Reservoir Computing *in materio*: A Computational Framework for *in materio* Computing

Matthew Dale* & Susan Stepney
Department of Computer Science
University of York, UK
Email: md596@york.ac.uk & susan.stepney@york.ac.uk

Julian F. Miller & Martin Trefzer
Department of Electronics
University of York, UK
Email: julian.miller@york.ac.uk & martin.trefzer@york.ac.uk

***Abstract*—The Reservoir Computing (RC) framework is said to have the potential to transfer onto any input-driven dynamical system, provided two properties are present: (i) a fading memory, and (ii) input separability. A typical reservoir consists of a fixed network of recurrently connected processing units; however recent hardware implementations have shown reservoirs are not ultimately bound by this architecture. Previously, we have demonstrated how the RC framework can be applied to randomly-formed carbon nanotube composites to solve computational tasks. Here, we apply the RC framework to an evolvable substrate and compare performance to an already established *in materio* training technique, referred to as evolution *in materio*. The results show that by adding the programmable reservoir layer, reservoir computing *in materio* can significantly outperform the original evolution *in materio* implementation. This suggests the RC framework offers improved performance, even across non-temporal tasks, when combined with the evolution *in materio* technique.**

I. Introduction

Research in unconventional computing has shown that many diverse dynamical systems can be exploited to perform computational tasks [1]. Exploiting computation at the substrate-level offers potential advantages over classical computing architectures, such as exploiting physical and material constraints that could offer solutions "for free", or at least computationally cheaper [25]. However, extracting computation from these systems and physically *programming* them is a challenging task. This is sometimes exacerbated by a desire for minimal abstraction, i.e. exploiting physical phenomena directly, and wanting a level of programmability that enables the system to perform a variety of tasks. Hybrid digital-analogue computers have the potential to fulfil these needs, where the digital system is trained to extract computation performed implicitly from an analogue substrate. Kirchhoff-Lukasiewicz Machines (KLM) [19] which are based on Rubel's computational model of analogue computation (the extended analogue computer [23]) are a prime example.

For analogue computers both the program and architecture may be indistinguishable or inseparable, i.e. to program the machine requires a change in the machine architecture. However, by placing additional layers within the machine's architecture the amount of change required is reducible, for example, by adding a standardised reconfigurable "middle" layer. Mills implemented a middle layer for KLMs using Lukasiewicz logic arrays that perform piecewise linear functions on measured values from the material. To control the logic arrays a vector of bits referred to as the "overlay" [19] was used. The semantic "program", the overlay, is used to describe the reconfigurable layer that exploits the implicit material function. The computational functions being utilised are therefore a result of the material's configuration, typically achieved through the selection of inputs, outputs and control functions/signals. Mills describes this as a paradigm of *analogy* [20] where the computing device is not explicitly told to perform an operation and provide a readable output, but rather trained to exploit an implicit function that results from the material's configuration.

A similar computing paradigm – without the middle "read-out" layer – has been applied to computational composites consisting of randomly dispersed carbon nanotubes. These semi-conducting substrates have been shown to exhibit interesting computational properties across a variety of tasks using evolved configurations [3]. Applying evolved control signals, the unknown electrical properties of these composites are perturbed and trained to produce physical solutions to computational tasks. Training these composites is done through the evolutionary selection of input-output mappings and evolvable control stimuli, using a technique referred to as evolution *in materio* [18]. The training method attempts to discover those material configurations that display interesting computational features related to a given task, amongst a vast space of configurations that is constrained only by physical laws.

Evolution *in materio* (EiM) is a growing field with new hardware developments, experimental materials and directions appearing from different research groups, e.g. [2], [17]. However, what is classed here as the "vanilla" technique has its limitations. Modelling and understanding what computational properties evolution is exploiting, both in the full-embodied system, and within the material itself is again challenging. The vanilla technique is argued here to be restricted to a repertoire of tasks that are small-scale and non-temporal – unless further modified.

In [7], [8], we have shown that by applying the Reservoir Computing (RC) framework to such materials, we can exploit interesting temporal properties that were previously unused. We call this Reservoir Computing *in materio*, RCiM. The addition of the RC framework has enabled us to attempt temporal problems requiring both dynamic behaviour and memory. Here, we investigate how well the new reservoir/material framework compares in performance to the vanilla EiM technique, on a non-temporal task. The task is to perform classification on the Fisher Iris data set across a range of carbon nanotube composites, including a control case (a resistor array) and similar density materials used in the vanilla system [21], [22]. The final section presents two simple analysis techniques that could provide further understanding as to what reservoir/material properties are being exploited by evolution.

II. Reservoir Computing

Reservoir Computing emerged as an efficient mechanism for training recurrent neural networks but also evolved into a general theoretical model for many dynamical systems [24]. By applying only a relatively simple training mechanism many physical systems have become exploitable unconventional computers (see [6]).

Reservoirs function as temporal *kernels* [14] and can represent any excitable non-linear medium, given the medium can: (i) create a high-dimensional projection of the input into observable *reservoir states*; and (ii) possess a fading memory of previous inputs and internal states. These prerequisite properties are described by the *separation*, *approximation* [15] and *echo state* properties [11]. With these properties a reservoir can realise any non-linear filter with bounded memory, and with the aid of a trained readout approximate any function.

To interpret the material as a reservoir we define the observed reservoir states $x(n)$ of the material as a combination of the implicit material function and the *discrete* observation of the material:

$$x(n) = \Omega(\mathcal{E}(W_{in}u(t))) \tag{1}$$

where $\Omega(n)$ is the observation of the macroscopic material behaviour and $\mathcal{E}(t)$ the continuous microscopic material function when driven by the input $u(t)$ and the input weight matrix W_{in}.

Training of the reservoir's linear readout is typically done using Ridge Regression [13], manipulating the weights W_{out} to reduce the error (Normalised Root Mean Squared Error *NRMSE*) between the training signal $y_{target}(n)$ and the reservoir output $y(n)$:

$$W_{out} = Y_{target}X^T(XX^T + \beta I)^{-1} \tag{2}$$

where I is the identity matrix and β the Tikhonov regularisation parameter. The trained *in materio* reservoir is defined in eqn.(3), the same as a traditional echo state network.

$$y(n) = W_{out}x(n) \tag{3}$$

III. Experimental Platform

The materials used in this RCiM experiment consist of various concentrations (w.r.t weight) of single-walled carbon nanotubes, randomly dispersed in an insulating polymer: poly-butyl-methacrylate (PBMA) or poly-methyl-methacrylate (PMMA). Each material is deposited onto a gold/chromium microelectrode array of 12 electrodes used to form input-outputs to the material (see Fig.1). The electronic properties of the dispersed nanotubes are approximately one-third metallic nanotubes and two-thirds semiconducting nanotubes. The relative size of the nanotubes (100nm to 1000nm length and diameter between 0.8nm and 1.2nm) is significantly less than the gap between the electrodes (between 100 to $150\,\mu\text{m}$), suggesting that the nanotubes form conductive pathways between the electrodes. The polymer mixed with the nanotubes acts both as a dielectric and as a suspension for the nanotube network. In previous experiments, it has been observed that both a conductive network is formed and a high computational performance is reached around a percolation threshold of 1% carbon nanotubes [7], [8], [16].

To interface the computer with the material, two input/output data acquisition cards are used. Each card is controlled by a MATLAB interface and is set to either input, or read, analogue voltages from the microelectrode array. To allow each card to communicate with any electrode, a cross-point switch is used. The evolutionary process can reconfigure the switch, and route different input-output combinations.

IV. Machine Learning Classification

The Iris data set[1] (also known as *Fisher's* Iris data set) is a well-known multivariate data type classification problem, and has been used to benchmark the vanilla evolution *in materio* technique [4], [21], [22]. The task is to classify three species of the Iris plant given the four attributes of petal and sepal length and width. The Iris data set contains 150 instances, with 50 instances of each class/species.

The data set was evenly divided into training and testing sets of 75 randomised instances, containing 25 instances of each class. Each class in the reservoir framework is represented as a separate reservoir output (Fig.2) with a binary value, and each attribute is represented by a floating-point input voltage.

V. Training Reservoir Computers

Each material is deposited onto a microelectrode array, providing electrical inputs and outputs to the system. Using computer-controlled evolution, the role of each electrode is decided, as well as reservoir input parameters. For example, evolution decides which electrodes are inputs $u(n)$, or outputs $x(n)$, and evolves a *weight* value for each input (see Fig.2). In addition to the weights, another parameter is evolved, called the *leak rate*. The leak rate parameter α is used to match the dynamics of the reservoir to the task input and/or target output. The leak rate parameter is applied post-collection of states, eqn.(4).

$$\tilde{x}(n) = (1-\alpha)x(n-1) + \alpha\Omega(\mathcal{E}(W_{in}u(t))) \tag{4}$$

[1]The Iris data set can be found on the UCI Machine learning repository at: https://archive.ics.uci.edu/ml/

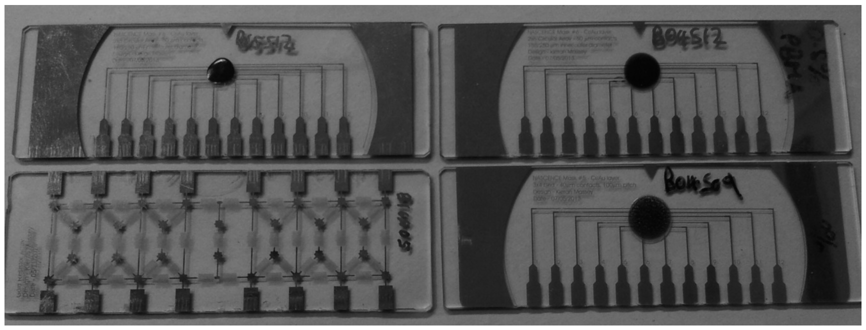


Fig. 1: Substrates under test. Top left, SWCNT/PBMA mixture with a concentration of 1% SWCNT by weight. Top right, SWCNT/PBMA 0.53%. Bottom left, gold resistor array. Bottom right, SWCNT/PMMA 0.1%

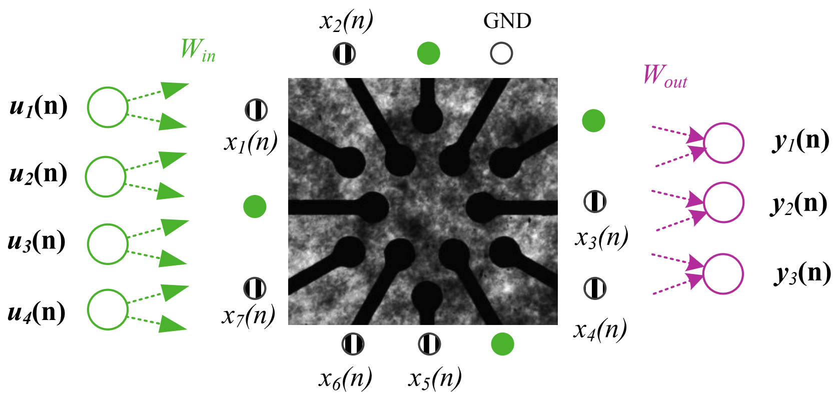


Fig. 2: Input-Output mapping of the task inputs $u(n)$ and observed reservoir states $x(n)$. For this task 4 inputs are required ($u_{1:4}(n)$), each input is multiplied by the input weight matrix W_{in} and applied to an electrode (green). To form the 3 output classes $y_{1:3}(n)$, the reservoir states ($x_{1:7}(n)$) are multiplied by the output matrix W_{out} (the trained readout layer).

This turns eqn.(3) into eqn.(5) – this parameter, however, is assumed to have minimal effect on reservoir performance as data points (*t*) have no temporal relationship.

$$y(n) = W_{out}\tilde{x}(n) \tag{5}$$

In each experiment we apply an elitist evolutionary strategy (ES) of $1+\lambda$ ($\lambda = 4$) for 150 generations, with 20 independent runs. At each generation, a population of five individuals (configurations) are created containing: the "fittest" individual from the previous generation, and the mutation-only offspring (λ) of said individual. When combined with Gaussian mutation operators, the implementation of this strategy acts somewhat like a hill-climber algorithm. This was chosen to reduce the retention effect of degenerative fitness jumps often experienced by rapidly changing the configuration parameters; most notably found when flipping inputs to outputs and vice versa. However, applying Gaussian adaptation to the $(1+\lambda)$ ES could push training towards local optima. Therefore, to train these materials requires some level of comprise, as the material may not be truly "reset", i.e. the material may retain charge from previous inputs and evaluations, or, the material may have permanently changed its phase space – a pertinent issue if the underlying material structure is non-stationary.

Training and testing is split into two phases. Using the *training set*, evolution selects appropriate input-output mappings and input weights, then ridge regression trains the readout layer. In testing, a final evaluation of the material configuration and trained readout performance is carried out on previously unseen *test set* data.

A. *Fitness Evaluation*

To train the reservoir and evolve a solution, the *NRMSE* between the trained output and the target output is used to define fitness. Here the task is to classify binary classes, rather than a time-series output, so a threshold mechanism is used to translate the trained outputs into binary classes. To evaluate the accuracy of the binary classifier, and to conduct a fair comparison with the previous vanilla EiM results, the fitness calculation in [21] is applied:

$$\text{fitness} = \frac{TP \times TN}{(TP + FP)(TN + FN)} \tag{6}$$

where *TP* is the number of true positives, *TN* is true negatives, *FP* is false positives, and *FN* is false negatives.

To select a threshold, a simple optimisation loop was used (post-state collection). The best threshold was then attached and stored to the given configuration and reimplemented at

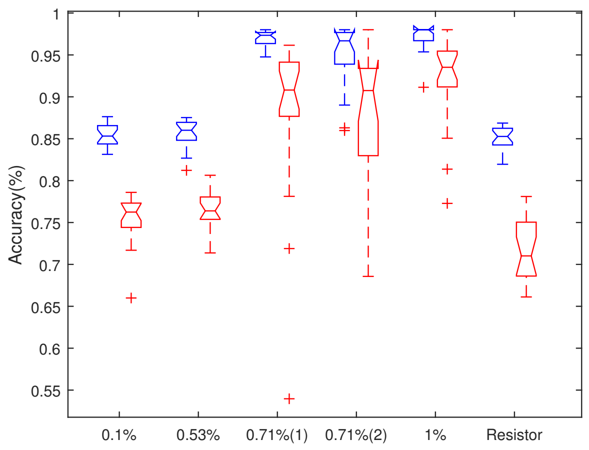


Fig. 3: Evolved accuracies of each material for the *Iris* task.

TABLE I: RCiM compared to EiM. Average (mean) accuracies (in percentages) are shown for training and test data. For RCiM, standard deviation (std) is provided.

System : Material	Training	std	Test	std
EiM [21] : PMMA 0.71%	84.7	—	77.1	—
RCiM : PMMA 0.71% (1)	96.96	0.92	88.15	10.01
RCiM : PMMA 0.71% (2)	94.9	3.92	87.91	7.85
EiM [22] : PBMA 1%	93.33	—	87.87	—
RCiM : PBMA 1%	97.14	1.61	91.90	5.48
CGP [21]	97.7	—	93.6	—

the testing phase. The implementation of this fitness schema is applied here only for comparison purposes; fitness might be better represented using cross-entropy [12].

VI. Experimental Results

Fig.3 shows the evolved accuracies of each material using the RCiM framework. Each material shows training accuracy (blue, left) and test accuracy (red, right) across 20 evolutionary runs. The 0.71%(1) and 0.71%(2) entries refer to two different samples of the same concentration. In this figure, we see the control material (resistor array) has statistically lower accuracies than the lower density materials, suggesting material properties beyond resistivity are being exploited. As the distributions are non-normal, the non-parametric Wilcoxon rank sum test (equivalent to Mann-Whitney U-test) is used to test the null hypothesis H_0 = the medians of the samples are the same. A value $p < 0.05$ indicates statistically significant rejection of H_0 at the 95% confidence level. The non-parametric Vargha-Delaney effect size A [26] is also given (only for $p < 0.05$ in Table II and Table III). $A > 0.71$ symbolises a large effect size and therefore the significance is of substantial importance and not simply an effect of the sample size. The rejection of H_0 with a large effect size in Table II shows that the materials provide a significant contribution to the solution, and that the solution is not dominated by contributions from rest of the system, or from resistivity alone.

TABLE II: Wilcoxon/Mann-Whitney (U) and Vargha-Delaney A effect sizes, between different density materials and the control.

Test materials	p U	effect size A
0.1% / Resistor	0.002	0.78
0.53% / Resistor	0.0001	0.85
1% / 0.71% (1)	0.086	—
1% / 0.71% (2)	0.051	—

TABLE III: Wilcoxon/Mann-Whitney (U) and Vargha-Delaney A effect sizes, between PBMA 1% material and simulated reservoirs.

Test materials	p U	effect size A
1% / 7-node ESN	0.51	—
1% / 7-node (sampled) ESN	0.74	—
1% / 50-node ESN	0.0481	0.684

Fig.3 shows RCiM performance varies with respect to carbon nanotube density, with densities of 0.71% and above producing the highest accuracies. However, Table II shows no statistical significance between the higher concentration materials (0.71% and 1%), demonstrating similar medians and distributions with p-values > 0.05. This suggests once a network of nanotubes is established – effectively determined by nanotube density – an increase in computational performance is minimal. The same hypothesis is discussed in [16].

Table I shows the comparison between EiM, RCiM and an *in silico* evolutionary optimisation technique called Cartesian Genetic Programming (CGP) [21], [22]. The reported EiM accuracies in Table I come from two different hardware configurations: [21] uses Mecobo 3.0, a digital stimulation and measurement board; [22] uses Mecobo 3.5, an analogue stimulation and measurement board. The accuracies quoted are assumed to be mean values, across 10 runs for [22] and 20 runs for [21]. Standard deviation was not provided for the EiM method and is therefore omitted from Table I. In our comparison experiments we use 20 runs and provide standard deviations.

As shown in Table I, our RCiM method outperforms the EiM technique across both Mecobo platforms. The mean accuracies of both Mecobo platforms fall outside the 95% confidence levels of our RCiM experiments, suggesting our increase in performance is significant. The performances of our physical reservoirs also compare favourably to the *in silico* evolutionary programming technique (CGP). The CGP mean accuracy, although higher than the RCiM accuracy, falls within the 95% confidence level of the PBMA 1% material (that is, it is not statistically significantly higher).

For the final comparison, we present evolvable *in silico* reservoirs (*Echo State Networks* (ESNs) [11]). Fig.4 compares the performance of the best material (1% PBMA) to three evolved simulated reservoirs, with training accuracy (blue, left) and test accuracy (red, right) given across 20 evolutionary runs. In this experiment, two reservoir sizes are compared; a network with 7 internal nodes and two networks containing 50

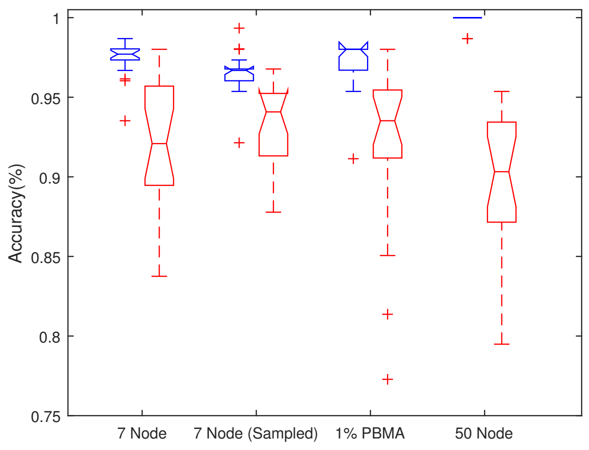


Fig. 4: Experimental results of three simulated reservoirs and the 1% PBMA material.

internal nodes. The 50-node reservoirs are split into different readout types; one where all nodes are accessible to the readout layer and another where the readout can only access 7 randomly chosen nodes. The hypothesis is that when extracting states from our material, we only sample a small subset of the materials state space – in our case, through roughly 7 electrodes. It is therefore imperative to observe how this effects simulated reservoirs.

The similarity in accuracies found in Fig.4 suggests the material forms a trainable reservoir within a similar performance range of the 7-node and the sampled reservoirs. As shown in Table III, there is no statistically significant difference in performance when comparing the material to the sampled and 7-node simulated reservoirs. However, the 50-node network is statistically significant for the Mann-Whitney test (i.e. different medians).

Upon examination of Fig.4, we can see the material may be exploiting the benefits of a smaller readout combined with a larger network for this task, much like the sampled network. This is shown on the 50-node network where overfitting is present, but reduced in the 50-node network where only 7 neuron states are sampled to form the output. This *sampling effect*, in the context of the number of electrodes available, is an interesting and possibly advantageous, or limiting factor, that needs to be further understood.

In summary, three key points are highlighted in the experimental results,: (i) the material is both significant to the computational system and possesses properties beyond resistivity (Table II); (ii) RCiM performance varies with respect to nanotube density (Fig.3), but, variation in performance is less significant at higher densities (Table II); and (iii) our RCiM method, on this task, can compete with two optimised *in silico* methods, CGP and simulated reservoirs (Table I, Fig.4 and Table III), and can *outperform* vanilla EiM (Table I).

A. Discussion

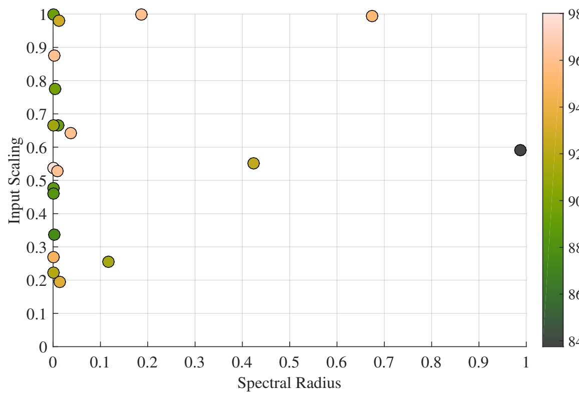
The results of these experiments are intriguing, as they show an increase in performance over EiM when applying the reservoir computing framework and its programmable output layer. This might appear counterintuitive as the direct programming nature of EiM should arguably outperform a system that adds a layer of general-purpose programmability; an increase in programmability typically results in a sacrifice in performance or evolvability [5]. Yet, this does not appear to be the case. However, this exception might be attributed to the new measurement and stimulation technique. For example, an increase in performance might be a result of: (i) stimulating the material with variations of the input causes interesting interactions not present in the vanilla technique alone; (ii) a conductive network might not be present across all electrodes and instead several networks may exist across the array, therefore additional inputs-outputs grant access to each of these networks; (iii) combining the outputs and weighting their importance allows training to exploit the whole material rather than exploiting a single area around a particular electrode (relating closely to (ii)). These hypothesised explanations still require further exploration.

The material's ability to suppress any recurrent dependencies, as seen by the time-independent nature of the task, may suggest certain configurations act more akin to non-temporal kernels or Extreme Learning Machines (ELM) [10], rather than typical temporal reservoirs. To attempt time-independent tasks with simulated reservoirs, reducing the spectral radius and increasing the input scaling parameter would intuitively minimise any dependencies. Others however have proposed techniques to "break" the recurrent dependency on previous states and inputs altogether [9]. The interesting point here is that different configurations – on the same materials – have been shown to exploit temporal features on time-dependent tasks and also to vary considerably across different configurations [7], [8]. This suggests tuning of the echo state property is possible through configuration, rather than the material behaving entirely as a non-temporal reservoir/kernel/ELM. Further analysis of the materials and the type of reservoir dynamics required, or exploited, for this task may indicate why this plausible.

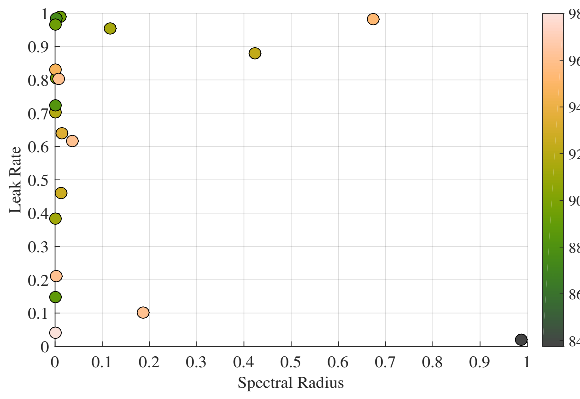
VII. Analysing Reservoir and Material Characteristics

As part of the investigation three evolved simulated reservoirs (*Echo State Networks*) of different sizes were added for comparison (Fig.4). The general parameters and structure for these networks are: a sparse reservoir connectivity of 10%, uniformly distributed weights between $[-1\ 1]$ for input and reservoir weights, tanh neurons, no feedback weights, and a readout trained using ridge regression.

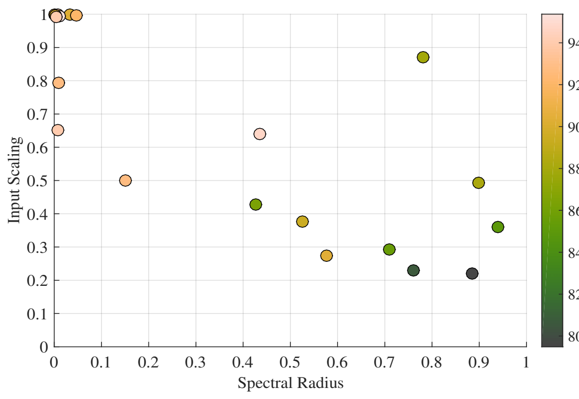
Each reservoir has three key parameters that were selected through evolution; the *spectral radius*, *input scaling* and *leak rate*. Each parameter is adjusted to tune the dynamical behaviour and memory of the reservoir. In echo state networks, the spectral radius determines how fast the influence of the input degrades and the stability of the reservoirs activations. The Input scaling parameter is used to tune the non-linearity of the reservoir and tune the proportional effect the current input



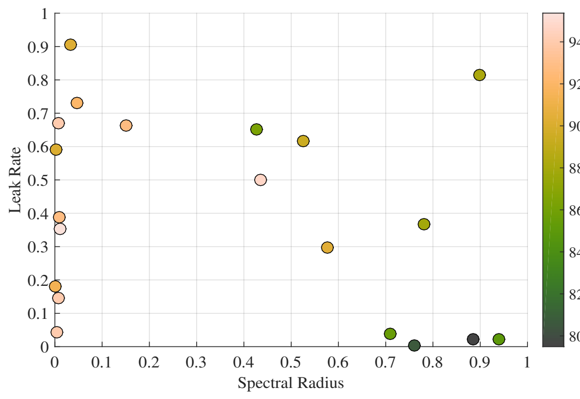
(a) 7 node network - Spectral Radius vs. Input Scaling



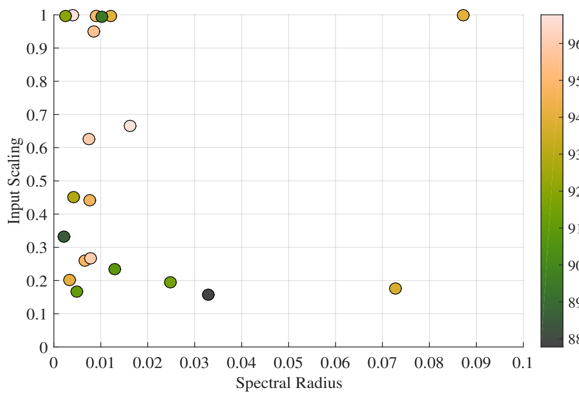
(b) 7 node network - Spectral Radius vs. Leak Rate



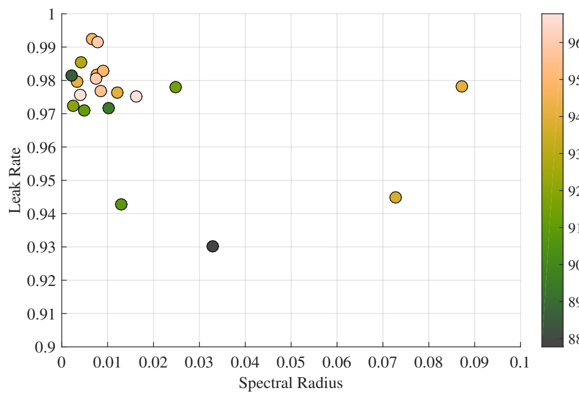
(c) 50 node network - Spectral Radius vs. Input Scaling



(d) 50 node network - Spectral Radius vs. Leak Rate

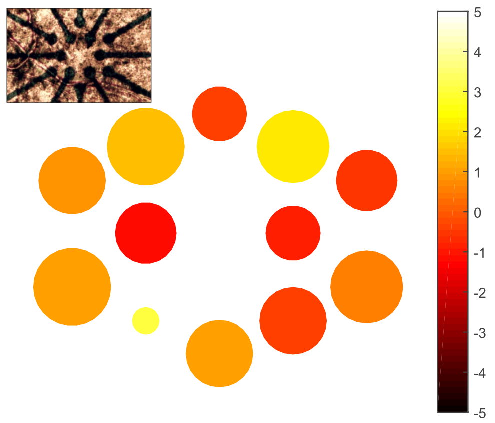


(e) 7 node (sampled) network - Spectral Radius vs. Input Scaling

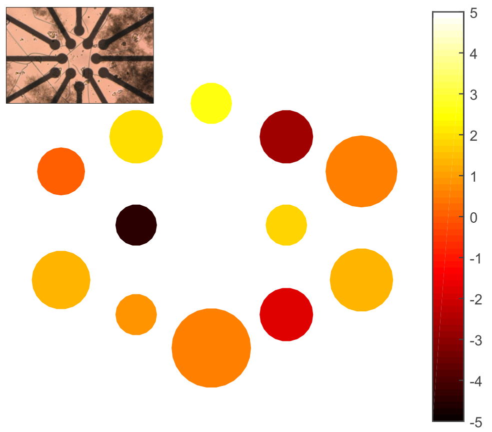


(f) 7 node (sampled) network - Spectral Radius vs. Leak Rate

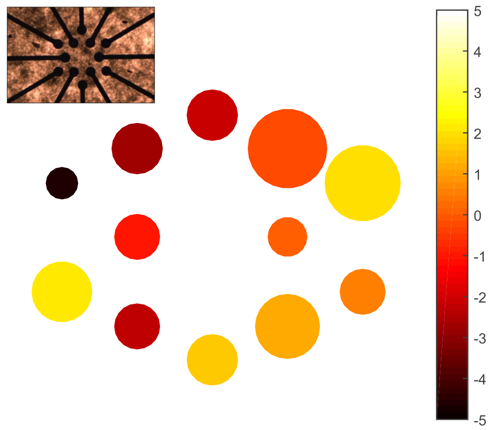
Fig. 5: Each plot displays 20 evolved Echo State Networks with network sizes of; 7 nodes (a & b), 50 nodes (c & d), and 50 nodes with only 7 nodes being used to train/form the output (e & f). The parameters under evolution are input scaling, leak rate and spectral radius. The colour map shows test accuracy of each evolved parameter set.



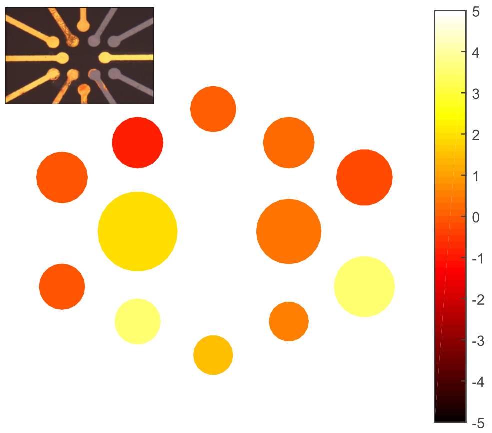
(a) 0.53% PMMA



(b) 0.71% PMMA(1)



(c) 0.71% PMMA(2)



(d) 1% PBMA

Fig. 6: Evolved circular electrode plot: Each plot shows how frequently an electrode is selected as an input (circle size) for each material, and the average input weight (circle colour), i.e. average voltage intensity, converged upon through evolution.

has compared to previous activations. The final parameter, leak rate, is used to match the speed of the reservoir's dynamics to the task input and/or output, essentially applying additional filtering to the activation of each node [13].

Fig.5 shows the relationship between parameters and task performance for each evolved network. By visualising the evolved parameters and their relative performance, we can determine the desired dynamics for a given task and suggest what dynamics need to be exploited within the material. From the graphs we can see each network (independent of size) typically possesses the following for the *Iris* task:

- Low spectral radius; i.e. does not require a long memory (output depends upon current input) and the reservoir is stable.
- Input scaling around 1; neurons sometimes saturate and become nonlinear (see 50-node, Fig.5c). However, there is more variation in dynamics in the 7-node and sampled networks.
- Large variation in leak rates (or close to 1); suggesting the parameter does not significantly effect performance – as expected with other parameters that reduce the reservoirs memory capacity. See Fig.5b, 5d and 5f.

These evolved dynamics align closely with the known *unperturbed* electrical properties of our materials; both stable with modest non-linear *I-V* characteristics. This might suggest why the *in materio* reservoirs perform similarly to *in silico* reservoirs, even after relatively few iterations (typically <50 generations).

Discovering task-specific parameters for simulated reservoirs – prior to physical implementation – can reduce experiment times. It can also inform the experimenter as to what tasks are reasonable to attempt. With future experiments using new materials, e.g. using materials with dynamic structure, these desirable dynamic traits can "seed" experiments with known configurations that produce such dynamics.

Mapping the dynamical relationships between different architectural reservoir systems can be enlightening, not only to benchmark systems but to help identify what electrical properties of the material respond well to reservoir-style training. However, identifying exactly what properties in the material are being exploited by evolution is challenging. As discussed in [6], analysing and modelling the nanotube structures and electrical pathways being utilised is difficult. In Fig.6 we introduce a simple visualisation that highlights areas of interesting activity frequently exploited by evolution. The figure displays the electrode arrangement for each material, showing what frequency an electrode is selected as an input (circle size) and what average voltage value is supplied (circle colour) across different evolutionary runs.

This simple visualisation correlates visual carbon nanotube groupings with areas of possible interest. It also highlights regions where there might be weak connectivity between electrodes. For example, in Fig.6a, the low frequency at which an electrode is chosen as a stimulus source (shown in circle size) may indicate local isolation in that region of the network/material. This quick visualisation is a useful tool in identifying materials with limited connectivity or homogeneity, as dispersion of the carbon nanotubes is largely stochastic.

VIII. Conclusion

In this investigation we show that adding on the reservoir framework to the evolution *in materio* technique, materials experience no degradation in performance. This result is significant as the reservoir framework allows both access to the temporal domain and provides a level of general-purpose functionality not present before. This is demonstrated across several materials, including a control (a resistor array) and similar materials used in comparison work. This initial experiment demonstrates that reservoir computing *in materio* (RCiM) can *outperform* evolution *in materio* (EiM), and in some cases match the performance of *in silico* techniques.

The final section of this work presents visualisation tools to aid understanding of the required task dynamics. These tools can be used to identify suitable tasks for the material, and identify regions of interesting activity/conductivity in the material often exploited by evolution.

Acknowledgements

Matthew Dale is funded by a Defence Science and Technology Laboratory (DSTL) PhD studentship. The authors thank the EU NASCENCE Project for providing the SWCNT materials used in this work.

References

[1] A. Adamatzky, L. Bull, and B. D. L. Costello. *Unconventional computing 2007*. Luniver Press, 2007.

[2] S. Bose, C. Lawrence, Z. Liu, K. Makarenko, R. van Damme, H. Broersma, and W. van der Wiel. Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nature nanotechnology*, doi:10.1038/nnano.2015.207, 2015.

[3] H. Broersma, J. F. Miller, and S. Nichele. Computational matter: Evolving computational functions in nanoscale materials. In A. Adamatzky, editor, *Advances in Unconventional Computing*, volume 2, pages 397–428. Springer, 2017.

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

[5] M. Conrad. The price of programmability. *The universal turing machine a half-century survey*, pages 261–281, 1995.

[6] M. Dale, J. F. Miller, and S. Stepney. Reservoir computing as a model for in-materio computing. In A. Adamatzky, editor, *Advances in Unconventional Computing*, volume 1, pages 533–571. Springer, 2017.

[7] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer. Evolving carbon nanotube reservoir computers. In *Unconventional Computation and Natural Computation: 15th International Conference, UCNC 2016, Manchester, UK, July 11-15, 2016*, pages 49–61. Springer International Publishing, 2016.

[8] M. Dale, S. Stepney, J. F. Miller, and M. Trefzer. Reservoir computing in materio: An evaluation of configuration through evolution. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.

[9] M. J. Embrechts, L. A. Alexandre, and J. D. Linton. Reservoir computing for static pattern recognition. In *17th European Symposium on Artificial Neural Networks (ESANN 2009)*, 2009.

[10] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

[11] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

[12] D. M. Kline and V. L. Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4):310–318, 2005.

[13] M. Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.

[14] M. Lukoševičius, H. Jaeger, and B. Schrauwen. Reservoir computing trends. *KI-Künstliche Intelligenz*, 26(4):365–371, 2012.

[15] W. Maass. Liquid state machines: motivation, theory, and applications. *Computability in context: computation and logic in the real world*, pages 275–296, 2010.

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

[17] M. Massey, A. Kotsialos, D. Volpati, E. Vissol-Gaudin, C. Pearson, L. Bowen, B. Obara, D. Zeze, C. Groves, and M. Petty. Evolution of electronic circuits using carbon nanotube composites. *Scientific Reports*, 6, 2016.

[18] J. F. Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In *NASA/DoD Conference on Evolvable Hardware 2002*, pages 167–176. IEEE, 2002.

[19] J. W. Mills. Polymer processors. Technical report, Technical Report TR580, Department of Computer Science, University of Indiana, 1995.

[20] J. W. Mills. The nature of the extended analog computer. *Physica D: Nonlinear Phenomena*, 237(9):1235–1256, 2008.

[21] M. Mohid, J. F. Miller, S. Harding, G. Tufte, O. R. Lykkebø, M. K. Massey, and M. C. Petty. Evolution-in-materio: Solving machine learning classification problems using materials. In *PPSN XIII, Parallel Problem Solving from Nature*, pages 721–730. Springer, 2014.

[22] M. Mohid, J. F. Miller, S. L. Harding, G. Tufte, M. K. Massey, and M. C. Petty. Evolving solutions to computational problems using carbon nanotubes. *International Journal of Unconventional Computing*, 11(3/4):245–281, 2015.

[23] L. A. Rubel. The extended analog computer. *Advances in Applied Mathematics*, 14(1):39–50, 1993.

[24] B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European symposium on artificial neural networks*. Citeseer, 2007.

[25] S. Stepney. The neglected pillar of material computation. *Physica D: Nonlinear Phenomena*, 237(9):1157–1164, 2008.

[26] A. Vargha and H. D. Delaney. A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.