Using Neural Networks to Predict the Functionality of Reconfigurable Nano-material Networks

Klaus Greff†, Ruud van Damme*, Jan Koutník†, Hajo Broersma*, Julia Mikhal*,
Celestine Lawrence*, Wilfred van der Wiel*, and Jürgen Schmidhuber†
†IDSIA, USI-SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland
Email: {klaus,hkou,juergen}@idsia.ch
*Faculty of EEMCS, CTIT Institute for ICT Research, and MESA+ Institute for Nanotechnology
University of Twente, The Netherlands
Email: {r.m.j.vandamme,h.j.broersma,c.p.lawrence,w.g.vanderwiel}@utwente.nl

***Abstract*—This paper demonstrates how neural networks can be applied to model and predict the functional behaviour of disordered nano-particle and nano-tube networks. In recently published experimental work, nano-particle and nano-tube networks show promising functionality for future reconfigurable devices, without a predefined design. The nano-material has been treated as a black-box, and the principle of evolution-in-materio, involving genetic algorithms, has been used to find appropriate configuration voltages to enable the target functionality. In order to support future experiments and the development of useful devices based on disordered nano-materials, we developed simulation tools for predicting candidate functionalities. One of these tools is based on a physical model, but the one described and analysed in this paper is based on an artificial neural network model. The advantage of this newly presented approach is that, after training the neural network to match either the real material or its physical model, it can be configured using gradient descent instead of a black-box optimisation, speeding up the search for functionality. The neural networks do not simulate the physical properties, but rather approximate the nano-material's transfer functions. The functions found using this new technique were verified back on the nano-material's physical model and on a real material network. It can be concluded from the reported experiments with these neural network models that they model the simulated nano-material quite accurately. The differentiable, neural network-based material model is used to find logic gates, as a proof of principle. This shows that the new approach has great potential for partly replacing costly and time-consuming experiments with the real nano-material. Therefore, this approach has a high relevance for future computing, either as an alternative to digital computing or as an alternative way of producing multi-functional reconfigurable devices.**

Keywords–nano-material network; neural network; simulation; unconventional computation; evolution-in-nanomaterio.

I. Introduction and Motivation

This paper is an extended version of the preliminary work reported in [1] that was presented by the first author at the FUTURE COMPUTING 2016 meeting in Rome. It demonstrates parts of the research that has been carried out within the framework of the FP7-project NASCENCE: NAnoSCale Engineering for Novel Computation using Evolution [2]. More details on the conceptual ideas and outcomes of this project can be found in [3], including many references to recently published work within the framework of the project. In the NASCENCE project, that was funded by the European Community, disordered networks of gold nano-particles have been successfully used to produce reconfigurable logic, with a very high degree of stability and reproducibility [4]. More recently, similar results have been obtained within the NASCENCE project with networks consisting of composite materials based on nano-tubes [5]. These breakthroughs present a proof of principle that indicates very promising prospects for using such nano-materials to perform more complicated computational tasks. But there are still many challenges that have to be addressed, and the production of and experimentation with these nano-material networks is very costly and time-consuming. An example network that has been used in the experimental work of [4] is shown as an atomic-force microscopy (AFM) image in Figure 1. We apologise for the rather bad visibility that is in fact quite typical for AFM images with such a high resolution (the small dots represent nano-particles of about 20 nm in diameter).

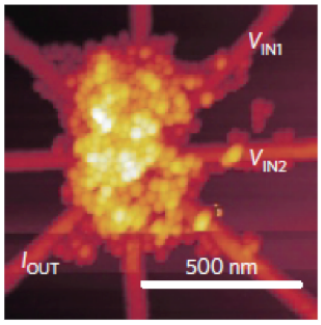


Figure 1. AFM image of one of the fabricated nano-particle networks in [4]

In order to predict candidate computational tasks of such networks, but avoiding the waste of scarce and expensive resources involved in experimentally exploring these nano-particle networks, we developed simulation tools for examining and predicting the capabilities of these nano-material systems. One of the considered simulation tools [6] is an extension of existing tools for simulating nano-particle interactions, like SPICE [7] or SIMON [8]. The latter tools are all based on Monte-Carlo simulations, using a physical model, and they

have been validated for known designed systems from literature consisting of very small numbers of particles. Although these methods can, in principle, handle arbitrary systems of any size, their scalability and use for networks consisting of hundreds of nano-particles is a serious issue. Moreover, nano-particle networks like the ones used in [4] to date cannot be produced according to a predefined specific design.

To illustrate this, an example of one of the fabricated nano-particle networks that was used in the experimental work of [4] is shown as an AFM image in Figure 1. The AFM image clearly demonstrates the disordered nature of the fabricated nano-particle networks. Therefore, due to their disordered nature, it is not possible to use a physical model to accurately describe and predict the properties of such nano-material systems.

In the earlier mentioned conference paper [1], an alternative approach has been introduced and analysed, with little detail due to the page limit. We provide more details of the approach and results here. This novel approach is based on training artificial Neural Networks in order to model and investigate the nano-particle networks. Neural Networks (NNs; [9] [10] [11]) have proven to be powerful function approximators and have, especially recently, been applied to a wide variety of domains with great success [12] [13] [14]. Being essentially treated as black-boxes themselves, NNs do not facilitate a better understanding of the underlying quantum-mechanical processes that take place in the nano-material. For that purpose, physical models are more appropriate. But, in contrast to physical models, NNs provide differentiable models and thus offer interesting possibilities to explore the computational capabilities of the nano-material. In the sequel, NNs will be used, in particular, to search for configurations of input voltages such that the nano-material computes different Boolean logic functions, such as AND, OR, NOR, NAND, and XOR.

To enable the exploration of the computational capabilities of the nano-material by an NN, the NN needs to be trained first with data collected from the measurements on the nano-material. Since a physical model and an associated validated simulation tool for the nano-particle networks have already been developed [6], such training data can be obtained from the simulated nano-material. This also provides an opportunity for predicting functionalities in small nano-particle networks that have not been fabricated yet. This in turn can inform electrical engineers on the minimum requirements necessary for obtaining such functionalities, without the burden of costly and time-consuming fabrication and experimentation. As soon as the NN has been trained, searching for arbitrary target functions is very fast, and can happen without any access to the nano-material or its physical model.

The rest of the paper is organised as follows. Section II provides some technical details on the gold nano-particle networks that have been used in the experimental work of [4]. This section also describes the choices that have been made and that form the basis for the physical-model based simulation tool of [6], combining a genetic algorithm with Monte-Carlo simulations for charge transport. Section III presents simulation results obtained with these tools for an example network. Section IV shows how an NN was trained for modelling the example network, using data collected from the physical-model based simulation tool. An analysis of the results is presented in Section V. Section VI is a newly added section that contains results of experiments that were performed after the FUTURE COMPUTING 2016 meeting. In these experiments a real material sample was used for collecting data to train an NN-model and predict functionalities, that were also validated afterwards using the real material. We finish the paper with a short section with some conclusions as well as an outlook to future work in Section VII.

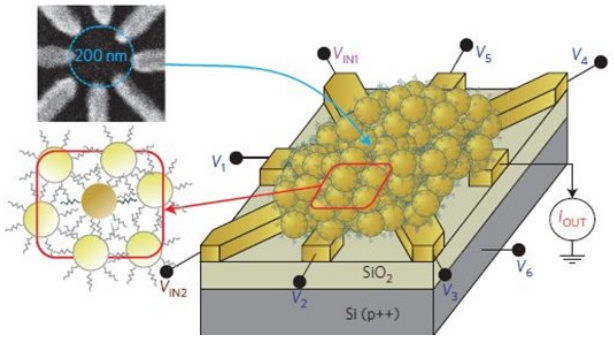


Figure 2. Illustration of a disordered network of gold nano-particles

II. Nano-particle Networks and their Simulation

In collaboration with the NanoElectronics group at the MESA+ institute of the University of Twente, networks consisting of commercially available nano-particles of size 5-20 nm consisting of gold, and junctions of alkanedithiol of length 1-3 nm have been produced. The alkanedithiols stick to the metal and can form junctions (tunnel barriers) between particles. Figure 2 shows an illustration of such a network. The central circular region is about 200 nm in diameter. Recall that an AFM image of a real network has been shown in Figure 1. In a later stage, other organic molecules have been tried as well, but this is not relevant for the physical model or the simulation tool, only for the setting of the parameter values.

The networks investigated in the experimental work of [4] are relatively large (in the order of a hundred particles) and disordered. An artist's impression of such a network is given in Figure 3. More details on the production process and the electrical characterisation of these networks can be found in [4].

Under energetically favourable circumstances, the transport of electrons in such nano-particle networks is governed by the Coulomb blockade effect [15]: transport is blocked, except at almost discrete energy levels; there exactly one electron can jump. The dynamics of such a system is governed by stochastic processes: electrons on particles can tunnel through junctions with a certain probability. For such systems, there are basically two simulation methods to one's disposal: Monte-Carlo Methods and the Master Equation Method [16] [17]. Since the number of particles is large, the Monte-Carlo Method

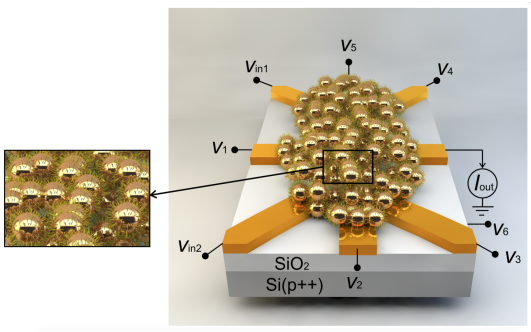


Figure 3. Artist's impression of a disordered nano-particle network

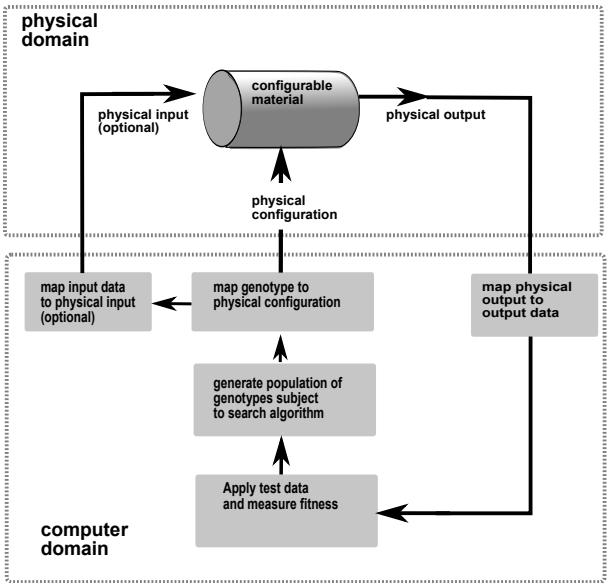


Figure 4. Concept of evolution-in-materio [18]

is the best candidate. This method simulates the tunnelling times of electrons stochastically. To get meaningful results, one needs to run the algorithm in the order of a million times. Doing so, the stochastic process gives averaged values of the charges, currents, voltages, etc. The approach for obtaining functionality is based on these physical effects combined with genetic algorithms for finding suitable settings of certain control voltages on configuration leads, as explained in more detail on the example network in the next section. This section concludes with a short explanation of the basic principles of evolution-in-materio and genetic algorithms.

Evolution-in-materio (EIM) is a term coined by Miller and Downing [18] that refers to the manipulation of physical systems using a form of computer-controlled artificial evolution [18]–[22]. It is a type of unconstrained evolution in which, through the application of physical signals, various intrinsic properties of a material can be heightened or configured so that a useful computational function is achieved.

The central idea of EIM is that the application of some physical signals to a material (configuration variables) can cause it to alter how it responds to an applied physical input signal and how it generates a measurable physical output (see Figure 4) [18].

Physical outputs from the material are converted to output data and a numerical fitness score is assigned based on how close the output is to a desired response. This fitness is assigned to the member of the population that supplied the configuration variables. Ideally, the material would be able to be reset before the application of new configuration instructions. This is likely to be important as without the ability to reset the material, it may retain a memory from past configurations. This could lead to the same configuration having different fitness values depending on the history of interactions with the material.

Mappings need to be devised which convert problem domain data into suitable signals to apply to the material. An input-mapping needs to be devised to map problem domain inputs to physical input signals. An output-mapping is required to convert measured variables from the material into a numerical value which can be used to solve a computational problem. Finally, a configuration-mapping is required to convert numerical values held on a computer into physical variables that are used to "program or configure" the material.

EIM is a bottom-up approach where the intrinsic underlying physics of materials is exploited as a computational medium. In contrast to a traditional design process where a computational substrate, e.g., silicon, is precisely engineered, EIM uses a bottom-up approach to manipulate materials with the aim of producing computation. Yoshihito discussed a closely related concept of "material processors" which he describes as material systems that can process information by using the properties of the material [23]. Zauner describes a related term which he refers to as "informed matter" [24]. It is interesting that inspection of much earlier research publications reveals that ideas similar to EIM, albeit without computers, were conceived in the late 1950s (particularly by Gordon Pask, see [25], [26]).

One of the key ingredients in EIM is the use of genetic algorithms, a form of computer-controlled artificial evolution. The main elements of a genetic algorithm are shown in Algorithm 1.

In evolutionary computing, the term genotype (or chromosome) is used to refer to the string of numbers that defines a solution to a search problem. The individual elements of the genotype are commonly referred to as genes. To solve a computational problem requires an assessment of how well a particular genotype represents a solution to the computational search problem. This is called a fitness function. The "survival-of-the-fittest" principle of Darwinian evolution is implemented by using a form of fitness-based selection that is more likely to choose solutions for the next generation that are fitter rather than poorer. Mutation is an operation that changes a genotype by making random alterations to some genes, with a certain probability. Recombination is a process of generating one or more new genotypes by recombining genes from two or more genotypes. Sometimes, genotypes from one generation are

Algorithm 1 Genetic Algorithm

1: Generate an initial population of size p. Set the number of generations g to 0
2: **repeat**
3: Calculate the fitness of each member of the population
4: Select a number of parents according to the quality of their fitness
5: Recombine some, if not all, of the parents to create offspring genotypes
6: Mutate some parents and offspring
7: Form a new population from the mutated parents and the offspring
8: Optional: promote a number of unaltered parents from Step 4 to the new population
9: Increment the number of generations $g \leftarrow g + 1$
10: **until** (g equals the number of generations required) **or** (the fitness is acceptable)

promoted directly to the next generation; this is referred to as elitism (see the optional step in Algorithm 1).

III. An Illustrative Example

As an example which is still relatively small and manageable, but shows interesting features, the described methods have been explored on the symmetric 4×4-grid consisting of the components shown in Figure 5.

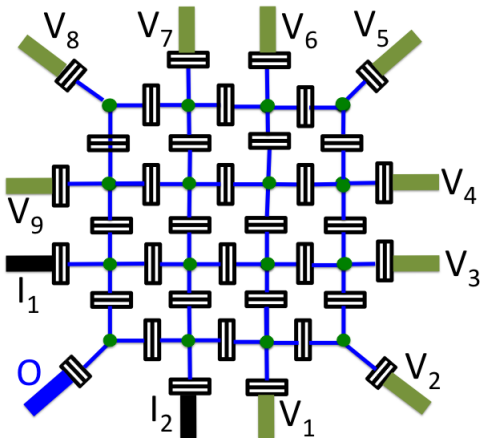
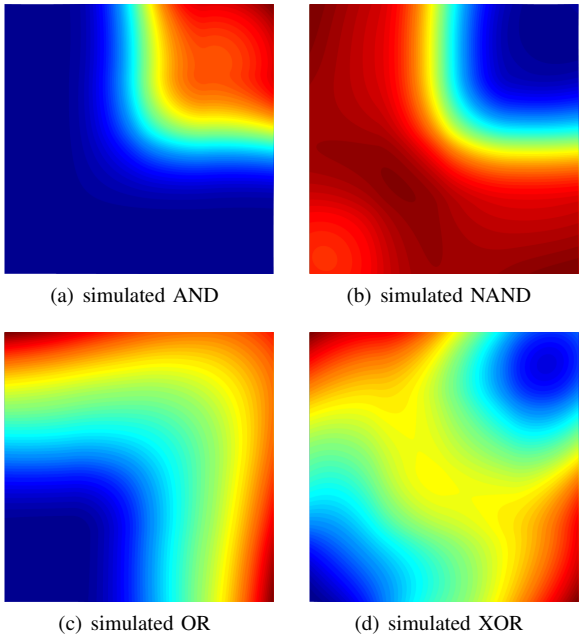


Figure 5. A symmetric 4×4-grid of 16 nano-particles with leads

In Figure 5, the 16 green dots represent the nano-particles; in between are the tunnelling junctions, with fixed C and R values. The two input leads and the single output lead are depicted as I_1 and I_2 and O, respectively. Voltages are applied to the configuration leads V_1-V_9, according to a genetic algorithm, and also to a back gate; this back gate is connected through tunnel barriers (a silicon oxide layer) to all nano-particles (for convenience we have not shown the back gate in the figure).

The fitness of the sets of configuration voltages is determined by how close the output for the four input combinations of the Boolean truth table is to the desired logic. More details can be found in [6].

1) Evolved Boolean Logic: Applying the developed simulation tool to the small network of Figure 5, it was possible to evolve all basic Boolean logic gates, using different computed (simulated and optimised) settings of the values of the free variables (the configuration leads voltages and the back gate voltage). The solutions for four of the cases, namely AND, NAND, OR and XOR, are illustrated as contour plots in Figure 6. The four plots are functions of the two input signals; the voltages of both inputs range from 0 to 10 mV, horizontally as well as vertically; the colour scheme ranges from blue for small values to red for high values of the output.



(a) simulated AND (b) simulated NAND

(c) simulated OR (d) simulated XOR

Figure 6. Contour plots of simulated evolved logic in the 4×4-grid

2) Discussion: From an electrical engineering point of view, the simulation results are very interesting, for several reasons. First of all, the example of Figure 5 can be configured into any of the basic Boolean logic gates, using only 16 nano-particles of size 5-20 nm. If one would like to design and build the same functionality with transistors, one would require at least 10 transistors. Secondly, in the designed circuit one would have to rewire the input and apply it at different places, whereas in the nano-particle network each of the input signals is applied at exactly one place. Even with current transistor sizes below 20 nm, the designed circuit would require the same or more space, and would dissipate substantially more energy.

It is interesting to note, that in the experiments with the real material samples [4], all basic Boolean gates were also evolved within an area with a diameter of around 200 nm, but with only six control voltages. However, currently these samples consist of 100-150 particles that are self-assembled into a disordered network.

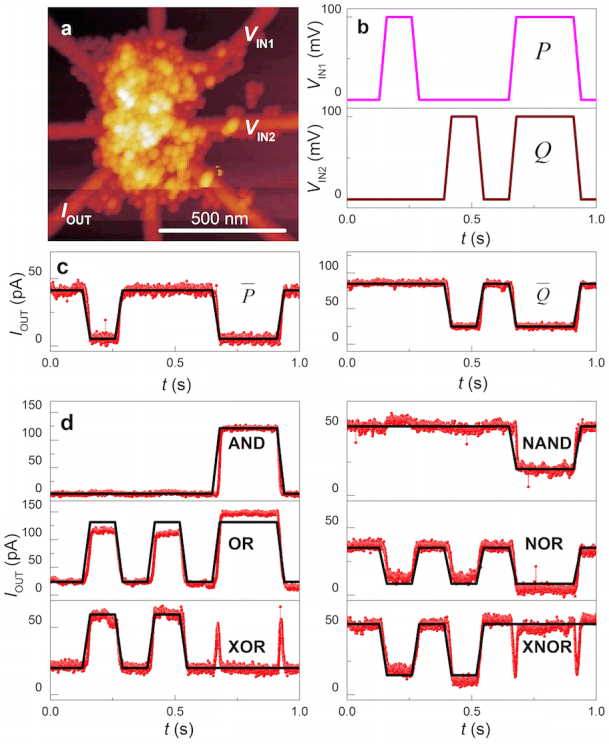


Figure 7. AFM image of an NP-network (**a**), the input voltages in mV applied to V_{IN1} and V_{IN2} (**b**) and the different logic outputs in pA read from I_{OUT} (**c** and **d**) [4]

In Figure 7(a) we see the same AFM image as before of one of the real networks that was used for the experiments in [4], where the two input electrodes and the output electrode are denoted by V_{IN1}, V_{IN2} and I_{OUT}, respectively. Time-dependent signals in the order of a hundred mV were applied to the input electrodes as illustrated in Figure 7(b), and a time-dependent current in the order of a hundred pA was read from the output electrode. The other five electrodes and the back gate have been used to apply different sets of static configuration voltages. Using a genetic algorithm, suitable sets of configuration voltages have been found to produce the output functions of Figure 7(c,d). Red symbols are experimental data, solid black curves are expected output signals (matched to the amplitudes of the experimental data). We observe two clear negators (inverters) for the input functions P and Q in Figure 7(c), and we observe a variety of Boolean logic gates in Figure 7(d), including the universal NAND and NOR gate. Supplementary work in [4] reveals that all these gates show a great stability and reproducibility. For the exclusive gates (XOR, XNOR) spike-like features are observed at the rising and falling edges of the (1,1) input, as might have been expected for a finite slope in the input signals. More details can be found in [4].

The remarkable thing here is not that we can produce logic gates using the electrical and physical properties of charge transport in neighbouring nano-particles. What is remarkable, is that we can do this with one and the same sample of a disordered nano-particle network in a circular region of about 200 nm in diameter, and by using only six configuration voltages. This shows the great potential for the approach. Note that a similar designed reconfigurable device based on today's transistor technology would require about the same space, and it would also require rewiring of the input signals to multiple inputs.

The experimental results as well as the simulations show the great potential for the approach, both in the bottom-up and top-down design regime. This could have a huge impact on future computing, either as an alternative approach to digital computing or as an alternative way to produce reconfigurable multi-functional devices, e.g., to support further down-scaling of digital components. Currently, we are not aware of any production techniques for constructing samples that come anywhere close to the 4×4-grid structure of Figure 5.

To obtain more insight in the underlying currents and physical phenomena, and with the long term goal to fully understand what is going on in terms of electron jumps and currents through these nano-particle networks, in [6] visualisation tools have been developed to analyse the processes that are taking place over time. In Figure 8, some pictures visualising the currents through the network are presented, in this case, as an example, when the network was configured as an AND. The amplitude of the currents is proportional to the area of the red arrows in the figure. The currents are all averaged over time. The tool also enables the calculation of variances, and it can show fast animations of the electron jumps as well. It is still an open problem to deduce an explanation for the patterns and jumps that cause the 4×4-grid to behave as a logic AND (or one of the other basic Boolean logic gates).

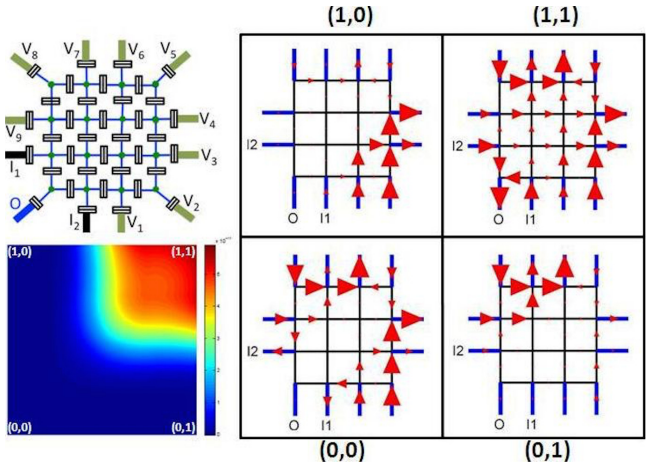


Figure 8. Averaged current patterns for simulated AND in the 4×4-grid

Figure 8 gives an impression of how complicated the traffic of electrons in such networks can get, and can hopefully in the future lead to more insight as to why they behave as logic. The plots do not explain anything as yet. We are currently trying to find macroscopic laws from the numerics, like real (first

order) phase transitions, but so far without success. We have made similar pictures for the other Boolean logic functions, but they are not relevant for this paper. Therefore, we omitted them until we are in a better position to explain the relationship between the flow pattern and the observed logic.

In the next two sections, the use of artificial NNs to simulate the nano-material will be explained, as well as how to use the data collected from the above physical-model based simulations to explore the functionalities in the example of the 4×4-grid.

IV. Neural Networks

The key idea of this paper is to show how to use an NN-model for approximating the mapping from the input voltages to the output current, by training the NN on many randomly chosen examples. By using this approach, the aim is that the NN becomes a differentiable model of the potentially complex structures and processes that take place inside the nano-material. While the example that we use throughout will only treat the search for Boolean logic functions performed by the nano-particle network, the future goal is to use the NN-models in a more general sense to assess:

- **Complexity** of the nano-material: based on the training process, one can estimate whether the material contains learnable, interesting structures, only noise (if the training process cannot minimise the error), or simple structures (if the training of the NN-model turns out to be trivial).
- **Predictability**: if the trained NN becomes a good model of the nano-material, then it can further be used to predict outputs of the nano-material for any given inputs, and for searching for target functionalities.
- **Function**: the NN-model can be used as a substitute for the real nano-material. One can search for functionality in the NN-model first instead of performing measurements in the real nano-material. This can be particularly useful if the nano-material has, e.g., a limited lifespan or in case the experiments in the lab are very costly and time-consuming.

An example of a specific NN that turned out to be suitable for our experiments is illustrated in Figure 9.

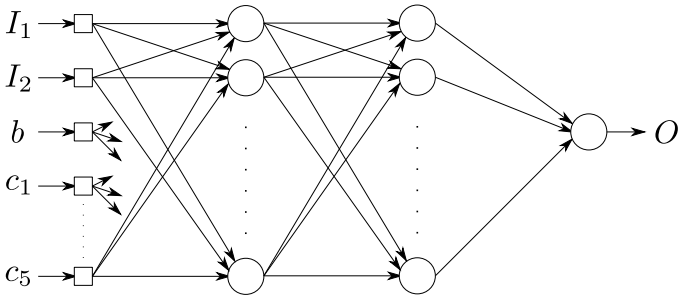


Figure 9. Illustration of an NN with two hidden layers

The NN of Figure 9 is used in the sequel for approximating the mapping from the input and configuration voltages to the output current, by training them using many randomly chosen examples, generated with the physical-model based simulation tool. By solving that task for a specific nano-particle network, the trained NN becomes a differentiable model for the complex quantum-mechanic interactions within the material sample. The voltages on the configuration leads and the input electrodes are scaled to have zero mean and unit variance, and serve as inputs to the NN. Referring to the 4×4-grid of Figure 5, in Figure 9, I_1 and I_2 denote the two input leads, b denotes the back gate, and $c_1, c_2, \ldots, c_5$ denote the other leads, in a symmetric fashion, so c_1 represents V_1 and V_9, and so on, whereas O denotes the output lead. A simplified illustration of the 4×4-grid is presented in Figure 10. The training objective is to minimise the Mean Squared Error (MSE) on the (also standardised) current of the output.

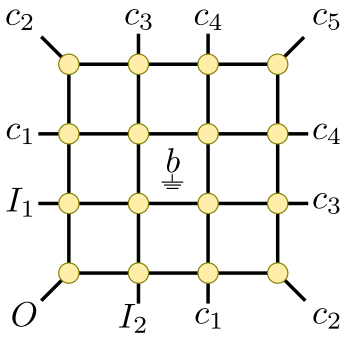


Figure 10. Simulated 4×4 nano-particle grid used for data generation

Deep feed-forward NNs have proven to be powerful function approximators, and especially recently they have been applied very successfully in a wide range of domains. They consist of a sequence of layers, where each layer computes an affine projection of its inputs followed by a pointwise non-linear function ψ:

$$\mathbf{h} = \psi(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

where $\theta = \{\mathbf{W}, \mathbf{b}\}$ are the parameters of the layer.

By stacking these layers, one can build non-linear functions of varying expressiveness that are differentiable. In theory, these networks can approximate any function to arbitrary precision given enough layers and hidden units, and they also work very well in practice. This motivates the choice to use such deep feed-forward NNs to model the input-output characteristics of the nano-particle networks that were described earlier.

Figure 11 illustrates the different phases in the use of NN-simulations to search for functionality in the nano-material, starting with the training of the NN-model from data obtained from the real material or its physical model, then querying the trained NN-model for configurations that should provide a target functionality, and finally checking whether this functionality is indeed observed in the real nano-material or physical model.

For the purpose of training, the NNs are presented with pairs of input and output combinations taken from the real material or the physical model. The inputs are passed in, propagated through the NN, and the resulting output of the NN-model

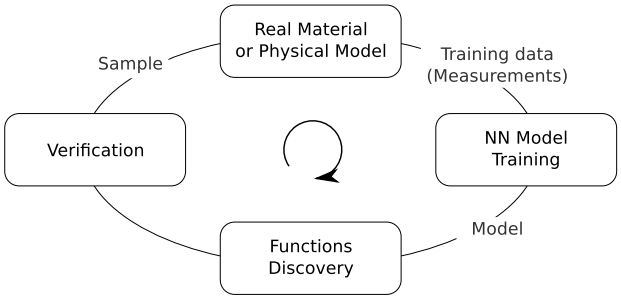


Figure 11. Closing the loop between the real nano-material or its physical model and the NN-simulation

is compared to the output of the real material or physical model. The difference between the output of the NN-model and the desired output (error) is propagated back through the NN-model in order to calculate the error gradient with respect to the particular network parameters θ. The full training is done by Stochastic Gradient Descend (SGD), meaning that the above is repeated over and over again while adjusting the parameters of the NN-model a little bit at each step. For efficiency reasons the gradients are calculated for batches of input and output combinations, hence a stochastic approximation of the full gradient is obtained.

Being essentially black boxes themselves, NNs do not directly facilitate a better understanding of the underlying quantum-mechanical processes, but they offer interesting possibilities to explore the computational capabilities of the nano-material. The trained NN is a differentiable approximate model of the nano-material: it maps inputs to outputs in roughly the same way as the real nano-material. This property can be used to run the model "backwards": find inputs that produce certain desired outputs by using a backpropagation algorithm to perform gradient descent, this time not on the weights but on the inputs. Here, it was required to go even further and use backpropagation to search for functions; in particular, the aim is to find settings of the configuration leads such that various combinations of the input leads (logic pairs) produce corresponding desired (logic) outputs. For this purpose, the NN must first be sufficiently well trained, which depends on data collected from the nano-material or the physical model. Afterwards, searching for arbitrary functions is very fast and can happen independently of the nano-material or the physical model.

A. Clustering and Visualisation

Some structure in the data can be difficult to recover by modelling it with an NN. To deal with these cases, we also employ two standard techniques from datamining: visualisation and clustering. Visualisation harnesses the powerful human pattern recognition abilities to discover structure that would otherwise remain hidden. Clustering, on the other hand, is an unsupervised learning method that tries to automatically partition the data into subsets (clusters) that are similar within, but dissimilar in-between clusters. The most commonly used clustering method is the K-Means method, which minimises the following function:

$$f(\mathbf{X}, \mathbf{S}, \theta) = \sum_{k=1}^{K} \sum_{\mathbf{x} \in S_k} \|\mathbf{x} - \mu_k\|^2 ,$$

where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is the data, $\mathbf{S}$ is the partitioning of the data, and $\theta = \{\mu_1, \ldots, \mu_K\}$ are the cluster centres. This minimisation is done by initially setting the cluster centres randomly and then alternating between the two steps: 1) reassigning the points to the cluster with the closest centre and 2) moving all μ_i to the centre of their cluster i.

B. Closing the Loop

The configurations of the nano-material or its physical model that were searched for in the NN-model should then be confronted back with the source of the original data – the real material or the physical model. The purpose of such experiments is to close the loop, as illustrated in Figure 11, between the real materials, their simulations and the data mining models. This serves as a verification of whether the NN-models are sufficiently suitable models for representing the properties of the real material.

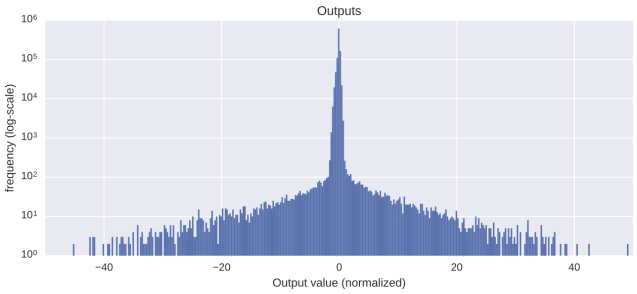
V. Results of Simulations on 4×4 Grid

In this section, the results of the NN-modelling of the 4×4 grids of Figure 10 are presented. The main subsections describe the way the data was collected and processed (Subsection V-A), and the way the NN-model was obtained and analysed (Subsection V-B). It should be emphasised again that this means we developed an alternative NN-model based on the data obtained from a physical model of a nano-particle network, not on data obtained from a real nano-particle network. In Section VI, we present similar results based on data obtained from a real nano-tube network.

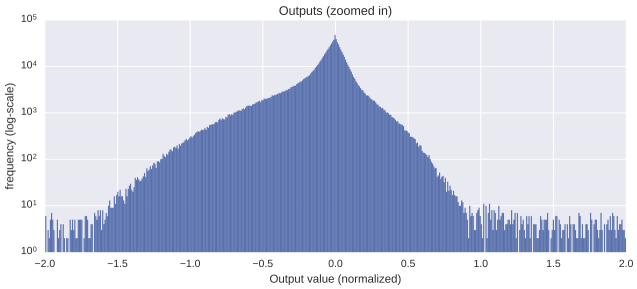
A. Data Description

The experiments that are described here are on a simulated 4×4 grid (see Figure 10). The goal is to see 1) how well we can model the nano-material with a neural network and 2) to see whether we can use a trained NN to find logic functions. The simulated material consists of 16 gold nano-particles arranged in a 4×4 grid. It has a total of 13 electrodes (12 leads on the periphery and one backgate) attached to it. Two of these leads were used as inputs (I_1, I_2), one as output (O) and the other 10 as configuration leads. The goal of the measurements was to explore the functions that the nano-grid with 2 inputs can compute while varying the voltages at the configuration leads. Only symmetric functions (with $f(a, b) = f(b, a)$) were considered. So, in order to keep things simple, the 10 configuration leads were restricted to be symmetric, thus leading to 6 degrees of freedom for configuring the function. The nano-material was formally treated as a function $f_\theta(I_1, I_2)$

parameterised by $\theta = \{c_1, c_2, c_3, c_4, c_5, b\}$. To model the whole space of functions it can embody, a deep feed-forward NN was trained in order to predict the output current from the two inputs and 6 configuration voltages. Roughly one million samples were collected from simulations, where each sample corresponds to a run with these 8 voltages set at random, and one resulting output current that has been reached after a settling period. Figure 12 shows the distribution of the target samples in the collected data.



(a)



(b)

Figure 12. Initial distribution (a) and zooming in (b) of the training targets as contained in the data generated from the simulation

Figure 12(a) shows the initial distribution of the training targets as contained in the data generated from the simulation. It is clear that the data to be modelled forms a narrow peak around zero. The tails contain only widely distributed values close to zero. Zooming in the data as illustrated in Figure 12(b) provides a better insight into the interesting simulation outputs. From the data samples collected, 90% of these samples were used for training the NN-model and the rest for validation. For completeness, the visualisation of the parameter sweep across the two inputs is depicted in Figure 13.

The scatter plot of Figure 13 shows the outputs as a function of the varying input voltages, while the remaining leads are supplied with random voltages.

B. Analysis

The voltages on the configuration leads and the inputs are scaled to have zero mean and unit variance and serve as inputs to the NN-network, as depicted in Figure 9. The training objective is to minimise the mean squared error (MSE) on the standardised output voltage. The optimisation is done

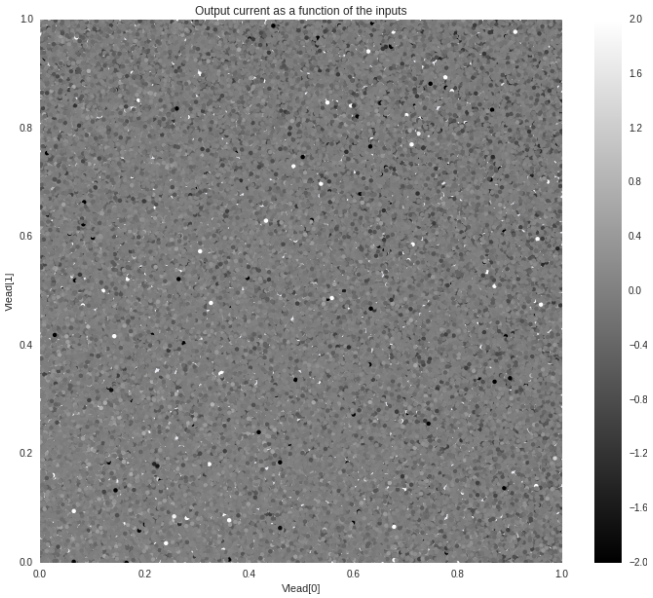


Figure 13. Scatter plot of the outputs

using minibatch stochastic gradient descent with Nesterov-style momentum. Training is stopped once the MSE on the validation set did not decrease for 5 epochs, or after a maximum of 100 epochs.

The hyperparameters of the networks were optimised by random search over 40 runs [27]–[29]. The following parameters were sampled at random:

- learning rate η log-uniform from $[10^{-4}, 10^{-1}]$
- number of hidden layers from $\{1, 2, 5, 10\}$
- number of units in each layer from $\{8, 16, 32, 64, 128\}$
- activation function from {ReLU, tanh, sigmoid}.

The best performing network had 2 hidden layers with 128 rectified linear units each and was trained with a learning rate of $\eta \approx 1.6 \cdot 10^{-2}$. This is the network we use for the rest of the analysis.

1) Using the Model: The trained neural network model was used to find the logic functions by backpropagation of the error all the way to the inputs, i.e., following the error gradient for the inputs performs the gradient descent towards the input configurations that represent the example logic functions.

The dataset contains samples (input vectors) that have different values for the input leads, but share the same (random) values for the configuration leads along with the desired output values (perceptual aliasing). Gradient descent is then used to minimise the mean square error by adjusting the values for the configuration leads, but their values are kept the same for all examples. Formally, given the NN-model $\hat{f}$ of the nano-material, we define an error over our N input/output pairs

$((I_1^{(i)}, I_2^{(i)}, O^{(i)}))$:

$$E = \sum_{i=1}^{N} \frac{1}{2}(\hat{f}(I_1^{(i)}, I_2^{(i)}, \theta) - O^{(i)})^2$$

The gradient is then calculated using backpropagation:

$$\frac{\partial E}{\partial \theta} = (\hat{f}(I_1^{(i)}, I_2^{(i)}, \theta) - O^{(i)}) \frac{\partial \hat{f}}{\partial \theta}$$

2) Global Optimisation: One problem with the method described above, is that it only performs a local search, which means that the solution it converges to might correspond to a bad local minimum (unlike when using evolutionary methods to configure the materials or the material models). In order to mitigate this problem, it was decided to first sample 10,000 random starting points (settings of the configuration leads), and perform just 10 iterations of the described local search on them. Only the starting point that leads to the lowest error is then optimised further for another 5,000 epochs, in order to obtain the final solution. In this way, we reduce the risk of getting stuck in a poor local minimum. Each search comprises 420K evaluations of the neural network, for a total of about a minute on a modern CPU.

3) Removing Outliers: After training several NNs on the simulation data, we discovered that more than half of the total prediction error stems from less than 0.1% of the data. These samples which account for most of the error also turn out to be outliers in terms of their output values, as can be seen in Figure 14.

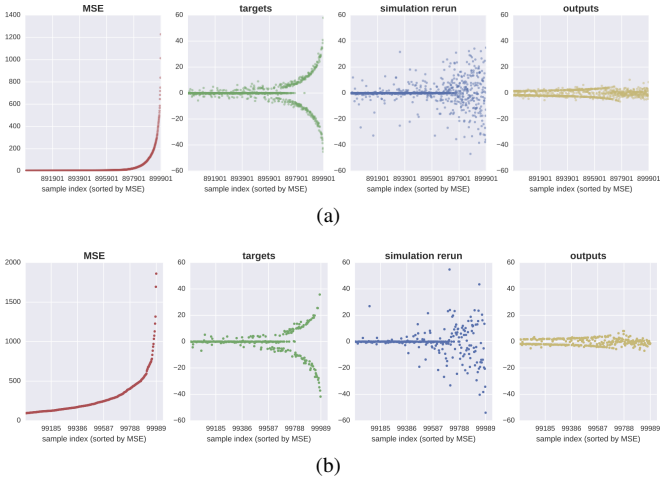


Figure 14. (a) Training and (b) validation errors of the worst 10% of the data

Figure 14 shows in (a) the training and in (b) the validation errors of the worst 10% of the data. We can see that as the mean square error (between the targets and outputs) goes up, it becomes hard to predict the data. After re-running the simulation, the targets become different due to the stochastic behaviour of the simulation, which is confirmed by non-predictability of such data using the NN-model.

Figure 15 provides more insight into the distribution of the most and least predictable data fragments.

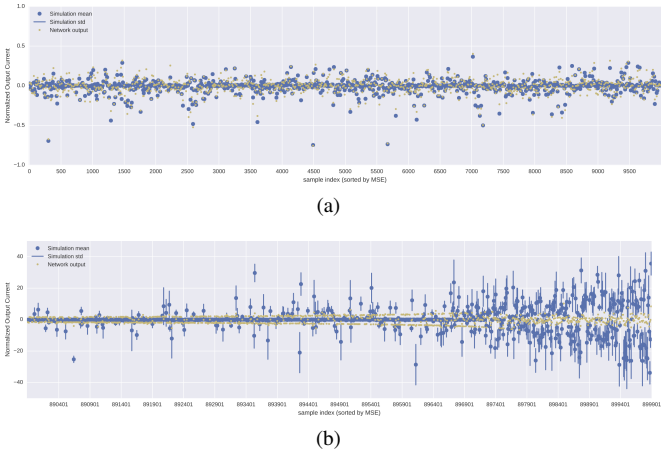


Figure 15. Most and least predictable data

The plots of Figure 15 show in (a) the first 10 k and in (b) the last 10 k data sorted according to the MSE. The simulation was executed multiple times. Vertical lines at each position depicts the distribution of the outputs for the given input configuration (the mean and the standard deviation). We can see that as the distributions get wider, the NN-model tends to have a harder time to predict the simulation outputs. The gradient for training the NN-network is thus dominated by the few unpredictable samples, leading the model to ignore the bulk of the data. For these reasons, it was decided to remove those outliers entirely from the dataset. After that the NN-network performed much better in modelling the material, as can be seen from Figure 16.

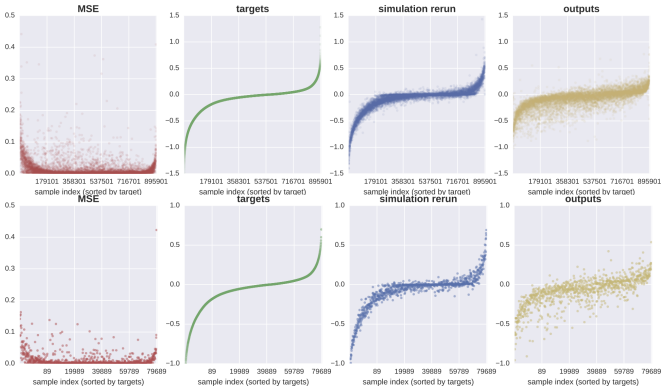


Figure 16. Experimental results after cleaning the data

Figure 16 shows the experimental results after cleaning the data. The 0.05% of the data (which covers the unpredictable subset) was removed in (a) from the training sets, and in (b) from the validation sets. We can see that re-running the simulations provides similar targets, hence the NN has no longer problems with predicting the simulation output. All results from here on have been obtained from the "cleaned" data.

C. Logic Gates Search by Gradient Descent

The overall aim was to find configurations for the simulated 4×4 nano-grid that turns it into a multi-functional reconfigurable device for computing some well-known Boolean logic functions, just like the physical-model based simulation did: AND, OR, XOR, NAND, NOR, XNOR. For this goal, we first had to decide which values of (x_0, x_1) to map (False, True) to. The obvious choice is $(0, 1)$, but values of $(0.2, 0.8)$, for example, would also still be acceptable. To circumvent the problem of choosing these values, the same gradient descent method was used to adjust the values for True and False for the inputs as well.

In particular, the following was done for each function:

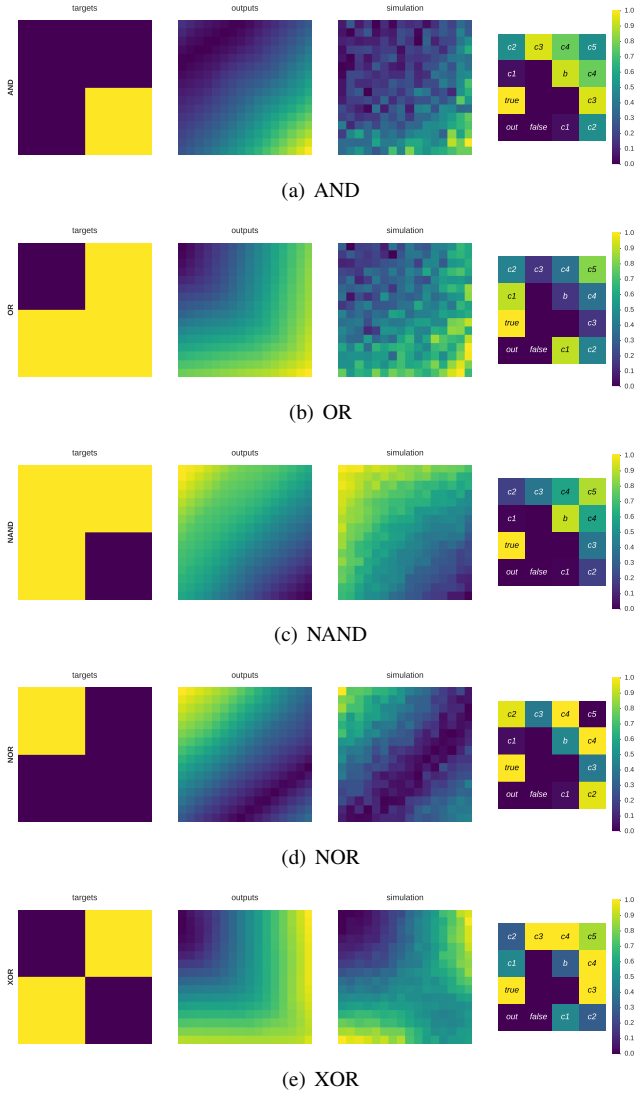
1) generate eight random numbers, while assuring that $x_1 > x_0$
2) using these values, create a set of four input/output pairs (see Table I)
3) perform gradient descent on these 8 values, while maintaining $x_1 > x_0$

The scheme by which the four input/output pairs are created from the eight random values $x_0, x_1, \ldots, x_7$ for any of the logic functions, is explained in Table I, in this case for the logic OR. Note that this depends on $x_0 < x_1$.

TABLE I. THE SCHEME FOR CREATING THE FOUR INPUT/OUTPUT PAIRS (FOR THE LOGIC OR)

I_1	I_2	c_1	c_2	c_3	c_4	c_5	b	out
x_0	x_0	x_2	x_3	x_4	x_5	x_6	x_7	OR(F, F)
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	OR(F, T)
x_1	x_0	x_2	x_3	x_4	x_5	x_6	x_7	OR(T, F)
x_1	x_1	x_2	x_3	x_4	x_5	x_6	x_7	OR(T, T)

First, a global search for a good start point was performed and then the resulting vector was optimised further. The results can be seen in Figure 17. There, the resulting configurations for five logic functions are illustrated (one logic function per row). The leftmost column shows the desired output for the logic function, while the second column presents the actual response of the trained NN-model. In the rightmost column, the corresponding configurations are visualised. Using these back in the simulations based on the physical model, we obtain the outputs as presented in the third column.



(a) AND

(b) OR

(c) NAND

(d) NOR

(e) XOR

Figure 17. Gates found using the NN-model, with configuration voltages (rightmost column), NN-model output (second column), desired logic (leftmost column), and the regenerated outputs of the physical material model (third column)

D. Results

From the plots in Figure 17, it can be seen that most of the target Boolean functions can be performed by the nano-material (according to the NN-model). Note that, what is most important is the response of the model in the corners, since we do not really care about values in between True and False. The smooth plots are just there to show how the model is behaving, and to give an impression on how robust the solutions are. For AND, OR, NAND, NOR and XOR, the values in the corners match the desired outputs quite well. In contrast, the search for the XNOR function failed to produce equally satisfactory results, indicating that this function might be difficult to perform for the nano-material under the given setup.

E. Verification in the Physical Model

The configurations found to represent the gates in the NN-model were then used to simulate the physical material model in order to see how well the model represents the original material model. The third column in Figure 17 shows the responses of the physical material model. Comparing these visualisations to the second column in Figure 17, one can see that the configurations found using the NN-model were successfully used to configure the original physical Monte Carlo-based model, and they produce similar outputs. We can observe noise, superimposed on the physical simulation outputs caused by the stochasticity of the simulation, whereas the NN-outputs are smooth because the NN is a deterministic model. The main

result is that, even including the noise, the measurements of the physical model constitute the same logic functions as those found in the NN-model. To conclude, the NN-model can be trained to simulate the physical model, and the configurations found using the NN-model can be used to produce the same desired behaviour of the physical material model.

VI. Results of Simulations on SWCNT-sample

Very recently, we conducted similar experiments with real material samples composed of thin films of composites of single-walled carbon nano-tubes (SWCNTs). Instead of depositing nano-particles on an electrode array, the material deposited was a mixture of SWCNTs randomly mixed in an insulating material. The insulating material was either PMMA/PBMA (Polymethy/butyl methacralate) [30]. SWCNTs are mixed with PMMA or PBMA and dissolved in anisole (methoxybenzene). About 20 μL of material is then drop-dispensed onto the electrode array. This is dried at 100°C for 30 min to leave a film over the electrodes. SWCNTs are conducting or semi-conducting and the role of the PMMA/PBMA is to introduce insulating regions within the nano-tube network, to create non-linear current versus voltage characteristics.

Once these nano-tube networks have been fabricated, the approach and methodology are exactly the same as already described in earlier sections, except that we did not use a physical model for gathering training date: First, input-output data combinations were collected by more or less random measurements on the real material. Then, these combinations were used to train an NN-model. Finally, the NN-model was used to predict suitable settings of the configuration variables for the material to act as a Boolean logic gate, and these settings were validated by trying them on the real material. We complete this paper by giving a short description of one of the experiments and its results for one particular material sample.

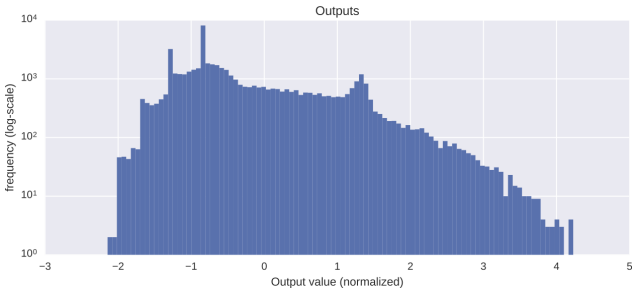


Figure 18. Distribution of the training targets obtained from the SWCNT-sample

In this experiment, roughly 55 k combinations of input-output date were collected from the SWCNT-sample. Two out of a total of eight electrodes were used as the two inputs for the logic function, one was used to read out an output, and the remaining six were used to configure the material. The output was sampled at 100 kHz, and averaged and rescaled between 0 and 1. The distribution of the output values is depicted in Figure 18.

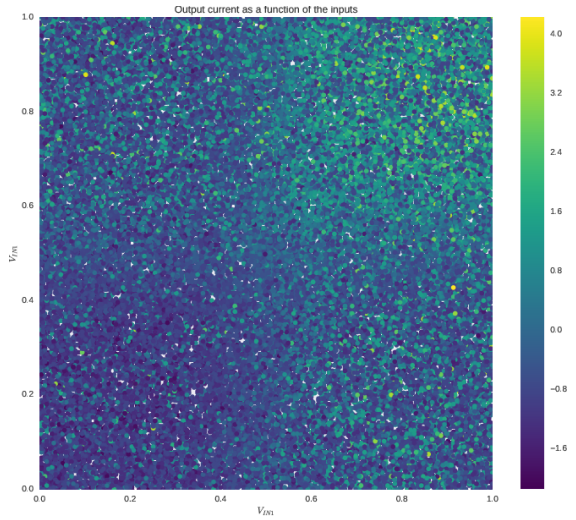


Figure 19. Scatter plot of a sweep across the two selected input electrodes

The scatter plot in Figure 19 shows a sweep across the two input electrodes while configuring the other six configuration electrodes with random voltages. One can clearly observe the variability of outputs provided by the SWCNT-sample.

A new NN-model with more hidden layers was used for the simulations. It consisted of a feed-forward network with 5 hidden layers, each layer having 120 units; the applied output function was ELU [31]. This NN-model was trained with the ADAM [32] optimiser for 50 epochs. Figure 20 shows how well the NN-model was able to match the training data.

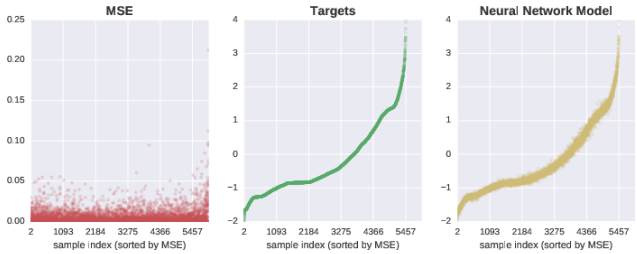
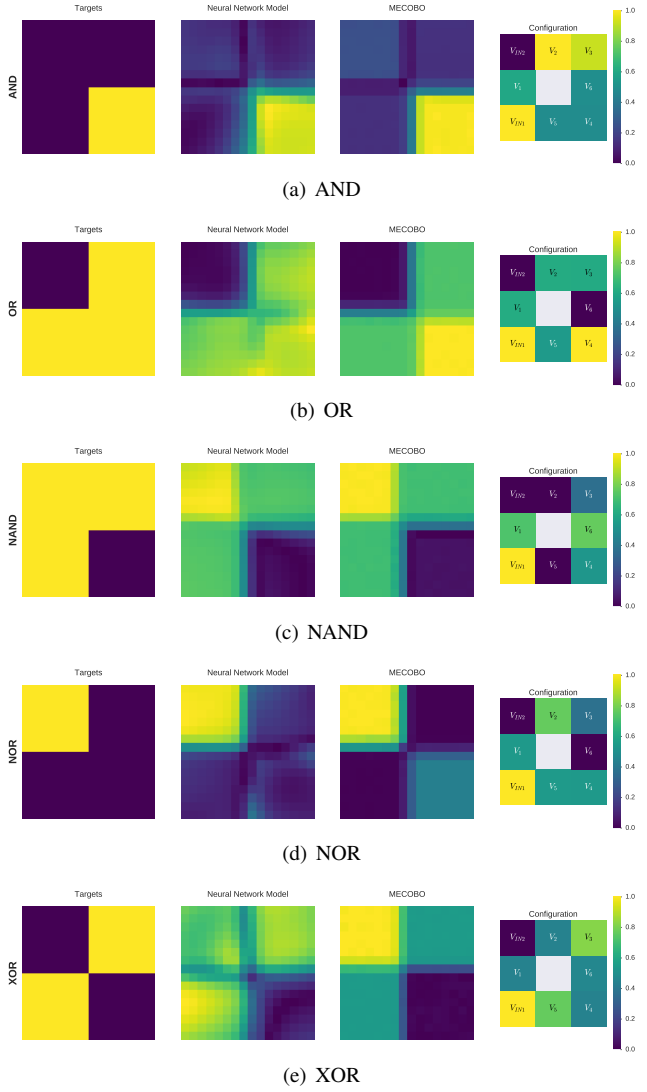


Figure 20. NN-model outputs compared to the outputs of the SWCNT-sample on the validation set

As in the nano-particle network example, suitable configuration voltages for different logic gates were searched in the trained NN-model, using gradient descent. The found configurations were then verified on the same SWCNT-sample that was used for the training data collection, hence closing the modelling loop. The results are visually summarised in Figure 21.

As in our earlier nano-particle network example, the resulting configurations for five logic functions are illustrated (one logic function per row). The leftmost column shows the desired output for the logic function, while the second column presents the actual response of the trained NN-model. In the rightmost



(a) AND

(b) OR

(c) NAND

(d) NOR

(e) XOR

Figure 21. Gates found using the NN-model, with configuration voltages (rightmost column), NN-model output (second column), desired logic (leftmost column), and the regenerated outputs of the SWCNT-sample (third column)

column, the corresponding configurations are visualised. Using these back on the SWCNT-sample, we obtained the outputs as presented in the third column. Except for the XOR function, we see quite a good match between the predicted configurations from the NN-model and the actual performance of the SWCNT-sample provided with these configurations.

VII. Conclusion

This paper has demonstrated how an artificial Neural Network model can be applied to look for configuration voltage settings that enable different standard Boolean logic functions in the same piece of material consisting of a disordered nano-particle or nano-tube network. The training of the Neural Network was based on generated random data from a physical-model based simulation tool in case of the nano-particle networks, and on real data in case of the nano-tube networks. The results are promising and can inform the electrical engineers about possible functional capabilities of these material systems, without the need of fabricating and doing costly and time-consuming trial-and-error experiments on real nano-material networks. Of course, it is obvious that such experiments are unavoidable if it comes to actually testing real networks for the predicted functionalities. In fact, the capability of reconfigurable Boolean logic in small samples of nano-material networks has meanwhile been confirmed experimentally. It is likely, that this proof of concept will be the starting point for exciting new research, and open up the opportunity for a totally new approach to developing multi-functional stand-alone devices.

Next steps in this direction first of all involve the simulation of more real material samples, especially for larger nano-material networks and for more complex functions. For this, also new experiments are needed, in order to produce and collect sufficiently many input-output combinations to enable proper training of the Neural Network. This also requires new fabrication techniques, involving larger nano-material networks on micro-electrode arrays with more contact electrodes interfacing with the material, and with a more sophisticated back gate. The requirements can be predicted by simulations, in particular if one wants to turn to more complicated functionalities, like computational tasks that are difficult to perform with digital computers. Secondly, it would be worthwhile to apply the same modelling and simulation approach to other materials that show interesting physical properties and behaviour, like networks of quantum dots, sheets of graphene, and mixtures of such materials. Note that the same approach was very recently applied to samples of biological material consisting of slime moulds for discovering Boolean gates [33]. Thirdly, a natural next step would be to integrate the Neural Network modelling approach with the evolutionary search technique. These are amongst the future research plans we want to pursue.

Acknowledgment

We acknowledge financial support from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 317662. We also thank the other partners in the NASCENCE project for producing the nano-material samples, and for their help in collecting the data.

References

[1] K. Greff, R. M. J. van Damme, J. Koutník, H. J. Broersma, J. Mikhal, C. P. Lawrence, J. Schmidhuber, and W. G. van der Wiel. Unconventional computing using evolution-in-nanomaterio: Neural networks meet nanoparticle networks. *FUTURE COMPUTING 2016*, pages 15–20, 2016.

[2] Hajo Broersma, Faustino Gomez, Julian Miller, Mike Petty, and Gunnar Tufte. Nascence project: Nanoscale engineering for novel computation using evolution. *International Journal of Unconventional Computing*, 8(4):313–317, 2012.

[3] H. J. 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: Prototypes, Models and Algorithms*, pages 397–428. 2016.

[4] S. K. Bose *et al.* Evolution of a designless nanoparticle network into reconfigurable boolean logic. *Nature Nanotechnology*, 10(12):1048–1052, 2015.

[5] M. K. Massey *et al.* Evolution of Electronic Circuits using Carbon Nanotube Composites. *Scientific Reports*, 6, 2016. Article number 32197.

[6] R. M. J. van Damme, H. J. Broersma, J. Mikhal, C. P. Lawrence, and W. G. van der Wiel. A simulation tool for evolving functionalities in disordered nanoparticle networks. *EPSyM, IEEE WCCI 2016*, 2016.

[7] L.W. Nagel and D.O. Pederson. Simulation program with integrated circuit emphasis. Memorandum ERL-M382, University of California, Berkeley, April 1973.

[8] Christoph Wasshuber, Hans Kosina, and Siegfried Selberherr. A simulator for single-electron tunnel devices and circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16:937–944, 1997.

[9] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 7:115–133, 1943.

[10] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[11] P. J. Werbos. Applications of advances in nonlinear sensitivity analysis. In *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*, pages 762–770, 1981.

[12] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. A committee of neural networks for traffic sign classification. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1918–1921, 2011.

[13] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2012*, 2012. Long preprint arXiv:1202.2745v1 [cs.CV].

[14] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014. [retrieved: Feb, 2016].

[15] A.N. Korotkov. *Coulomb Blockade and Digital Single-Electron Devices*, pages 157–189. Blackwell, Oxford, 1997.

[16] Christoph Wasshuber. *Computational Single-Electronics*. Springer-Verlag, 2001.

[17] Christoph Wasshuber. Single-Electronics – How it works. How it's used. How it's simulated. In *Proceedings of the International Symposium on Quality Electronic Design*, pages 502–507, 2012.

[18] Julian F Miller and Keith Downing. Evolution in materio: Looking beyond the silicon box. *Proceedings of NASA/DoD Evolvable Hardware Workshop*, pages 167–176, July 2002.

[19] Simon Harding and Julian Francis Miller. Evolution in materio. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 3220–3233. Springer, 2009.

[20] Simon L. Harding and Julian F. Miller. Evolution in materio: Evolving logic gates in liquid crystal. *International Journal of Unconventional Computing*, 3(4):243–257, 2007.

[21] Simon L. Harding, Julian F. Miller, and Edward A. Rietman. Evolution in materio: Exploiting the physics of materials for computation. *International Journal of Unconventional Computing*, 4(2):155–194, 2008.

[22] Julian F. Miller, Simon L. Harding, and Gunnar Tufte. Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*, 7:49–67, 2014.

[23] A. Yoshihito. Information processing using intelligent materials - information-processing architectures for material processors. *Intelligent Materials Systems and Structures*, 5:418–423, 1994.

[24] Klaus-Peter Zauner. From prescriptive programming of solid-state devices to orchestrated self-organisation of informed matter. In Jean-Pierre Banâtre, Pascal Fradet, Jean-Louis Giavitto, and Olivier Michel, editors, *Unconventional Programming Paradigms: International Workshop UPP 2004*, volume 3566, pages 47–55. Springer, 2004.

[25] P. Cariani. To evolve an ear: epistemological implications of Gordon Pask's electrochemical devices. *Systems Research*, 3:19–33, 1993.

[26] G. Pask. Physical analogues to the growth of a concept. In *Mechanisation of Thought Processes*, number 10 in National Physical Laboratory Symposium, pages 877–922. Her Majesty's Stationery Office, London, UK, 1958.

[27] R. L. Anderson. Recent advances in finding best operating conditions. *Journal of the American Statistical Association*, 48(264):789–798, December 1953.

[28] Francisco J. Solis and Roger J.-B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, February 1981.

[29] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[30] M. Mohid, J.F. Miller, S.L. Harding, G. Tufte, M.K. Massey, and M.C Petty. Evolution-in-materio: Solving computational problems using carbon nanotube-polymer composites. *Soft Computing*, xx:xx–xx, 2016. In press.

[31] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[32] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] S. Harding, J. Koutník, K. Greff, J. Schmidhuber, and A. Adamatzky. Discovering boolean gates in slime mould. *ArXiv paper, submitted*, 2016. arXiv:1607.02168 [retrieved: November, 2016].