissol-Gaudin et al. (2017) found empirically to work well. To perform classification on the dataset D, the data is split into two subsets: training data D^{train} and test data D^{test}. The training subset is used to evaluate and update the population during the evolutionary optimisation. During this generational loop, the best population member p_{best} is tracked and updated. Once the evolutionary period has elapsed, the best member p_{best} (i.e., the final solution) is evaluated using the test subset. Pseudocode describing the EiM optimisation process is presented in Algorithm 1, where $f(pop, Dataset)$ is a fitness function (e.g., Equation (6)) evaluating member(s) of a particular population on a data subset, and $BestFitness(pop, Dataset)$ returns the population member with the best fitness on the selected data subset. The trial and initial populations are generated by producing NP d-dimension vectors whose contents are uniformly distributed $\in [0, 1]$. These vectors are then scaled appropriately depending on the configuration parameters in use and their limits (found in Supplementary Table II).

2.3 Datasets

To assess the performance of the various materials and algorithms four datasets were used. Three of these datasets were generated. The first is a simple two-dimensional dataset (2DDS) problem with two attributes, a_1 and a_2, and two classes, 1 and 2, as seen in Figure 4(a). The second dataset is again two dimensional with two classes, but with

Algorithm 1: Pseudocode for DE based EiM.

Initialise a random population p;
Evaluate initial population fitnesses $f(p, D^{train})$;
$p_{best} = BestFitness(p, D^{train})$;
$i = 0$;
while $i < MaxNumberIterations$ **do**
    Generate trial population t;
    Evaluate trial population $f(t, D^{train})$;
    Update population p with respect to t;
    $p_{best} = BestFitness(p, D^{train})$;
    $i = i + 1$;
end
Evaluate p_{best} using the test data $f(p_{best}, D^{test})$;

data generated concentrically, as seen in Figure 4(b). Both datasets contain 500 samples in each class, which were then split $80\% - 20\%$ to create balanced training and test datasets respectively. The 2DDS presents a linearly separable problem which should be easily solved by basic classifiers. The con2DDS is a more challenging dataset, requiring a classifier to exploit non-linearities within its network to achieve better classification results. The third dataset is identical to the 2DDS, except the classes are flipped (i.e., reversed), referred to as the flipped2DDS. This inverted dataset is used to investigate whether the EiM processors have classification boundaries that are favourable to certain data orientations.

The final dataset considered is the UCI Mammographic Mass Data Set (Dheeru Dua, 2017), refereed to here as the MMDS. The MMDS was split $80\% - 20\%$ to create balanced training and test subset respectively, and is used to compare the proposed EiM systems with other common classification techniques.

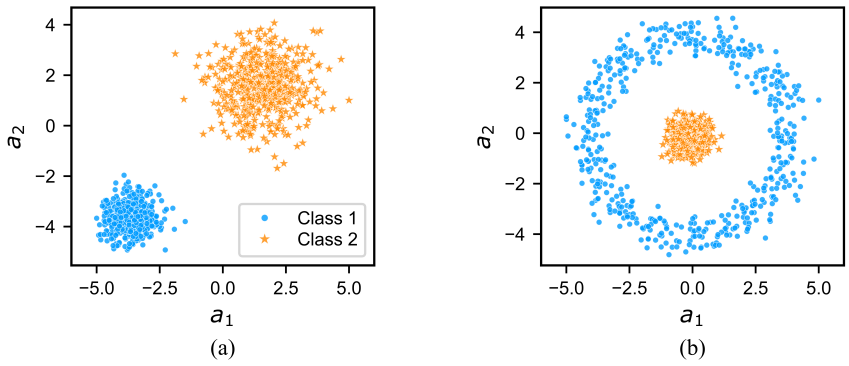

Figure 4: **The randomly generated datasets used to evaluate the performance of the different materials and algorithm configuration parameters.** The (a) 2DDS and (b) con2DDS datasets.

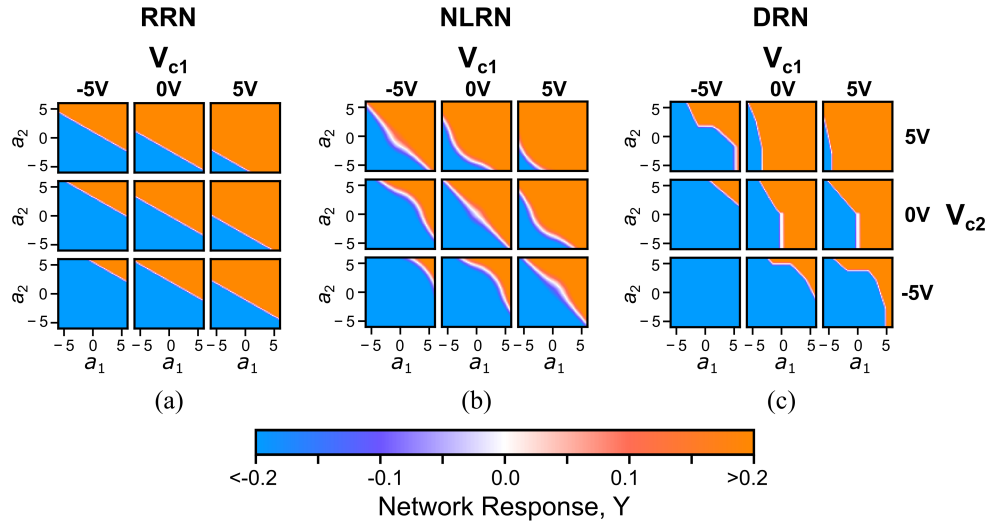

Figure 5: **Examples of the role that the material and configuration voltages have on untrained EiM processor responses.** Surface plot of the network response, Y as a function of input attributes a_1 and a_2 for an untrained (a) RRN, (b) NLRN, and (c) DRN processor, and the effect of varying their two configuration voltages V_{c1} and V_{c2}.

3 Interaction between algorithm and materials

3.1 Role of Material Properties and Configuration Voltages

We first examine the role of the material properties on processor performance. The networks considered had two input nodes, two outputs nodes and two configuration nodes. Figure 5(a), 5(b) and 5(c) show surface plots of network response Y, relating to the sum of voltages from output nodes, for a range of data inputs, a_1 and a_2, for the RRN, NLRN and DRN models respectively. Here, the configuration voltages V_{c1} and V_{c2} are varied so that their role is highlighted. The network response Y is used as the criterion by which classification is made (Equation (5)) and represents the output space which the inputs have been mapped onto. We denote the line $Y = 0$, which is the threshold between the classes, as the decision boundary. For the unconfigured ($V_{c1} = V_{c2} = 0V$) RRN processor, the decision boundary is a straight line which passes through the origin. Unlike the RRN, the NLRN processor can achieve a smooth curved decision boundary due to its non-linear conduction characteristics. By contrast, for DRN processors, the decision boundary has distinct bends or 'kinks' brought about by rapid changes in conductivity when a diode is turned on. Hence, we observe that the intrinsic properties of the nanomaterial, namely randomness and degree of (non-) linearity in conduction, have significant effects upon the shape of decision boundaries in untrained processors. Even nominally the same material can have significantly different boundary shapes due to the differences in inter-node characteristics.

A material without any evolvable stimuli can be used to map some inputs to a new output space. However, EiM processors have been shown to provide superior classification performance when trained configuration voltages are applied to the network (Vissol-Gaudin et al., 2016b). Thus, in Figure 5 we also display the impact of changing the two configuration voltages for the considered networks. EiM processors can

Table 1: The mean test fitness, standard deviation, and best test fitness after 30 iterations, for the different processor types using the basic algorithm.

Dataset	Processor Type	mean(Φ^{test})	std(Φ^{test})	Best Φ^{test}
2DDS	RRN	0.000	0.000	0.000
	NLRN	0.000	0.000	0.000
	DRN	0.005	0.001	0.000
flipped2DDS	RRN	0.514	0.004	0.500
	NLRN	0.500	0.000	0.500
	DRN	0.500	0.000	0.500

often operate at low voltages (Vissol-Gaudin et al., 2018; Chen et al., 2020; Dale et al., 2016a) so here we consider the effect of applying -5V to 5V. These voltage stimuli behave similarly to input biases, however not only 'shifting' the output, but also varying how the inter electrode IV characteristics are being exploited, thereby altering how the inputs are mapped to the output space. Within the linear RRN network, application of configuration voltages only leads to a translation of the decision boundary, as a consequence of superposition, and that in no cases is a rotation of the decision boundary observed. For the NLRN and DRN network, again we observe a translation of the decision boundary when configuration voltages are applied, together with some changes in curvature, although not to an extent where the relative orientation of the two classes is changed. These observations are of significant practical importance, as it shows that the orientation of the decision boundary for both Ohmic and non-Ohmic solid nanomaterials is a function of the random IV characteristics determined during fabrication. An inability to rotate the decision boundary via the configuration voltages to satisfy a particular classification problem may be expected to limit the fitness which an algorithm could obtain.

To investigate this finding further, the performance of the different networks as EiM processors was assessed for the 2DDS problem. Fifteen different individual RRN, NLRN and DRN networks were considered and optimised as a classifier using the DE algorithm as described in §2.2. Each network had two inputs, two outputs and three configuration nodes. To ensure that randomness within the algorithm optimisation process did not obscure other trends, the evolution process was repeated on each network 5 times. The test fitness (Φ^{test}) results for each of the processor types after 30 iterations are shown in Table 1. All the RRN and NLRN processors achieved a zero fitness (i.e., 100% accuracy). However, some of the DRN processors failed to achieve a zero fitness. The more linear networks have simpler decision boundaries which are more favourable to the simple 2DDS. Examples of the network response for a trained RRN, NLRN and DRN achieving 100% accuracy are shown in Figure 6.

Using the same material processors and repeating the experiment on the flipped2DDS (Table 1) shows all the processor types becoming 'stuck', in this case being unable to improve their fitnesses below 0.5. This is because the processors now have unfavourable initial slopes of the decision boundary with respect to the dataset. As the basic algorithm is unable to rotate the decision boundary or make large changes in its curvature, the best that the EA can achieve is to misclassify 50% of the data. These results provide an explanation as to why some experimental data (Clegg et al., 2014) show that nanomaterial composites fail to achieve acceptable classification accuracies following training.

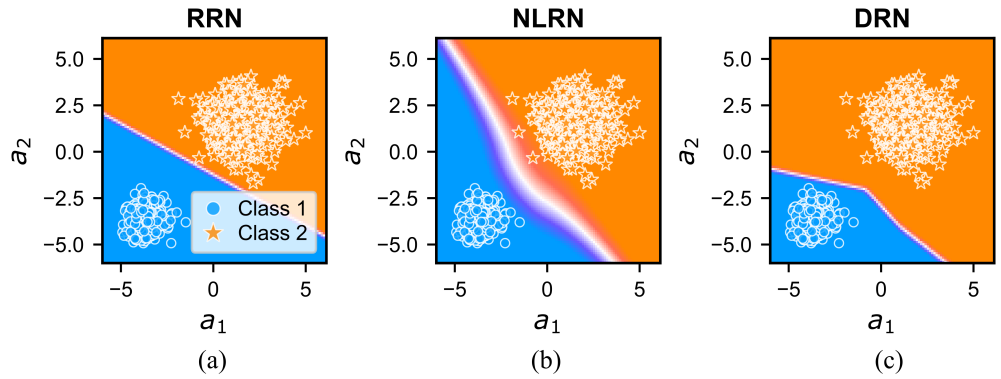

Figure 6: **Example processor responses following training using the basic algorithm for 30 iterations.** Surface plot of the network response, Y as a function of input attributes a_1 and a_2 for a (a) RRN, (b) NLRN, and (c) DRN processor, using the basic EiM algorithm on the 2DDS. The same colour scale as Figure 5 is used.

3.2 Impact of Electrode Reconfiguration and Weighting

While on the one hand, the results in §3.1 suggest it should be possible to screen candidate processors prior to possibly lengthy and unsuccessful training, we argue that they also suggest a method to exploit nanomaterial composites more effectively. The success or not of the DE algorithm to exploit the nanomaterial is limited by the inter-node IV characteristics, which for solid nanomaterials, is determined at creation. However, it is possible to re-arrange the input and configuration nodes and thus realise different arrangements of inter-node IV characteristics without changing the material. Input electrodes can be used for data driven or configuration voltages, the function of an input electrode and the ability to 'shuffle' or relocated different inputs can be easily controlled while programming. On the other hand, a multiplexer could be used to physically reconnect an electrode as either an input (e.g., to a DAC) or output (e.g., to a ADC).

Indeed, some EiM approaches have this flexibility by allowing the EA to modify node location (Bose et al., 2015; Clegg et al., 2014). We reason that changes to the inter-node IV characteristics are likely to result in uncovering exploitable configurations of the nanomaterial composite.

This supposition is explored in Figure 7 which shows the output response Y, with no configuration voltages applied, but with randomly reordered input data and configuration connections to the same network. Here, only a few random arrangements are shown and we note that these examples are not meant to be exhaustive. The number of possible input permutations scales quickly with the number of inputs (e.g., for materials with four input nodes there are twenty four possible input arrangements etc.), so here we focus on the limitations of this technique. It is observed that different arrangements have a different shape of decision boundary that could be exploited to fit classification data. Furthermore, some slight variations in the slope of the decision boundary are observed, albeit in a discontinuous fashion. However, due to the interpretation scheme defined by (4) and (5), it is always the case that (e.g.) two positive inputs can only provide a positive (and therefore class 2) output. This in turn suggests that introducing evolving multiplication factors or 'weights' for the applied input

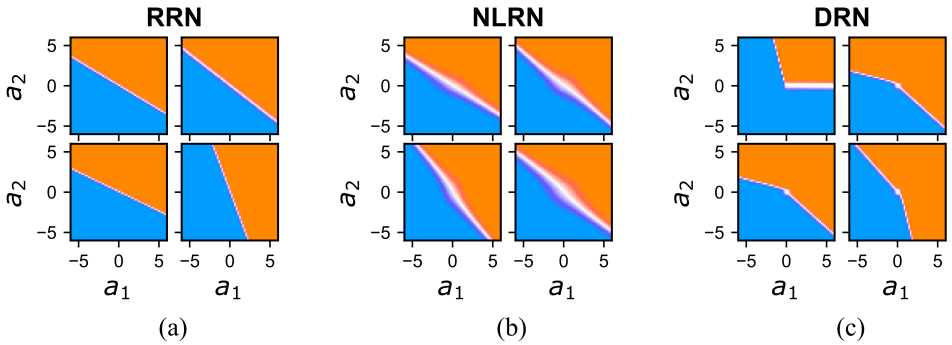

Figure 7: **Examples of the impact that electrode reconfiguration can have on untrained EiM processor responses.** Surface plot of the network response Y as a function of input attributes a_1 and a_2 for four randomly selected shuffle genes (i.e., permutations of the input voltage order) of an unconfigured (a) RRN, (b) NLRN, and (c) DRN processor. The same colour scale as Figure 5 is used.

voltages or read output voltages may be beneficial, as this may allow unconstrained rotation of the decision boundary. Other styles of interpretation scheme, previously mention in §2.2, might prove similarly limited without their own additional evolvable parameters.

We investigated the impact of varying output weights on the material response Y for an unconfigured RRN, NLRN and DRN processor, as seen in Figure 8. Inverting the polarity of one output weight allowed the introduction of more complex behaviour into the material's response, such as the gradient for the RRN processor, and complex boundary shapes of the NLRN and DRN processors. As expected due to symmetry, inverting the polarity of both output weights caused the classes to swap. Using output weights of opposing polarities generally made the decision boundary less 'sharp' since subtracting values leads to Y values closer to zero, which in turn would make a physical EiM processor more susceptible to noise. Figure 8 also shows the effect of input weights to the same RRN, NLRN and DRN networks. It is notable that changing input and output weights yield different responses, since input weights cause a change in input voltages which must propagate though the material network. This is particularly important for the DRN, since due to its asymmetric non-linear IV characteristics, changing the input weights impacts when diodes are turned on, so inverting the polarity of both input weights will not simply cause the classes to swap. Thus, we can draw a distinction between the effect of using input and output weights: input weights allow for variability on how the nanomaterials currently selected inter-node IV characteristic is exploited, whereas output weights allow for variation in how the output signals are interpreted and combined.

4 Advanced EiM Algorithm

To exploit the distinct impacts of re-arranged electrodes, and changing input or output weights upon the decision boundary, we modified the vector of decision variables as follows:

$$\mathbf{X} = [V_{c1}\ V_{c2}\ ...\ V_{cP}\ G_{sh}\ m_1\ m_2 ...\ m_Q\ l_1\ l_2\ ...\ l_R]^T. \tag{7}$$

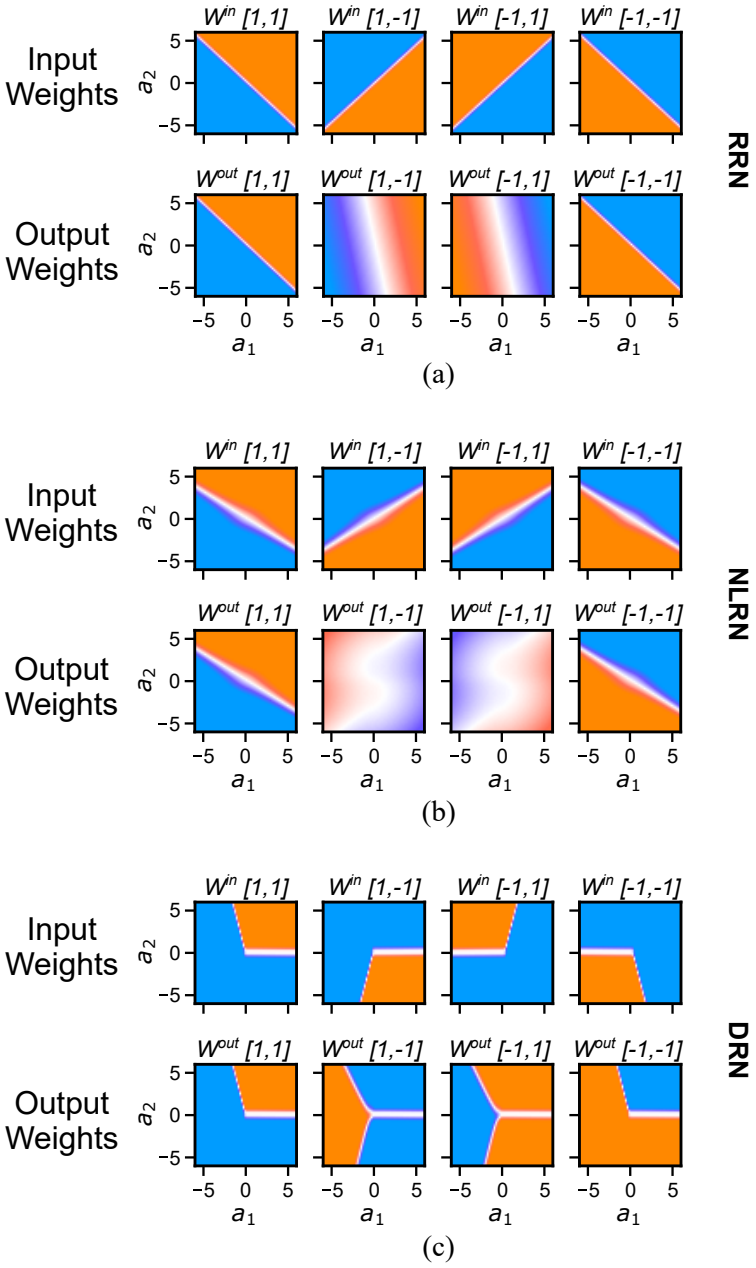

Figure 8: **Examples of the effect of varying the input and output weightings on untrained EiM processor responses.** Surface plot of the network response Y as a function of inputs a_1 and a_2 for an unconfigured (a) RRN, (b) NLRN, and (c) DRN processor with various output or input weightings applied. The same colour scale as Figure 5 is used.

Here, G_{sh} is termed the 'shuffle' gene, which allows for reassignment of input electrodes to access different inter-node arrangements as shown in Figure 7. G_{sh} is an integer which defines a particular permutation of the ordering for where input and configuration voltages are applied. The input weights, $l_r \in [-1, 1]$ scale the input voltages V_r^{in} applied at the data driven input electrodes r due to an input attribute a_r, such that:

$$V_r^{in}(k) = l_r \times a_r(k), \tag{8}$$

where the total number of data driven input electrodes is R. Finally, the output weights, $m_q \in [-2, 2]$ for each output electrode, q allow for changes in the network response, Y, as follows:

$$Y(k) = \sum_{q=1}^{Q} m_q V_q^{out}(k). \tag{9}$$

This well-defined framework allows a reliable investigate into the roles and benefits that additional algorithm configuration parameters may have on the of different materials.

5 Simulation Results

5.1 Electrode Allocation and Material Properties

Understanding how many configuration and output electrodes are required for good classifier performance will enable more reliable EiM processor construction. To investigate this, fifteen RRN, NLRN and DRN processors were generated, each with a fixed size of ten nodes. The classification performance of these systems was considered using the con2DDS, where the advanced EiM algorithm ran for 50 iterations, and was repeated five times on each network to mitigate randomness in convergence. Results displayed in Supplementary Figure 1 show that after four repetitions of the DE algorithm the standard deviation of a materials performance is both low and settles. Similarly, the effect of averaging over a number of the same type but different randomly generated materials was considered (Supplementary Figure 2), where low standard deviations are achieved and the performance started to settle if more then five to ten materials where considered. Therefore, in these experiments, fifteen of each processor type were chosen to ensure that the average capability of a particular material type could be analysed without any one high or low performance processor skewing the results, while the simulation times could remain within reasonable lengths. Two nodes were designated as data inputs; the remaining eight nodes were designated as either configuration nodes, output nodes, or if unused were left floating. The effect of varying the number of configuration and output nodes was then investigated, with the results presented in Figure 9.

First we observe that the difference between nanomaterial conductivities (i.e., whether they are RRN, NLRN or DRN) is more important in determining performance than electrode allocation. The number of configuration nodes seems to play a less important role when evolving the RRN, NLRN and DRN materials, and suggests that only a few voltage stimuli are required. However, we speculate that materials with more complex properties, or large sparsely connected networks, will benefit more from the voltage configuration stimuli.

The DRN and NLRN EiM processors perform better than the RRN processors, this is because the materials non-linear IV characteristics are being exploited to better curve the decision boundary and fit to the concentric data. The RRN processors can only

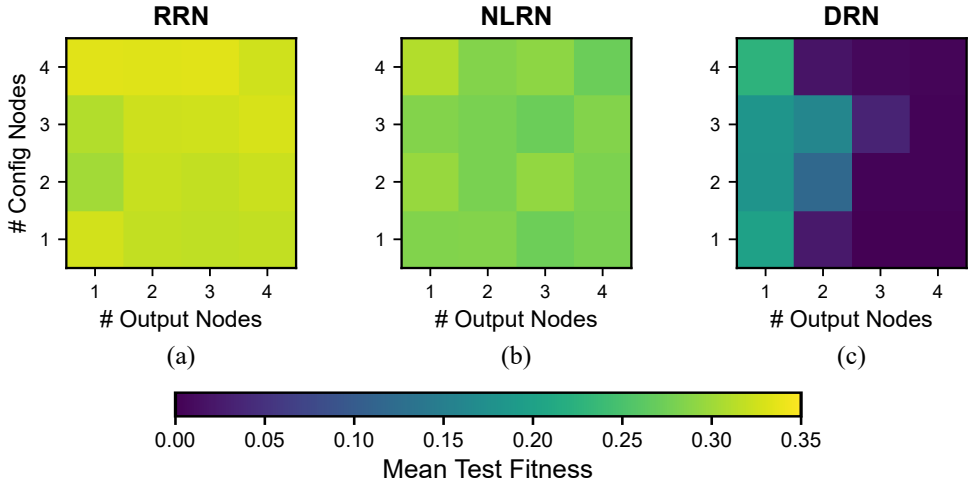

Figure 9: **Effect of varying which nodes are allocated as either configuration or output electrodes on the final test fitness after training.** Surface plot of the mean test fitness (from 15 material processors, each with 5 DE repetitions) after 50 iterations using the advanced EiM algorithm on the (a) RRN, (b) NLRN, and (c) DRN networks to classify the con2DDS. The materials all had a fixed size of ten nodes, but the number of nodes allocated as configuration or outputs was varied. Unallocated nodes were left floating.

produce linear decision boundaries which can only be placed tangentially to the data, similar to that shown in Figure 10(a), thus severely limiting the final fitness. The NLRN processors, however, only achieve marginally better results than the RRN processors. The materials non-linearities help fit the data, as shown in Figure 10(b), but neither additional configuration nodes nor additional output nodes lead to significant improve-

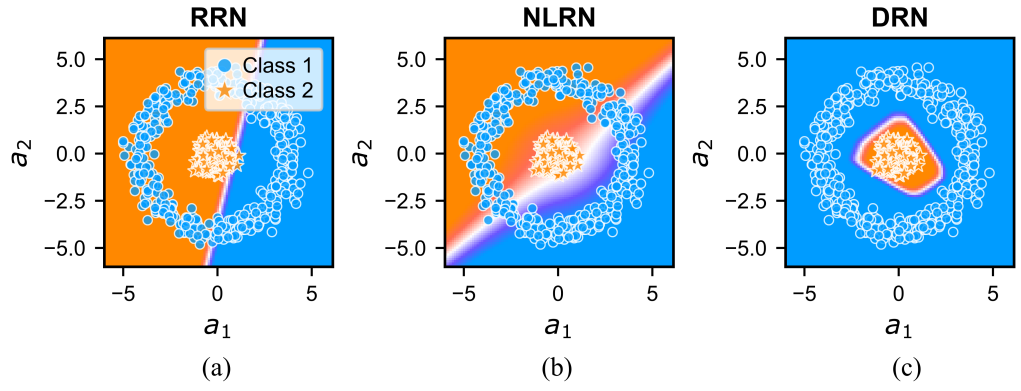

Figure 10: **Example processor responses following training using the advanced EiM algorithm for 50 iterations.** Surface plot of the network response, Y as a function of input attributes a_1 and a_2 for a (a) RRN, (b) NLRN and (c) DRN processor, using the advanced EiM algorithm with all the additional configuration parameters, on the con2DDS. The same colour scale as Figure 5 is used.

ment. We speculate that the voltage outputs from the NLRN are too similar, meaning it is harder to combine the outputs to produce an enclosed decision boundary. The DRN networks contain more abrupt non-linearities within its IV characteristics, allowing its output voltages to be more easily combined to generate enclosed decision boundaries and achieve better fitnesses, similar to the response shown in Figure 10(c). However, Figure 9 shows that it is necessary to combine more than one output to achieve an enclosed area, and for consistently good results more than two output nodes are required.

We note that the materials discussed here all have monotonically increasing IV characteristics. In this case, a material with only a single output cannot successfully classify the con2DDS (or an XOR problem); instead, at least 2 outputs are required. However, materials containing a Negative Differential Region (NDR) can be exploited to solve the XOR problem with only one output (Bose et al., 2015; Chen et al., 2020).

5.2 Modifying the Decision Vector

In the following we investigate the impacts of including the 'shuffle' gene, input weight and output weight configuration parameters into the decision vector, both individually and in combination, for the three networks, when solving the classification problems. Considering the results discussed in §5.1, and using the same fifteen RRN, NLRN and DRN material processors, three nodes were allocated as outputs, two nodes as inputs, and the remaining five as configuration nodes.

The con2DDS was used to train the systems for 50 iterations, but with the differing combinations of decision parameters. The convergence of the mean best member (p_{best}) training fitness at each iteration from the 75 recorded fitnesses (the 15 different materials considered, each of which had the DE algorithm repeated 5 times) is shown in Fig 11(a), 11(c) and 11(e) for the RRN, NLRN and DRN processor types respectively. Box plots of the corresponding processor final test fitnesses are also shown in Fig 11(b), 11(d) and 11(f). For clarity, results are only displayed for the different individual additional configuration parameters (i.e., 'shuffle', input weights or output weights) in conjunction with configuration voltages. Additionally, the performance when all configuration parameters are enabled is displayed. Results for all remaining combinations of configuration parameters are shown in Supplementary Figures 3 and 4.

Using only configuration voltages leads to poor fitnesses in all the processor types considered. The inability of this evolutionary scheme to explore different IV characteristics or rotate the decision boundary leads to similar behaviour observed in §3.1 when using the flipped2DDS led to 'stuck at' faults. By contrast, using shuffle or input weights in the decision vector allows for a wider exploitation of the material characteristics, improving performance; although in some cases this leads to some mild overfitting. The ability of shuffle to discover useful IV characteristics and the use of input weights to exploit them is therefore fundamentally limited by the EiM processors material properties. This is easily distinguished by the fact that only schemes which used output weights could achieve test fitnesses below 0.160 (Supplementary Table IV). Therefore, for processors to achieve better fitnesses, it becomes essential that the interpretation scheme can evolve to successfully combine the material's outputs. This enables the introduction of more complex boundary features, such as a fully enclosed area which is needed for the concentric dataset.

While the scheme using all the additional configuration parameters achieved good eventual performance after the 50 iterations, it converged significantly slower than the output weight only scheme used for the NLRN and DRN processors. This suggests that not all the configuration parameters are needed, and their introduction can be either

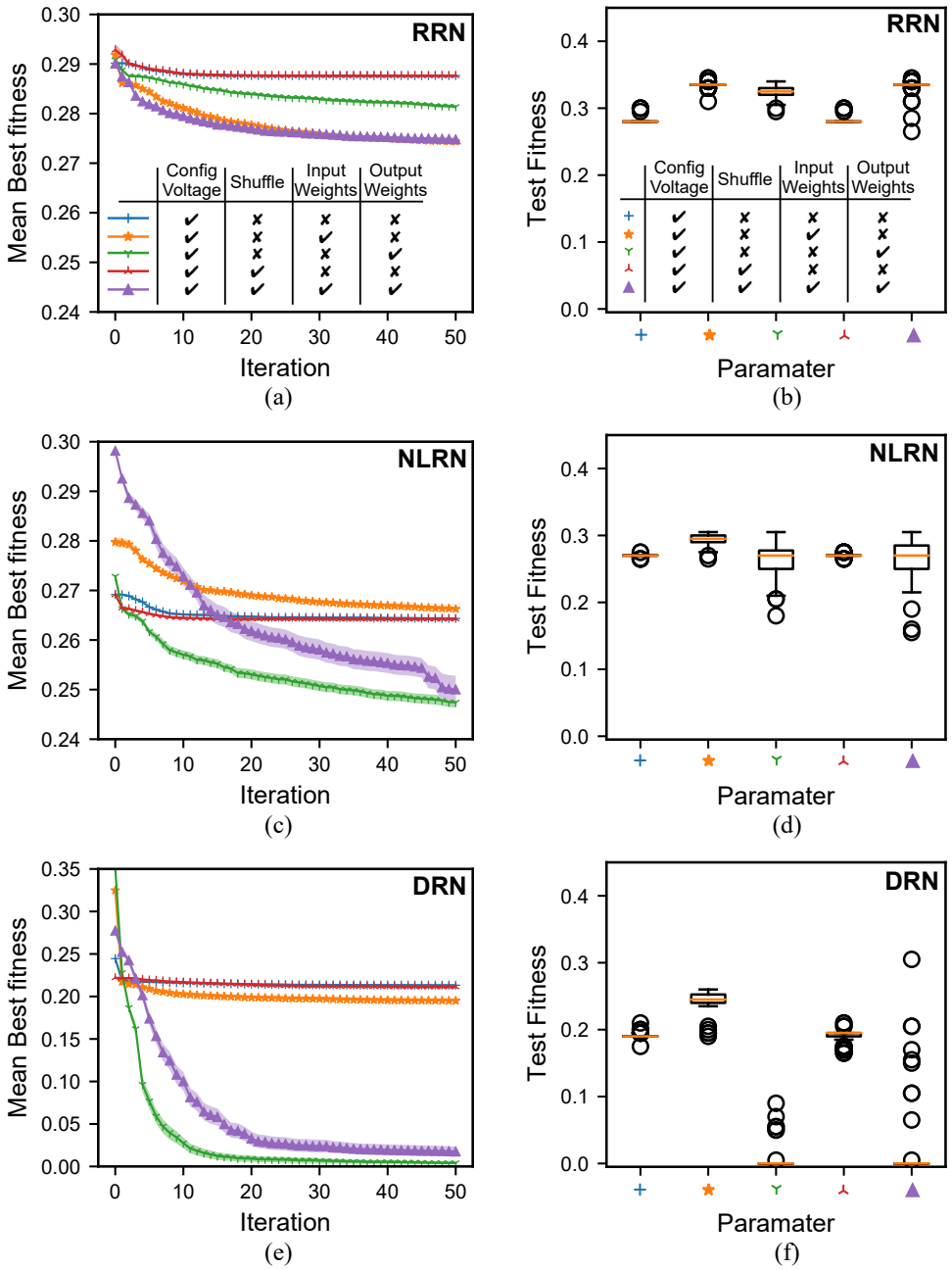

Figure 11: **Evolution of training fitness and final test fitness for EiM processors using different materials and configuration parameters, classifying the con2DDS.** The mean best training fitness & standard error (from 15 material processors, each with 5 DE repetitions) over 50 iterations, with different DE algorithm configuration parameters enabled, for (a) RRN, (c) NLRN, and (e) DRN processors using the con2DDS. These are paired with boxplots of the final test fitness results for the (b) RRN, (d) NLRN, and (f) DRN processors.

counter productive and/or inflates the search space. However, we would propose that a marginal decrease in convergence speed is worth the significant gains in flexibility which a solution can take.

This experiment was repeated for the 2DDS and flipped2DDS as seen in Supplementary Figure 5 and 6 respectively. Most configuration schemes solved the 2DDS and achieved a zero fitness within less than 10 iterations. However, all the processor types fail to optimise the flipped2DDS unless either input or output weights are used. This again highlights the limitations of the configuration voltages and shuffle gene which may only exploit the complexities of the given EiM processors' material properties. As discussed in §3.2, to enable the decision boundary to 'flip' or rotate, input or output weightings must be used.

These results support Miller and Downing's conjecture that materials with rich, complex physics should be good candidate EiM processors (Miller and Downing, 2002). We have shown that the form of non-linear conduction within the nanomaterial processor will play a role in eventual performance. The relative performance of one EiM processor with respect to another depends sensitively upon the EA and configuration parameters used, and furthermore, that the inherent capability of a nanomaterial for EiM may be 'hidden' or obstructed if an inappropriate algorithm is used. Design of algorithm and selection of nanomaterial are thus necessary to achieve good performance for a target application.

5.3 Performance Benchmarking

For the sake of comparison, the performance of the basic (configuration voltage only) and advanced (all additional configuration parameters) EiM algorithm was compared to some common classification techniques. The results are presented in Table 2, and include classification results on the con2DDS and also the MMDS. The con2DDS results are taken from the work discussed in §5.2. The MMDS results were generated using the same 15 ten node materials (with 5 repetitions) as in §5.1 but using 100 iterations.

The advanced EiM algorithm can lead to a performance increase in excess of 40%

Table 2: Classification performance of the discussed basic and advanced EiM algorithms, other common algorithms, and other work.

Dataset	Processor Type	mean(Φ^{test})	std(Φ^{test})	Best Φ^{test}
con2DDS	Basic EiM (NLRN)	0.270	0.002	0.265
	Basic EiM (DRN)	0.191	0.004	0.175
	Logistic Regression	-	-	0.490
	Random Forest	-	-	**0.000**
	Advanced EiM (NLRN)	0.266	0.029	0.155
	Advanced EiM (DRN)	0.000	0.058	**0.000**
MMDS	Basic EiM (NLRN)	0.485	0.000	0.485
	Basic EiM (DRN)	0.485	0.000	0.485
	Logistic Regression	-	-	0.228
	Random Forest	-	-	0.269
	Advanced EiM (NLRN)	0.236	0.020	0.204
	Advanced EiM (DRN)	0.245	0.037	0.210
	EiM (SWCNT/LC) (Vissol-Gaudin et al., 2017)	0.2051	-	**0.1885**

compared to the 'basic' EiM algorithm. However, if the nanomaterial processors' properties are unsuitable to the computational task, a better exploiting algorithm cannot achieve significant performance gains. Notably, when properly harnessed, these simple physically realisable networks can outperform both Logistic Regression and the Random Forest (100 trees) algorithm.

Vissol-Gaudin et al. (2017) used a SWCNT/LC EiM processor to classify the MMDS. These carbon nanotubes suspended in a liquid crystal mixture could move and form new connections using evolved voltage stimuli. Classification decisions were made using two output electrodes and an evolvable classification threshold. The SWCNT/LC EiM processor achieves a lower error then the simulated materials discussed in this paper. We hypothesise that the increase in performance is due to the materials ability to internally re-configure inter-node connections, altering the materials IV characteristics. The material processors considered in this paper all have fixed IV characteristics. While this might present a less flexible in-materio processor, it allows for a single processor to be trained on many tasks, and switch between tasks by simply re-calling the trained configuration parameters.

6 Discussion

These findings can be interpreted as confirmation of the suitability of nanomaterials for EiM processors due to the variation in conduction mechanisms and conductivity (Conte et al., 2017). Depending upon the material and device geometry chosen, the conduction pathways in a nanomaterial may have conduction that is Ohmic ($I \propto V$) or Poole-Frenkel ($I \propto V \exp(-(E_d - \beta\sqrt{V})/kT)$) in nature, or be limited by space-charge ($I \propto V^2$) or a Shottky/pn junction ($I \propto \exp(V)$), to name a few examples. Nanomaterials may also display a range of conduction mechanisms within the same composite as well as variation in their conductivity. However, the inter-electrode IV characteristic of many nanomaterial-based EiM processors reported to date are due to a percolation network of individual nanoparticles (Miller and Downing, 2002; Vissol-Gaudin et al., 2016a,b), and in turn, the apparent diversity in conduction mechanisms that a type of nanoparticle may offer will be reduced due to averaging along the conduction path. We suggest that higher performance EiM processors may be realised as the inter-electrode distance approaches that of individual nanoparticles. One of the possible benefits of nano-structured devices, is the possibility of negative differential resistance (NDR) which can be used to classify the XOR problem while only using a single material output (Bose et al., 2015; Chen et al., 2020). The material networks considered in this work were each assumed to have similar inter-node characteristics like those found in the literature. However, while maybe more challenging to produce, materials with heterogeneous properties might provide a more exploitable and better performing material network, and warrants further research.

Further to the modes of conduction observed, the nanomaterials used in EiM processors fall into one of two categories: static materials with fixed IV characteristics and dynamic materials which have variable IV characteristics. In this paper we only considered models of static nanomaterials, examples of which include carbon nanotubes suspended in polymer matrices (Dale et al., 2016a; Massey et al., 2015b; Clegg et al., 2014; Dale et al., 2017; Kotsialos et al., 2014) and random resistor networks (Dale et al., 2016a,b). However, nanomaterials can also be dynamically changed by applying a voltage, as in the case of liquid crystals (Harding and Miller, 2004, 2007), CNTs suspended in a liquid crystal matrix (Vissol-Gaudin et al., 2016b; Massey et al., 2016; Volpati et al., 2015), and memristors (Sillin et al., 2013). When processing non-temporal data, mate-

rials with no memory and a fast settling time are desirable to allow a large bandwidth (Chen et al., 2020). Although comparisons between static and dynamic EiM processors are sparse (Vissol-Gaudin, 2020), it appears that dynamic materials have better performance in some circumstances. We propose that reconfiguration of the electrical network in dynamic materials allows different inter-node IV characteristics to be continuously explored by the evolutionary algorithm, in much the same way as the shuffle algorithm parameter operates here, and that this may be at least part of the reason why some dynamic materials have better EiM performance. However, many dynamic materials cannot be reset to a previous configuration (e.g. as is the case for CNTs suspended in liquid crystal (Massey et al., 2016)), whereas the trained configuration parameters of a static material can be simply recalled. Whether the benefits of configuration recall in static materials or a dynamic, irreversible search space are more beneficial to EiM performance appears to be worthy of further study.

7 Conclusion

While Evolution in-Materio (EiM) is a promising unconventional computing paradigm, analysis of EiM systems remains limited due to slow fabrication and training processes. In this paper, EiM classifiers are produced by combining Differential Evolution with an electrical model which is used as a proxy for real EiM material processors. This allowed for fast and efficient *in-simulo* experimentation of EiM processors. Using this framework, foundational issues with EiM processors were investigated. Different materials and evolvable configurable parameters were investigated in succession, allowing their effect to be isolated and analysed.

The findings presented explain why some nanomaterial based EiM processors fail to achieve good performance, and how the exploiting algorithm can be modified to mitigate these effects. Significantly, it is observed that the complexity of the 'ideally selected' material scales with the complexity of the problem, with acceptable solutions to simple classification problems being found more quickly with simple random resistor networks, and the solution of more complex problems being favoured by more complex random non-linear networks. Furthermore, differing modes of reconfiguring the material, weighting the input data, and interpreting the material outputs are shown to have distinct advantages and limitations. Significantly, we have demonstrated how these methods can be used in concert to better exploit the material processor and leverage better performance for EiM classifiers.

Looking forward, these results show that to create high performance EiM processors, rational design of algorithm and selection of nanomaterial for a target application is necessary. The framework presented allows one to quantify these trade-offs and make informed decisions in the design of future EiM processors.

Acknowledgements

BB thanks the Department of Engineering, Durham University for research internship funding.

References

Bose, S. K., Lawrence, C. P., Liu, Z., Makarenko, K. S., van Damme, R. M. J., Broersma, H. J., and van der Wiel, W. G. (2015). Evolution of a designless nanoparticle network into reconfigurable Boolean logic. *Nature Nanotechnology*, 10(12):1048–1052.

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., and van der Wiel, W. G. (2020). Classification with a disordered dopant-atom network in silicon. *Nature*, 577(7790):341–345.

Clegg, K. D., Miller, J. F., Massey, M. K., and Petty, M. C. (2014). Practical issues for configuring carbon nanotube composite materials for computation. In *2014 IEEE International Conference on Evolvable Systems*, pages 61–68.

Conte, T. M., DeBenedictis, E. P., Gargini, P. A., and Track, E. (2017). Rebooting Computing: The Road Ahead. *Computer*, 50(1):20–29.

Dale, M., Miller, J. F., Stepney, S., and Trefzer, M. A. (2016a). Evolving Carbon Nanotube Reservoir Computers. In Amos, M. and CONDON, A., editors, *Unconventional Computation and Natural Computation*, Lecture Notes in Computer Science, pages 49–61, Cham. Springer International Publishing.

Dale, M., Stepney, S., Miller, J., and Trefzer, M. (2016b). Reservoir computing in materio: An evaluation of configuration through evolution. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*.

Dale, M., Stepney, S., Miller, J. F., and Trefzer, M. (2017). Reservoir computing in materio: A computational framework for in materio computing. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2178–2185.

Das, S. and Suganthan, P. N. (2011). Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31.

Dheeru Dua, E. K. T. (2017). UCI Machine Learning Repository. http://archive.ics.uci.edu/ml.

Greff, K., van Damme, R. M. J., Koutnik, J., Broersma, H. J., Mikhal, J. O., Lawrence, C. P., van der Wiel, W. G., and Schmidhuber, J. (2017). Using neural networks to predict the functionality of reconfigurable nano-material networks. In *International Journal on Advances in Intelligent Systems*, volume 9, pages 339–351. IARIA.

Harding, S. and Miller, J. (2004). Evolution in materio: A tone discriminator in liquid crystal. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1800–1807 Vol.2.

Harding, S. and Miller, J. F. (2007). Evolution In Materio: Evolving Logic Gates in Liquid Crystal. *International Journal of Unconventional Computing*, 3:243–257.

Howard, D., O'Connor, J., Letchford, J., Brett, J., Joseph, T., Lin, S., Furby, D., and Delaney, G. W. (2021). Getting a Grip: In Materio Evolution of Membrane Morphology for Soft Robotic Jamming Grippers. *arXiv:2111.01952 [cs]*.

Kotsialos, A., Massey, M. K., Qaiser, F., Zeze, D. A., Pearson, C., and Petty, M. C. (2014). Logic gate and circuit training on randomly dispersed carbon nanotubes. *International journal of unconventional computing.*, 10(5-6):473–497.

Lawson, J. W. and Wolpert, D. H. (2006). Adaptive Programming of Unconventional Nano-Architectures. *Journal of Computational and Theoretical Nanoscience*, 3(2):272–279.

Linden, D. (2001). A system for evolving antennas in-situ. In *Proceedings Third NASA/DoD Workshop on Evolvable Hardware. EH-2001*, pages 249–255.

Lykkebø, O. R., Nichele, S., and Tufte, G. (2015). An Investigation of Square Waves for Evolution in Carbon Nanotubes Material. *Artificial Life Conference Proceedings*, 27:503–510.

Massey, M., Volpati, D., Qaiser, F., Kotsialos, A., Pearson, C., Zeze, D., and Petty, M. (2015a). Alignment of liquid crystal/carbon nanotube dispersions for application in unconventional computing. *AIP Conference Proceedings*, 1648(1):280009.

Massey, M. K., Kotsialos, A., Qaiser, F., Zeze, D. A., Pearson, C., Volpati, D., Bowen, L., and Petty, M. C. (2015b). Computing with carbon nanotubes: Optimization of threshold logic gates using disordered nanotube/polymer composites. *Journal of Applied Physics*, 117(13):134903.

Massey, M. K., Kotsialos, A., Volpati, D., Vissol-Gaudin, E., Pearson, C., Bowen, L., Obara, B., Zeze, D. A., Groves, C., and Petty, M. C. (2016). Evolution of Electronic Circuits using Carbon Nanotube Composites. *Scientific Reports*, 6(1):32197.

Miller, J. and Downing, K. (2002). Evolution in materio: Looking beyond the silicon box. In *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, pages 167–176, Alexandria, VA, USA. IEEE Comput. Soc.

Miller, J. F., Harding, S. L., and Tufte, G. (2014). Evolution-in-materio: Evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67.

Pedersen, M. E. H. (2010). Good Parameters for Differential Evolution.

Ruiz-Euler, H.-C., Alegre-Ibarra, U., van de Ven, B., Broersma, H., Bobbert, P. A., and van der Wiel, W. G. (2020). Dopant Network Processing Units: Towards Efficient Neural-network Emulators with High-capacity Nanoelectronic Nodes. *arXiv:2007.12371 [cs, stat]*.

Sillin, H. O., Aguilera, R., Shieh, H.-H., Avizienis, A. V., Aono, M., Stieg, A. Z., and Gimzewski, J. K. (2013). A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology*, 24(38):384004.

Sloss, A. N. and Gustafson, S. (2019). 2019 Evolutionary Algorithms Review. *arXiv:1906.08870 [cs]*.

Storn, R. and Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.

Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *Nature*, 453(7191):80–83.

Viero, Y., Guérin, D., Vladyka, A., Alibart, F., Lenfant, S., Calame, M., and Vuillaume, D. (2018). Light-Stimulatable Molecules/Nanoparticles Networks for Switchable Logical Functions and Reservoir Computing. *Advanced Functional Materials*, 28(39):1801506.

Vissol-Gaudin, E. (2020). *Evolutionary Computation Based on Nanocomposite Training: Application to Data Classification*. Doctoral, Durham University, Durham University.

Vissol-Gaudin, E., Kotsialos, A., Groves, C., Pearson, C., Zeze, D., and Petty, M. (2017). Computing Based on Material Training: Application to Binary Classification Problems. In *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–8, Washington, DC. IEEE.

Vissol-Gaudin, E., Kotsialos, A., Groves, C., Pearson, C., Zeze, D., Petty, M., and Al Moubayed, N. (2018). Confidence Measures for Carbon-Nanotube / Liquid Crystals Classifiers. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Vissol-Gaudin, E., Kotsialos, A., Massey, M. K., Zeze, D. A., Pearson, C., Groves, C., and Petty, M. C. (2016a). Data Classification Using Carbon-Nanotubes and Evolutionary Algorithms. In Handl, J., Hart, E., Lewis, P. R., López-Ibáñez, M., Ochoa, G., and Paechter, B., editors, *Parallel Problem Solving from Nature – PPSN XIV*, Lecture Notes in Computer Science, pages 644–654, Cham. Springer International Publishing.

Vissol-Gaudin, E., Kotsialos, A., Massey, M. K., Zeze, D. A., Pearson, C., Groves, C., and Petty, M. C. (2016b). Training a Carbon-Nanotube/Liquid Crystal Data Classifier Using Evolutionary Algorithms. In Amos, M. and CONDON, A., editors, *Unconventional Computation and Natural Computation*, Lecture Notes in Computer Science, pages 130–141, Cham. Springer International Publishing.

Volpati, D., Massey, M. K., Johnson, D. W., Kotsialos, A., Qaiser, F., Pearson, C., Coleman, K. S., Tiburzi, G., Zeze, D. A., and Petty, M. C. (2015). Exploring the alignment of carbon nanotubes dispersed in a liquid crystal matrix using coplanar electrodes. *Journal of Applied Physics*, 117(12):125303.