
Book Section:
Dale, Matthew Nicholas, Stepney, Susan orcid.org/0000-0003-3146-5401, Miller, Julian Francis orcid.org/0000-0002-7692-9655 et al. (1 more author) (2016) Reservoir Computing in Materio:An Evaluation of Configuration through Evolution. In: IEEE Symposium Series on Computational Intelligence (SSCI), 2016. IEEE.

https://doi.org/10.1109/SSCI.2016.7850170

Reservoir Computing *in Materio*: An Evaluation of Configuration through Evolution

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*—Recent work has shown that computational substrates made from carbon nanotube/polymer mixtures can form trainable *Reservoir Computers*. This new reservoir computing platform uses computer based evolutionary algorithms to optimise a set of electrical control signals to induce reservoir properties within the substrate. In the training process, evolution decides the value of analogue control signals (voltages) and the location of inputs and outputs on the substrate that improve the performance of the subsequently trained reservoir readout.**

Here, we evaluate the performance of evolutionary search compared to randomly assigned electrical configurations. The substrate is trained and evaluated on time-series prediction using the *Santa Fe* Laser generated competition data (dataset A). In addition to the main investigation, we introduce two new features closely linked to the traditional reservoir computing architecture, adding an evolvable *input-weighting* mechanism and a reservoir *time-scaling* parameter.

The experimental results show evolved configurations across all four test substrates consistently produce reservoirs with greater performance than randomly configured reservoirs. The results also show that applying both input-weighting and time-scaling simultaneously can provide additional *tuning* to the task, improving performance. For one material, the evolved reservoir is shown to outperform – for this task – all other hardware-based reservoir computers found in the literature. The same material also outperforms a simple evolved simulated *Echo State Network* of the same size. The performance of this material is reported to be both consistent after long time-periods and after reconfiguration to other tasks.

I. Introduction

The future of conventional computing depends upon overcoming some fundamental engineering hurdles, such as dealing with an exponential growth in design complexity, the physical limits of transistor size, overcoming the processor-memory transfer (von Neumann) bottleneck, and satisfying an increasing desire for massively parallel, low-power, robust and fault-tolerant computing. The field of Unconventional Computing provides a unique insight into *non-standard* forms of computation where many of these engineering hurdles can be reduced, or avoided. In this work, we present an unconventional computing system where *computation* is extracted from a physical substrate in response to signals selected through artificial evolution. The substrate performing computation requires almost no design expertise, no separate memory entities, has natural parallelism, and requires little power (mW).

The theoretical framework of the proposed hardware-based Reservoir Computer is derived from two disciplines of research; *Evolution in Materio* and *Reservoir Computing*. Evolution *in materio* attempts to evolve physical computational machines from often design-less and unconstrained materials through *computer controlled evolution* [1]. Similar conceptual ideas can be seen in and around the cybernetics movement of the 1940s through pioneering cyberneticians such as Gordon Pask and Stafford Beer (see [1]). However, not until Thompson [2] exploited the low-level physics of silicon-based electronic devices through computer controlled evolution was it so distinctly demonstrated. Thompson's work showed that blind evolution could harness the unknown physical properties of modern electronic devices to create often unusual solutions. Harding *et al.* [3] continued this work, refining the technique and introducing new substrates, such as a liquid crystal display, magnetic quantum dots, a crystal lattice and an optical device. More recently, the EU-funded NASCENCE project [4] developed new evolvable nanosystems and unconventional hardware interfaces, including carbon nanotube based composites (with static and dynamic structures), disorganised gold-nanoparticle networks, and a bespoke computing platform [5]–[8]. The field of Reservoir Computing was originally conceived from two complementary independent investigations; designing a computational model for real-time continuous cortical microcircuits (Liquid State Machine) [9], and an efficient technique for training discrete artificial recurrent neural networks (Echo State Networks) [10]. After its inception, the reservoir model emerged as a potential computational model for many dynamical systems and has been applied to several systems, such as a bucket of water [11], optoelectronic and photonic systems [12]–[14], and memristive networks [15], [16].

In this paper, we compare the performance of evolutionary search *versus* random search to configure substrates into functional reservoir computers. To configure a substrate into a working reservoir a variety of parameters exist, such as the placement of task inputs and outputs on the electrode array interfacing the material, the placement of additional stimulation (referred to as *control signals*) and the voltage value of those signals. In previous work the values and choices of these parameters have been evolved using an evolutionary algorithm. Here we evaluate if there is a computational advantage to using evolution compared to just using randomly assigned parameters. The assumption that evolution is efficient at creating reservoirs in *materio* is currently unproven. Configuration

through random search has been documented only once in the evolution *in materio* literature, using Harding's evolvable Liquid Crystal Display [17]. In that work, Harding concluded that using random search alone was not sufficient to create the desired non-linear functions for that particular hardware platform. This assumption appears to have carried forward to other platforms within the NASCENCE project without further investigation. Conducting a separate investigation into random search for this new computational machine is necessary as there are fundamental differences in hardware and training methods.

After the evaluation of the search method, we introduce two new features based on counterparts found within simulated reservoirs: (i) *Input weighting*, a new input encoding analogous to traditional neural network input weighting where the input is directed to multiple locations on the substrate (with an assigned weight/scaling factor) rather than to a single location; (ii) *Time scaling*, a time-scale parameter that allows the search process to adapt the response of the material towards the time-scales of the task, reducing the mismatch between the natural time-scale of the material and the time-scale of the task.

II. Reservoir Computing Model

The reservoir model presents us with an abstract theory of computation that allows us to extract and exploit real-time computation from an analogue system. The model used is derived from the sub-fields *Echo State Networks* and *Liquid State Machines*.

The echo state network approach has become a simple and efficient training mechanism that removes the often cumbersome, internal gradient-descent based training used in traditional recurrent neural networks. In echo state networks, the discrete state of each neuron can be mathematically represented as an "echo" of its input and state history. This flavour of reservoir typically consists of a fixed random network of sigmoidal neurons exhibiting certain desirable properties, such as sparse connectivity and a fading memory (the *echo state property* [10]). The network is input-driven by a one- or multi-dimensional signal and its collected neuron activations are trained using a simple linear readout layer. Despite removing the internal training mechanism, echo state networks are found to be very competitive, simplifying the training process and avoiding expensive update cycles experienced in traditional recurrent networks.

The liquid state machine approach emerged as a model of computation for real-time adaptive (learning) computational systems. The motivation arose from a desire to create a model that could more accurately describe the continuous-time computational processes of biological neural networks compared to the Turing and other attractor-based models. For the purposes of our application, there are attractive features of this model, discussed in [18]: (i) heterogeneous behaviour can increase computational power; (ii) a non-linear projection of the input into a high-dimensional state space (similar to a *kernel* function) can theoretically lead to a *universal function approximator*, if the *Separation* and *Universal approximation* property is present; (iii) computational functions are "liquid", encompassing continuous time and states, rather than exhibiting discrete finite states, leading to a unique generalisation capability; (iv) the computational model promotes and provides a framework for "the invention of radically different artificial computing devices that exploit, rather than suppress, inherent properties of diverse physical substances" [18].

Combining the theory and practices from both approaches, we can conveniently apply the reservoir model on an evolvable substrate given three requirements: (i) a material that can exhibit desirable dynamical properties (fading memory, separation, and the universal approximation property) either naturally, or when given perturbation via external stimulus; (ii) a method of observation to access and separate localised states of the system; (iii) a mechanism to linearly combine and train states to perform a desired computational task. This new evolvable hardware-based reservoir computer was first demonstrated in [19].

The reservoir model can represent any excitable non-linear medium that produces a high-dimensional projection of the input $u(.)$ into reservoir states $x(.)$. In a conventional echo state network, the reservoir state update equation $x(n)$ is represented as:

$$x(n) = f(W_{in}u(n) + Wx(n-1) + W_{fb}y(n-1)) \quad (1)$$

where the weight matrices (W_{in}, W, W_{fb}) are collections of sparse connection weights to inputs (W_{in}), outputs (W_{fb} feedback connections), and internal neurons (W). The final trained output $y(n)$ is given when the reservoir states $x(n)$ are combined with the trained readout layer W_{out}:

$$y(n) = W_{out}x(n) \quad (2)$$

A reservoir primarily acts as an adaptable non-linear filter, performing non-linear functions on input data that can be subsequently extracted using a trained linear readout. This simplification of the training process avoids the inefficiencies of classical recurrent network training, but also implies the reservoir should be sufficiently rich to project the input into a high-dimensional space and therefore approximate many functions. This is sometimes referred to as the "kernel quality": the ability to separate features within the input (known as the *separation* property), or the functional degrees of freedom that exist in the reservoir (known as the *universal approximation* property) [20].

To apply the reservoir model to our system, we have to adapt the state update equation (1) to represent an observed state of the system. The observed reservoir state $x(n)$ combines the *continuous* (t) material and the *discrete* (n) observation function as:

$$x(n) = \Omega(\mathcal{E}(u(t), u_{config}(t))) \quad (3)$$

where $\Omega(n)$ is the observation of the macroscopic material behaviour (converted from analogue to digital), and $\mathcal{E}(t)$ the microscopic material function when driven by the input $u(t)$ and other evolved control signals $u_{config}(t)$ (converted from digital to analogue).

These additional control signals $u_{config}(t)$ are used to tune the electrical characteristics of the material. This can be viewed as perturbing the material into different dynamical regimes through the manipulation of electrical pathways. The hypothesis is that evolved control signals, and the placement of inputs and outputs, can alter the *quality* of the reservoir, tuning properties such as the *separation*, *approximation* and *echo state* property of the material function $\mathcal{E}(t)$.

III. Reservoir Parameters

When designing a reservoir some level of expertise is required to set parameters to task-dependent values. These values directly affect the reservoir's internal dynamics, memory, and general responsiveness. In an unconventional reservoir, manipulating the same desirable properties may require additional, and somewhat different parameters and techniques. For example, in an optoelectronic system it may require the tuning of a bias that controls the non-linearity of a signal modulator [21]. The unique parameters used in our *in materio* system describe the role and value of each electrode, for example, is an electrode assigned to be an input, output, or a control signal. These flexible parameters are selectively chosen through evolution to improve the performance of the readout layer. This follows the evolution *in materio* hypothesis that applying external stimuli, or some means of configuration (structural or electrical) can favourably manipulate the substrate's computational ability to solve a task. In addition to these parameters, we introduce here two new evolvable features from the traditional reservoir model, and evaluate if any significant performance boost can be acquired.

A. Input Weighting

The standard input mechanism used in the evolution *in materio* technique is to assign each input to a single electrode. In this experiment, we change the *one-for-one* input mechanism (Fig. 1a) to a *one-to-many* input mechanism (Fig. 1b) where the task input is supplied to multiple electrodes on the material, each being multiplied by some *weight*. This technique is more typical of the traditional reservoir computing method, where each input source is connected to the network via an input weight matrix W_{in} (see eqn.(1)). No experimental data, or discussed intuition on this type of input mechanism, is shown in the evolution *in materio* literature. The hypothesis here is that adding multiple signal sources could promote more complex interactions, activating regions where the material may be electrically weakened, or isolated from the input.

The input weights for our system are chosen through evolution and are bounded between $[-5\text{V}, 5\text{V}]$. When input-weighting is used, the control signals are not used. This is due to current hardware limitations on the size of the electrode array (12 electrodes). In total, evolution is restricted to 5 weighted inputs at any one time on the electrode array. In an ideal scenario, using both a weighted input mechanism and controls signals may be desirable, but not realistic on the current size of array.

B. Time Scaling: The Leak Rate Parameter

The ability to adjust the temporal response of the reservoir in respect to both the input and desired output can be advantageous. A simple example is to adjust the internal time-scale of the reservoir to the sampling rate at which the data was collected [22]. Time-warping invariant ESNs (TWIESN) do this when sampling from continuous data to discrete data overcoming common time-warping problems within recognition tasks [23]. Multiple time-scaling methods have been investigated for reservoir computing, including input and output re-sampling and time-scaling at precise points within the state collection, e.g. before and after any non-linearity is introduced [24].

In our physical reservoir system, the material will function at a natural time-scale which may or may not be adaptable. Developing a method to match the material and task time-scales could offer additional improvements in performance. To adjust the time-scale of the proposed reservoir system, we apply and adapt an external *Leak Rate* parameter derived from *Leaky Integrator* Echo State Networks (LI-ESN) [22], [24]. To fit the practicalities of the system, leaky integration has to be performed after the observation function. This effectively turns the leak rate parameter α into a simple adjustable digital low-pass filter, producing a smoothing effect which controls the speed of the reservoir's dynamics. This *filtered* reservoir state is:

$$\tilde{x}(n) = (1-\alpha)x(n-1) + \alpha\Omega(\mathcal{E}(u(t), u_{config}(t))) \quad (4)$$

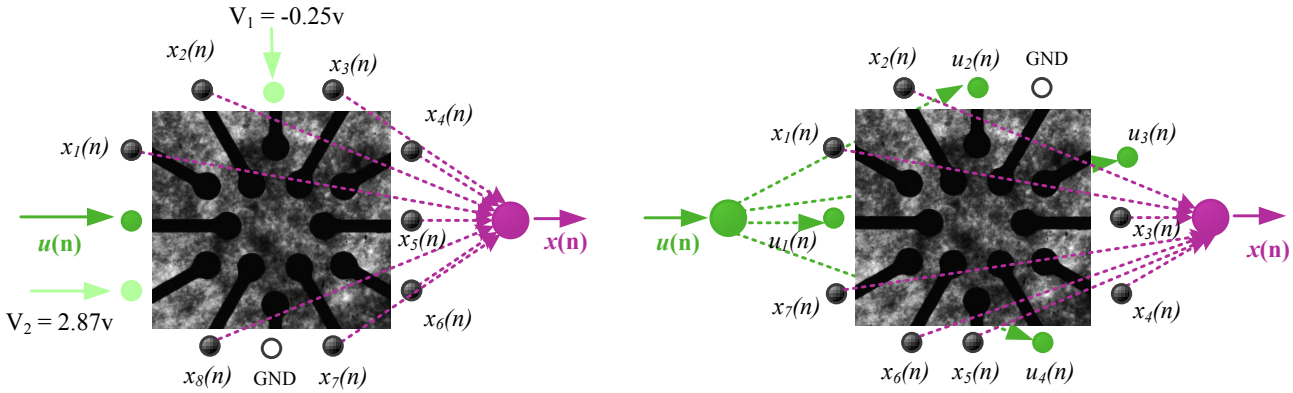
The parameter α has a range between [0,1]; it neither retains, nor leaks beyond the original boundaries of $x(n)$. When time-scaling is not used, $\alpha = 1$, and eqn.(4) reduces to eqn.(3).

IV. Experimental Set-up

The evolutionary training and evaluation of the material is conducted on a digital computer (see Fig. 2). The computer communicates with the material through two National Instruments Data Acquisition (DAQ) cards. The output DAQ card converts the evolved configuration into analogue voltages (task input and control signals) and digital control signals (describing the signal mapping). At the same time, the input DAQ card is set to receive analogue voltages. The number of input signals recorded by the card determines the number of *reservoir states* available from the material. Both input and output cards route to a 16×16 cross-point switch that converts the evolved mapping into the final electrode configuration.

A. Materials

In this experiment we investigate four substrates provided to us by the NASCENCE project. The first two test substrates consist of Single-Wall Carbon Nanotubes (SWCNT) with concentrations of 0.53% and 1% (w.r.t. weight) mixed with poly-butyl methacrylate (PBMA). The third consists of a SWCNT concentration of 0.1% mixed with poly-methyl methacrylate (PMMA). Each substrate is dissolved in anisole with approximately a 20ml mixture dispensed onto the electrode array, which is then heated until dry. A gold resistor array is used



(a) Static Voltages (b) Input Scaling

Fig. 1: Examples of the two input mechanisms used; a) configuration through static voltages (based on evolution *in materio* technique), and b) using multiple weighted inputs to various locations on the array (based on reservoir model). In the input-weighting scheme each input $u_i(n)$ is multiplied by a weight stored in the genotype.

as the fourth (control) substrate. The array is patterned onto a glass slide using etch-back photo-lithography. The resistor array was found to be a competitive reservoir when applied to other benchmark problems in [19].

V. Time-Series Prediction

In this experiment we have chosen a time-series prediction benchmark often used in reservoir computing literature. The task is to predict the next value of the Santa Fe time-series Competition Data (dataset A)[1]. This dataset holds original source data recorded from a Far-Infrared-Laser in a chaotic state. In the training process the first 5000 values of the dataset are used. This is sub-divided into three sets: 2500 values for the reservoir weight training process (training set), 1250 for the evaluation of each trained reservoir (validation set), and 1250 values to re-evaluate the final evolved reservoir (test set). The first 50 values of each sub-set are discarded as an initial washout period. Before applying the datasets to the material, a simple evaluation of task complexity was conducted. When comparing the original input and output of the test set, i.e. $E(u(n), y_{target})$, a *Normalised Root Mean Squared Error* (NRMSE) = 0.9771 was achieved. Using the linear model ($y = W_{out}u(n)$) trained on the target y_{target} an NRMSE = 0.9241 was achieved. These results imply a significant level of additional processing is required by the material to reduce the NRMSE.

VI. Training Process

A. Representing a Material Configuration

The mapping that constitutes an electrical *configuration* of the material is represented as a digital, 22-gene (genotype) string of integers and floating-point numbers. The first 12 genes in the genotype hold the functional role of each electrode on the array, i.e. whether an electrode is an input, output, or an additional control signal/weighted input. The next 4 genes represent the floating-point values for each control signal, or, the *weight value* if the input-weighting mechanism is used. An additional 4 *inactive* genes are added to allow evolution to dynamically select the size of the reservoir, i.e. the number of material states in use. The final two genes in the genotype hold the floating-point value for the time-scaling parameter (α) and the weight value for the one input that is always required.

The reservoir model provides flexibility in how many output electrodes (reservoir states) can form the task output. This differs from the evolution *in materio* technique where the number of output electrodes is predefined and task-dependent: an individual electrode will typically form one task output. To let evolution exploit this flexibility we have added inactive genes to the genotype. To implement this in hardware, we use all available channels on the 16-channel cross-point switch. This leads to a maximum of 4 inputs (or 4 control signals) and a maximum of 10 outputs that can be re-routed from the DAQ cards to the electrode array. When combined with the task input and ground signal, this creates a pool of 16 possible connections that can be mapped onto each electrode on the 12 electrode array. This collection of all possible connections is stored in the genotype as *active* and *inactive* genes. Here is an example of two individuals that have different genotypes in the same population: The first genotype consists of a ground connection, the task input, 3 control signals and 7 output electrodes – these represent the 12 *active* genes mapped to the electrodes. The remaining unassigned connections, i.e. the 1 control signal and 3 outputs are stored as *inactive* genes. The second individual's genotype consists of a ground, 5 inputs (using the input-weighting scheme) and 6 output electrodes, leaving 4 redundant output connections stored as inactive genes.

B. Training the Material

The evolutionary algorithm used to find task-specific signal mappings is based on a *mutation only* (1 + 4) evolutionary

[1]Dataset available at [25] and directly through MATLAB's Neural Network Toolbox Sample Data Sets: http://uk.mathworks.com/help/nnet/gs/neural-network-toolbox-sample-data-sets.html

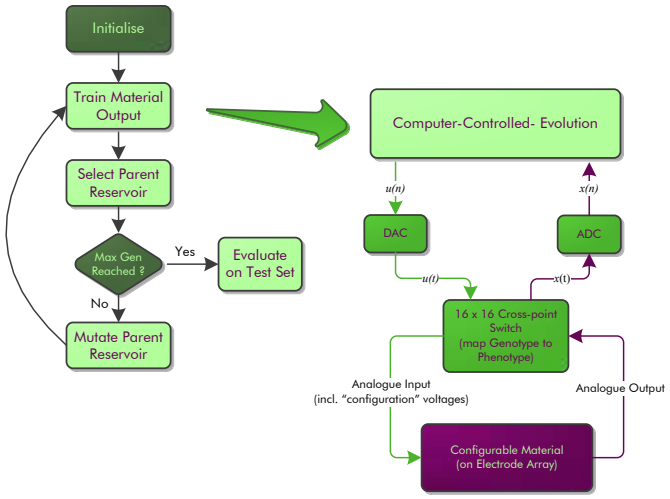


Fig. 2: The reservoir training process. (1) Material is stimulated by input(s) and control signals provided by computer controlled evolution. The *readout layer* W_{out} is trained on the *training* set. (2) The material is stimulated with the *validation* set. The trained readout is combined with the electrode readings and the reservoir is evaluated to give its fitness (error). (3) A new population is created from the reservoir readout and material configuration producing the smallest error. The process repeats until the maximum number of generations is reached. (4) The final evolved reservoir is re-evaluated on the *test* set, providing the final reported test error.

strategy (ES). This is compared to random search, where each configuration is a random initiation of the genotype selected from a uniform distribution of the minimum and maximum values possible for each gene. In both searches, a maximum of 750 fitness evaluations are conducted per run, for 10 runs. Each set of 10 runs typically takes 5 hours to complete for each material.

Other training techniques to find material configurations that create reservoir behaviour could be considered. Examples from evolution *in materio* include differential evolution [26], [27], particle swarm optimisation [27], genetic algorithms using crossover and mutation [7], [28], and global and local search [29].

The overall training process features four *signposts* (see Fig.2):

1) **Reservoir Creation**: The material configuration held in the genotype is loaded onto the cross-point switch, establishing communication between the DAQ cards and the material. The material is stimulated and the output response is trained on the *training set* using Ridge Regression (with *Tikhonov* regularisation) to create the output weights W_{out}:

$$W_{out} = y_{target} X^T (X X^T + \beta I)^{-1} \quad (5)$$

where $y_{target}(n)$ is the teacher signal, I the identity matrix, X the collected state matrix, and β the regularisation parameter. The material configuration and trained weights are then reapplied and the reservoir is evaluated on the *validation set*. The trained output of the reservoir is given by $y(n) = W_{out}\tilde{x}(n)$. The process is repeated for every individual in the population.
2) **Reservoir Selection**: The *validation set* error of each individual is compared. The reservoir producing the lowest error is selected and compared to the global best solution. If the error is below the global, the new reservoir becomes the parent and passes its genotype onto the next generation.
3) **Create a New Population**: A new population is created from the parent reservoir using a single random mutation. The mutational function on the genotype depends on a mutation probability assigned to: a one-for-one swap between active genes (20%), replace an active gene with an inactive gene (20%), or, adjust the value of the control signal/input weight/time-scaling parameter α (60%) using Gaussian noise (bounded by the min/max voltage range).
4) **Final Reservoir Assessment**: Once evolution is completed, the global best configuration and trained readout W_{out} is reloaded and reapplied. The material is then evaluated on the *test set*, giving the final reported NRMSE. This last stage tests the reservoirs generalisation to new data and its configuration stability, i.e. its repeatability and consistency of behaviour to the same configuration and stimulus.

TABLE I: The minimum and mean *test* error (NRMSE) for both search methods across 10 runs. (Standard deviation in brackets.)

Material	*Evo* Min.	*Rnd* Min.	*Evo* Avg.	*Rnd* Avg.
PMMA 0.1%	0.416	0.522	0.443 (.011)	0.651 (.106)
PBMA 0.53%	0.440	0.519	0.454 (.011)	0.656 (.111)
PBMA 1%	**0.242**	**0.439**	**0.306 (.056)**	**0.489 (.042)**
Resistor	0.498	0.582	0.536 (.023)	0.756 (.082)

VII. Experimental Results

In all substrates tested, evolution outperforms random search (see Table I). The results for the SWCNT/PBMA 1% material tend to outperform the other materials using both random and evolutionary search. On average, using random search with the SWCNT/PBMA 1% material still outperforms the best evolved resistor configuration. This result is significant when compared to the results found in the previous experiment [19] where the performance gap between the configured resistor and the configured materials was smaller than anticipated. These results also suggest improved reservoir performance/computational ability with a SWCNT density around the percolation threshold of 1%, as stated in [5].

The results for the newly introduced input-weighting and time-scaling features shown in Fig. 3 are somewhat mixed. When only applying the input-weighting mechanism, error improves on average compared to no features being used. When

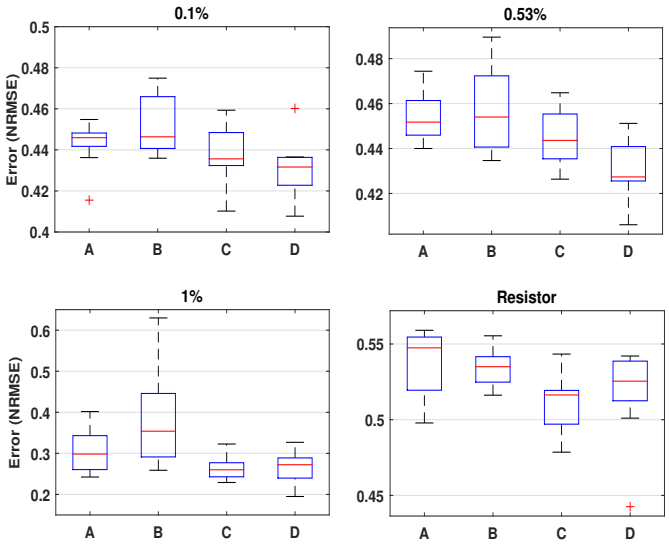


Fig. 3: Analysis of new features when added to each material: *test set* error decreases when input-weighting is used, and in some cases decreases further when time-scaling also used. Four combinations are assessed over 10 evolutionary runs: (A) No features used, i.e. applying only control signals with no time-scaling; (B) control signals and *time-scaling*; (C) *input-weighting* with no time-scaling; (D) both *input-weighting* and *time-scaling* applied to the reservoir.

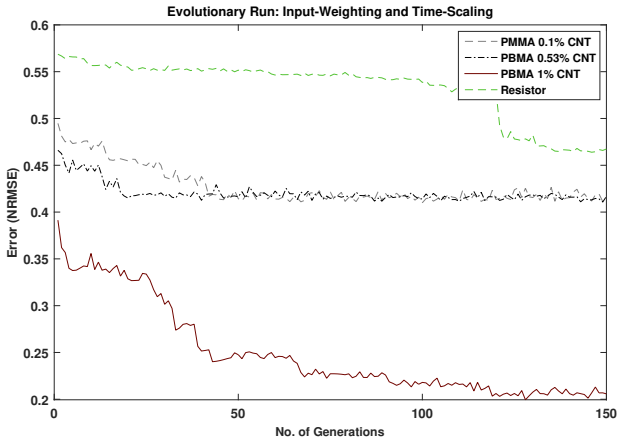


Fig. 4: An evolutionary run of each material with both time-scaling and input-scaling applied. The variation in error, both here and in Table I, suggests the complexity of the search space is somewhat dependent on nanotube density.

combining both input-weighting and time-scaling the *best* runs improve significantly, but the same improvement is not consistent across all runs. Despite only a 150 generations being performed, the best case found has an average improvement of 15%, and the worse case an improvement of 3%. The most notable performance increase is seen in the SWCNT/PBMA 1% material which already far outperforms the others. This implies both input-weighting and time-scaling can refine and tune both *poor* and *good* performance material reservoirs.

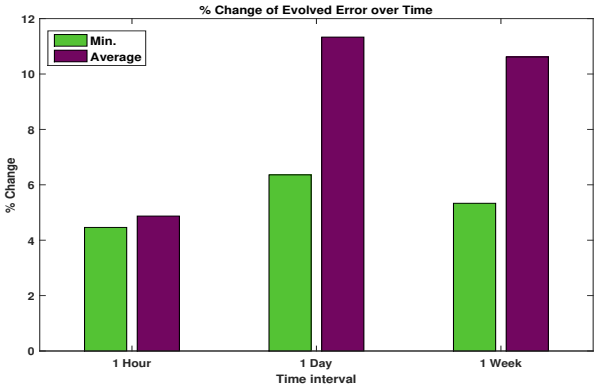


Fig. 5: A graph of the percentage change in error from the original evolved *test set* NRMSE of 0.195 over different time periods. This is shown for the best reservoir found with the SWCNT/PBMA 1% material using both time-scaling and input-weighting.

However, applying time-scaling by itself does not, on average, always offer an improvement in performance. This suggests some interesting relationship between time-scaling and the input-weighting mechanism that requires further investigation. Fig. 4 shows a single evolutionary run of each material using both input-weighting and time-scaling.

As a final experiment, a small consistency test was conducted showing the materials' change in performance over time. As the material is passive with a static structure, reproducing the same behaviour without large fluctuations in performance is assumed. The possibility of any *drift* in performance with time should be taken into consideration for any future applications, and in particular where the system is argued as a robust solution. In this consistency experiment, three separate sets of re-evaluations are carried out on the best performance reservoir found. Each evaluation set consists of 100 reassignments of the configuration and the trained readout after a time period of: i) an hour, ii) one day and iii) one week. Before the last set, the material was evolved several times to solve a different computational task. Fig. 5 shows the percentage change, i.e. % change = (new error – initial error) / abs(initial error)) $\times$ 100), between the first evolved error (NRMSE=0.195) and after each time-period. The results show *average error* after a time-period of a week increased by 10.6% from the initial NRMSE of 0.195 to 0.2157. This simple experiment provides an insight into the materials' drift in performance over time; an in-depth investigation into this phenomenon is still required.

VIII. Comparison to other reservoir systems

The proposed system, in the context of other reservoir systems, shows very competitive results. Table II shows a comparison between the in *materio* reservoir, simulated/numerical reservoirs and hardware reservoir computers. Three *evolved* (simulated) echo state networks (ESNs) are also provided. The evolvable parameters for these networks are; the *spectral radius* (controlling fading memory and dynamics), *input scaling*

TABLE II: Comparison table of other reservoir computing systems, with our system highlighted.

Reservoir Type	NMSE	NRMSE	Res. Nodes
Echo State Network (evolved)	0.009	0.098	50
Echo State Network [30]	0.018	0.134	50
Optoelec. (numerical) [31]	0.02	0.141	200
Optoelec. (numerical) [32], [33]	0.022	0.148	200
Mackey-Glass Oscillator [34]	0.023	0.151	50
In *materio* Reservoir	**0.038**	**0.195**	**7**
ESN (evolved and sampled)	0.042	0.205	7 (50)
Echo State Network (evolved)	0.055	0.235	7
Optoelec. (experimental) [32], [33]	0.106	0.326	200
Optoelectronic [13], [34]	0.123	0.35	400

(tuning non-linearity of the *tanh* neurons) and *leak rate* (time-scaling). Each evolved network uses the same evolutionary process and number of evaluations as the material. Two variations of these ESNs are also given; two networks where every node is used (i.e. 7 or 50 neurons), and a 50 node ESN with 7 nodes randomly sampled to form the trainable states.

Table II shows that the *in materio* (SWCNT/PBMA) reservoir outperforms all of the experimental optoelectronic reservoirs found in the literature on this task, with a significant reduction in the number of states used. The SWCNT/PBMA reservoir also outperforms the evolved 7 node ESN and the evolved/sampled 50 node ESN. The relationship in performance between the sampled and non-sampled networks could provide an insight into how the reservoir might scale with more electrodes, e.g. if the same relationship exists, a 50 node SWCNT/PBMA reservoir could produce an NRMSE < 0.098.

IX. Discussion of the reservoir model

In contrast to the output technique used in evolution *in materio*, the reservoir derives its output from the cumulative behaviours of multiple electrodes. The readout layer is used to selectively choose and separate interesting output signal patterns. This output mapping could lead to several advantages: (i) a layer of abstraction that extends the material's "programmability"; (ii) provide a more robust/fault tolerant output; (iii) suggest an output mechanism that can scale with hardware and task complexity; (iv) offer the opportunity to use multiple observation methods to define the task output, i.e. an output could combine electrical, thermal, optical and many more types of observation. However, a possible disadvantage to the reservoir implementation is a desire for more observable states, and therefore a more fine-grained observation mechanism to fully extract the materials computational complexity.

The conventional reservoir model, despite its advantages and suitability, does possess limitations; reservoirs sometimes deal poorly with *simultaneous* multiple time-scales [35]. A number of suggestions and demonstrations as to how this can be solved are discussed in [36], [37], such as creating hierarchical and modular reservoir systems. Implementing an extendible structure in hardware is an intriguing concept for several reasons; not only can it solve issues with time-scaling but it could result in a larger reservoir system with vast reservoir/material *diversity*. This diversity could come from multiple materials exhibiting different reservoir properties. A system like this could, in some sense, complete the vision of Miller's high-performance analogue computer made up of evolved materials that form functional primitives [1]. To extend the current system, both input-weighting and time-scaling would be useful features in implementation.

Tuning the dynamics and fading memory of our *in materio* reservoir is a difficult task resolved by evolving suitable material configurations. However, if configuration alone is not sufficient to induce the desired internal dynamics, other options can be explored from the reservoir computing literature. *Theorem 1.2* in [18] states that Liquid State Machines overcome the limitation of a fading memory if feedback is allowed to flow from the readouts back into the system. This is also a prominent mechanism found in Echo State Networks, as shown in eqn.(1). Feedback, and the flow of information in both directions, is a property often found exploited by the architecture of the neocortex [38]. Adding feedback to our system is an interesting avenue worth pursuing.

X. Conclusion

To understand and exploit the underlying physics of substrates requires a suitable computational model. Here we suggest and demonstrate the *Reservoir Computing* model as a possible candidate. The reservoir model combined with the evolution *in materio* technique observes, exploits and gives rise to different macroscopic material behaviours with no pre-knowledge of the system. It translates these behaviours through a trained linear output layer into a meaningful task output.

In this paper, we have demonstrated that configuring a material with random search is inefficient; evolution clearly provides a computational advantage across *all* the substrates investigated. However, a wider investigation is still required into what training algorithms could best be used to discover, or induce reservoir properties from materials. We have also introduced two new features (*time scaling* and *input weighting*) that were found to, in general, improve performance over the original technique. This improvement is greater in materials already possessing good reservoir properties, suggesting additional fine tuning of *in materio* reservoirs is possible. The results on the prediction task demonstrate competitive performance compared to other unconventional reservoirs, despite a large reduction in observable reservoir states. A simple consistency experiment also suggests the evolved solutions experience only a small drift in performance despite long time-periods and training on unrelated tasks. Finally, we have discussed a number of future modifications to the system, suggesting many routes to exploit the full advantages of this new substrate-based reservoir computing system.

ACKNOWLEDGEMENTS

This work was 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] J. F. Miller, S. Harding, and G. Tufte, "Evolution-in-materio: evolving computation in materials," *Evolutionary Intelligence*, vol. 7, no. 1, pp. 49–67, 2014.

[2] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined with physics," in *Evolvable Systems: From Biology to Hardware*. Springer, 1997, pp. 390–405.

[3] S. Harding, J. F. Miller, and E. A. Rietman, "Evolution in materio: Exploiting the physics of materials for computation," *Int J of Unconventional Computing*, pp. 155–194, 2008.

[4] H. Broersma, F. Gomez, J. Miller, M. Petty, and G. Tufte, "Nascence project: Nanoscale engineering for novel computation using evolution," *International Journal of Unconventional Computing*, vol. 8, no. 4, pp. 313–317, 2012.

[5] 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*, vol. 117, no. 13, p. 134903, 2015.

[6] D. Volpati, M. Massey, D. Johnson, A. Kotsialos, F. Qaiser, C. Pearson, K. Coleman, G. Tiburzi, D. Zeze, and M. Petty, "Exploring the alignment of carbon nanotubes dispersed in a liquid crystal matrix using coplanar electrodes," *Journal of Applied Physics*, vol. 117, no. 12, p. 125303, 2015.

[7] 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*, vol. doi:10.1038/nnano.2015.207, 2015.

[8] O. R. Lykkebø, S. Harding, G. Tufte, and J. F. Miller, "Mecobo: A hardware and software platform for in materio evolution," in *Unconventional Computation and Natural Computation*. Springer, 2014, pp. 267–279.

[9] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[10] 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*, vol. 148, p. 34, 2001.

[11] C. Fernando and S. Sojakka, "Pattern recognition in a bucket," in *Advances in Artificial Life*. Springer, 2003, pp. 588–597.

[12] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature Communications*, vol. 2, p. 468, 2011.

[13] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Optics Express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[14] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Communications*, vol. 5, p. 3541, 2014.

[15] M. S. Kulkarni and C. Teuscher, "Memristor-based reservoir computing," in *NANOARCH, 2012, IEEE/ACM International Symposium on Nanoscale Architectures*. IEEE, 2012, pp. 226–232.

[16] H. O. Sillin, R. Aguilera, H. Shieh, A. V. Avizienis, M. Aono, A. Z. Stieg, and J. K. Gimzewski, "A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing," *Nanotechnology*, vol. 24, no. 38, p. 384004, 2013.

[17] S. Harding, "Evolution in materio," Ph.D. dissertation, University of York, 2005.

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

[19] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer, "Evolving carbon nanotube reservoir computers," in *International Conference on Unconventional Computation and Natural Computation*. Springer, 2016, pp. 49–61.

[20] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural Networks*, vol. 20, no. 3, pp. 323–334, 2007.

[21] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Scientific Reports*, vol. 2, 2012.

[22] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural Networks*, vol. 20, no. 3, pp. 335–352, 2007.

[23] M. Lukoševičius, D. Popovici, H. Jaeger, and U. Siewert, "Time warping invariant echo state networks," Tech. Rep., 2006.

[24] B. Schrauwen, J. Defour, D. Verstraeten, and J. Van Campenhout, "The introduction of time-scales in reservoir computing, applied to isolated digits recognition," in *Artificial Neural Networks–ICANN 2007*. Springer, 2007, pp. 471–479.

[25] A. Weigend, *The Santa Fe Time Series Competition Data: Data set A, Laser generated data*, 1991 (accessed March, 2016). [Online]. Available: http://www-psych.stanford.edu/ andreas/Time-Series/SantaFe.html

[26] A. Kotsialos, M. Massey, F. Qaiser, D. Zeze, C. Pearson, and M. Petty, "Logic gate and circuit training on randomly dispersed carbon nanotubes." *International journal of unconventional computing.*, vol. 10, no. 5-6, pp. 473–497, 2014.

[27] E. Vissol-Gaudin, A. Kotsialos, M. K. Massey, D. A. Zeze, C. Pearson, C. Groves, and M. C. Petty, "Training a carbon-nanotube/liquid crystal data classifier using evolutionary algorithms," in *International Conference on Unconventional Computation and Natural Computation*. Springer, 2016, pp. 130–141.

[28] O. R. Lykkebo and G. Tufte, "Comparison and evaluation of signal representations for a carbon nanotube computational device," in *IEEE International Conference on Evolvable Systems (ICES 2014)*. IEEE, 2014, pp. 54–60.

[29] K. Greff, R. Damme, J. Koutnik, H. Broersma, J. Mikhal, C. Lawrence, W. Wiel, and J. Schmidhuber, "Unconventional computing using evolution-in-nanomaterio: neural networks meet nanoparticle networks," *Eighth International Conference on Future Computational Technologies and Applications, FUTURE COMPUTING 2016*, pp. 15–20, 2016.

[30] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE Transactions on Neural Networks*, vol. 22, no. 1, pp. 131–144, 2011.

[31] R. M. Nguimdo, G. Verschaffelt, J. Danckaert, and G. Van der Sande, "Reducing the phase sensitivity of laser-based optical reservoir computing systems," *Optics express*, vol. 24, no. 2, pp. 1238–1252, 2016.

[32] K. Hicke, M. A. Escalona-Morán, D. Brunner, M. C. Soriano, I. Fischer, and C. R. Mirasso, "Information processing using transient dynamics of semiconductor lasers subject to delayed feedback," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, no. 4, pp. 1 501 610–1 501 610, 2013.

[33] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature communications*, vol. 4, p. 1364, 2013.

[34] L. Appeltant, "Reservoir computing based on delay-dynamical systems," *These de Doctorat, Vrije Universiteit Brussel/Universitat de les Illes Balears*, 2012.

[35] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[36] H. Jaeger, "Discovering multi-scale dynamical features with hierarchical echo state networks," *Technical report No. 9*, 2007.

[37] Y. Xue, L. Yang, and S. Haykin, "Decoupled echo state networks with lateral inhibition," *Neural Networks*, vol. 20, no. 3, pp. 365–376, 2007.

[38] R. J. Douglas and K. A. Martin, "Recurrent neuronal circuits in the neocortex," *Current Biology*, vol. 17, no. 13, pp. R496–R500, 2007.