Unconventional Computing Using Evolution-in-Nanomaterio: Neural Networks meet Nanoparticle 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 and MESA+ Institutes for ICT Research and Nanotechnology
University of Twente, The Netherlands
Email: {r.m.j.vandamme,h.j.broersma,c.p.lawrence,w.g.vanderwiel}@utwente.nl

***Abstract*—Recently published experimental work on evolution-in-materio applied to nanoscale materials shows promising results for future reconfigurable devices. These experiments were performed on disordered nano-particle networks that have no predefined design. The material has been treated as a black-box, and genetic algorithms have been used to find appropriate configuration voltages to enable the target functionality. In order to support future experiments, we developed simulation tools for predicting candidate functionalities. One of these tools is based on a physical model, but the one we introduce in this paper is based on an artificial neural network. 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. The experiments we report here demonstrate that the neural network can model the simulated nano-material quite accurately. The differentiable, neural network-based material model is then 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 materials. 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–nanoparticle network; neural network; simulation; unconventional computation; evolution-in-nanomaterio.

I. Introduction and Motivation

Within the EU-project NASCENCE [4], disordered networks of gold nano-particles have been successfully used to produce reconfigurable logic with a very high degree of stability and reproducibility [3]. This breakthrough presents a proof of principle that indicates very promising prospects for using this material to perform more complicated computational tasks. In order to predict candidate computational tasks, but avoiding the waste of scarce and expensive resources involved in experimentally exploring these nano-particle networks, we developed simulation tools for examining the capabilities of these material systems. One of them [7] is an extension of existing tools for simulating nano-particle interactions, like SPICE [11] or SIMON [18]. The latter tools are all based on Monte-Carlo simulations, using a physical model, and have been validated for designed systems with small numbers of particles. Although these methods can, in principle, handle arbitrary systems of any size, their scalability is a serious issue. Moreover, nano-particle networks like the ones used in [3] to date cannot be produced according to a predefined specific design. Therefore, due to their disordered nature, it is not possible to use an accurate physical model for such material systems.

In this paper, an alternative approach is introduced. This novel approach is based on training artificial Neural Networks in order to model and investigate the nano-particle networks. Neural Networks (NN; [10] [13] [19]) have proven to be powerful function approximators and have, especially recently, been applied to a wide variety of domains with great success [5] [6] [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 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 material computes different Boolean logic functions, such as AND, OR, NOR, NAND, and XOR.

To enable the exploration of the computational capabilities of the material by an NN, the NN needs to be trained first with data collected from the measurements on the material. Since a physical model and an associated validated simulation tool for the nano-particle networks have already been developed [7], such training data can be obtained from the simulated 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 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 [3]. This section also describes the choices that have been made and that form the basis for the physical-model based simulation tool of [7], 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, followed by a short section with conclusions as

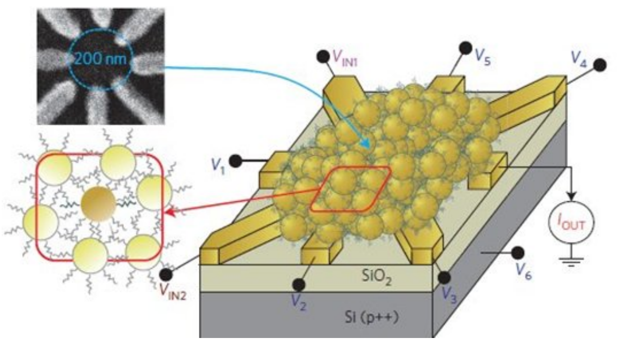

Figure 1. Illustration of a disordered network of gold NPs

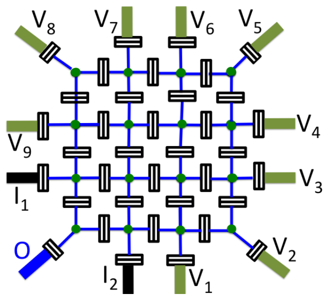

Figure 2. A symmetric 4×4-grid of 16 nano-particles with leads

well as an outlook to future work.

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 1 shows an illustration of such a network. The central circular region is about 200 nm in diameter. More details on the production process can be found in [3].

The networks investigated there are relatively large (in the order of a hundred particles) and disordered. The transport of electrons is governed by the Coulomb blockade effect [8]: 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 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. More details on the simulation tool can be found in [7].

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 2.

In Figure 2, 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. We omit the details due to page restrictions. Details can be found in [7].

1) Evolved Boolean Logic: Applying the developed simulation tool to the small network of Figure 2, 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 3. 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.

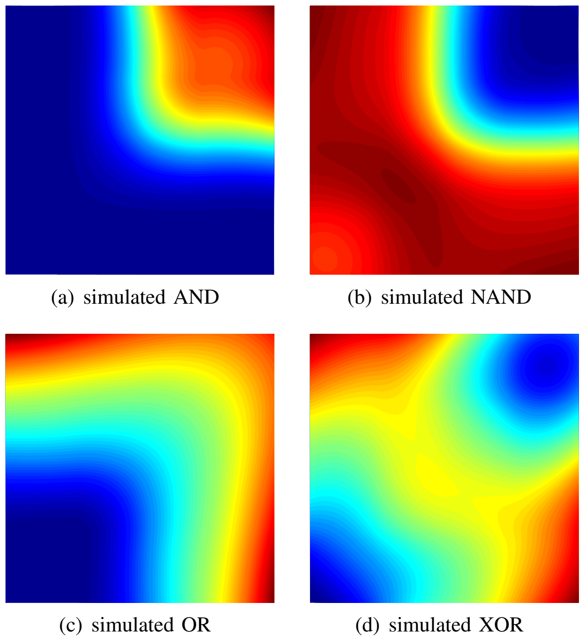

Figure 3. Contour plots of simulated evolved logic in the 4×4-grid

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 2 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 [3], 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. 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 2.

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 [7] visualisation tools have been developed to analyse the processes that are taking place over time. In Figure 4, 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).

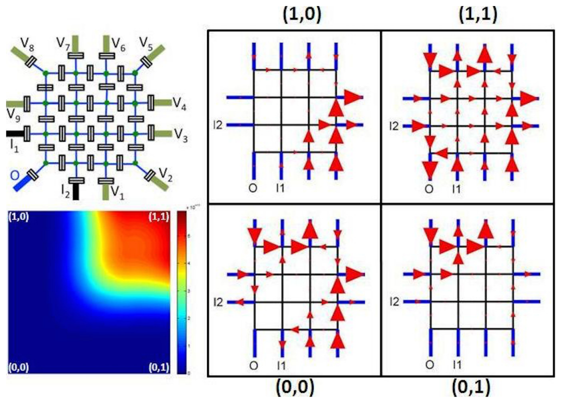

Figure 4. Averaged current patterns for simulated AND in the 4×4-grid

Figure 4 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.

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 4×4-grid.

IV. Neural Networks

Deep feed-forward NNs are powerful function approximators, and recently they have been very successfully applied 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 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. These NNs are 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.

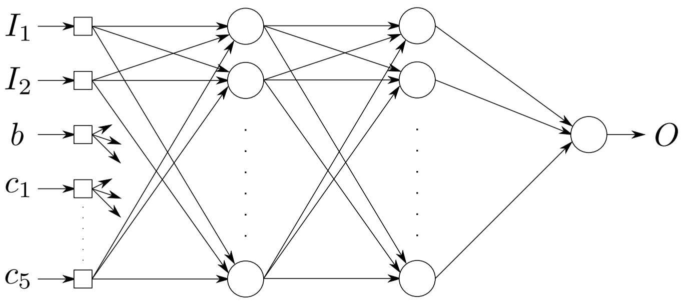

Figure 5. Illustration of an NN with two hidden layers

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. Referring to the 4×4-grid of Figure 2, in Figure 5, 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. The training objective is to minimise the Mean Squared Error (MSE) on the (also standardised) current of the output. The optimisation is done using Minibatch Stochastic Gradient Descent with Nesterov-style momentum [9] [12]. The Minibatch Stochastic Gradient Descent method is the de facto standard for optimising deep feed-forward NNs, whereas Nesterov-style momentum is a regularly used extension for improving convergence speed. There are other more elaborate optimisation schemes (like AdaGrad, RMSProp, ADAM, etc.), but their benefits are usually moderate, and there was no need for this added complexity here.

The choice of network architecture and training parameters comprise a set of hyperparameters that need to be set for this method. For the network, these are the number of hidden layers, the number of neurons in each layer and their activation functions. For training, they are the batch size, learning rate, the momentum coefficient, and stopping criterion. We fixed the batch size to 20, the momentum to 0.9, and stopped the training once the MSE on the validation set did not decrease for 5 epochs, or after a maximum of 100 epochs.

The hyperparameters of the NNs were optimised by random search over 40 runs [1] [15] [2]. In particular, we randomly sampled:

- 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 $\{\text{ReL}, \tanh, \text{sigmoid}\}$.

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

A. *Search*

The trained NN is a differentiable (approximate) model of the nano-material. This property can be used to run the model "backwards": find inputs that produce certain desired outputs by using the backpropagation algorithm to perform gradient descent, 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.

1) Local Search: To accomplish the above goal, it was necessary to produce a set of examples that have different values for the input leads but share the same (random) values for the configuration leads, along with the desired output values. Gradient descent is then used to minimise the MSE by adjusting the values for the configuration leads, while keeping their values the same for all examples. So formally, given our neural network model $\hat{f}$ of the nano-material, we define an error over our N input/output pairs $((I_1^{(i)}, I_2^{(i)}, O^{(i)}))$:

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

and then, using backpropagation, we effectively calculate:

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

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. To mitigate this, 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 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.

V. Results

In this section, we present the results of a network trained on 1M random function-evaluations collected from the simulated material. First of all, the following issue that was encountered and is illustrated in Figure 6 had to be resolved.

In Figure 6, only the 2,000 samples with the highest prediction error are shown. It can clearly be seen that the last ≈ 500 samples account for most of the total error. Also their target values are obviously outliers.

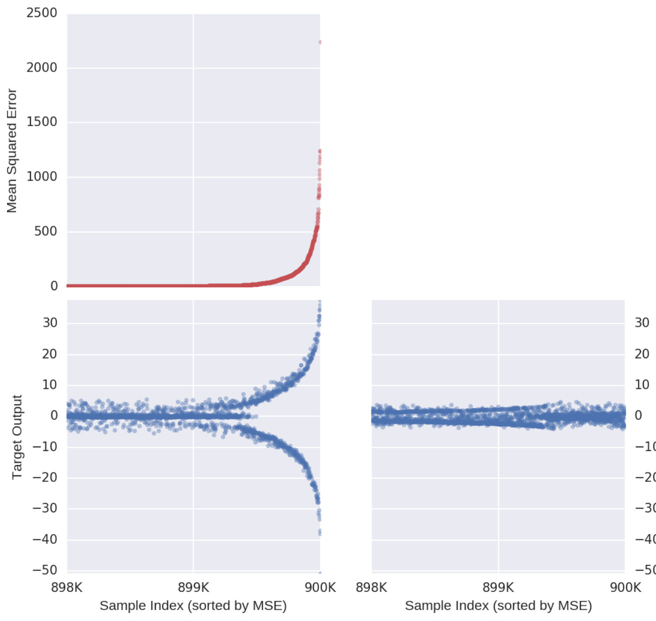

Figure 6. Prediction errors (top left), target outputs (bottom left), and network predictions (bottom right) for the simulated 4×4 nano-grid data

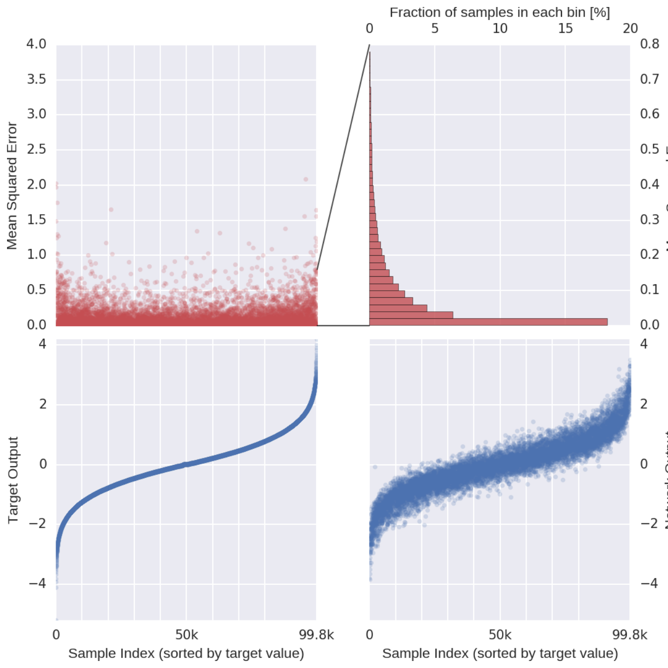

Figure 7. Prediction errors plus histogram (top), target outputs (bottom left), and network predictions (bottom right) for the *cleaned* nano-grid data

A. *Removing Outliers*

After training several NNs on the simulation data, it was 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 turned out to be outliers in terms of their output values, as can be concluded from Figure 6. Most of the data falls in the region $[-3, 3]$, but these 'bad' samples go up to ± 40. The gradient for training the NN is thus dominated by these few samples, causing 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 performed much better in modelling the material, as can be seen in Figure 7. There, the samples are sorted by target value. One can observe a clear correspondence between the predictions (bottom right) and the targets (bottom left). The histogram in the top right confirms that most of the predictive errors are very small (< 0.3).

So, as a consequence, all results in the sequel have been obtained from the 'cleaned' data.

B. Logic Gates

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)

As described in Section IV-A2, first a global search is performed, in order to find a good starting point, and then that vector is optimised further. The results are presented in Figure 8. There, the resulting configurations for six logic functions are illustrated (one logic function per row). The leftmost column shows the desired output for the logic function, while the middle column presents the actual response of the trained NN model. In the rightmost column, the corresponding configurations are visualised.

From the plots in Figure 8, it can be concluded 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, and NOR, the values in the corners match the desired outputs very well. In contrast, the search

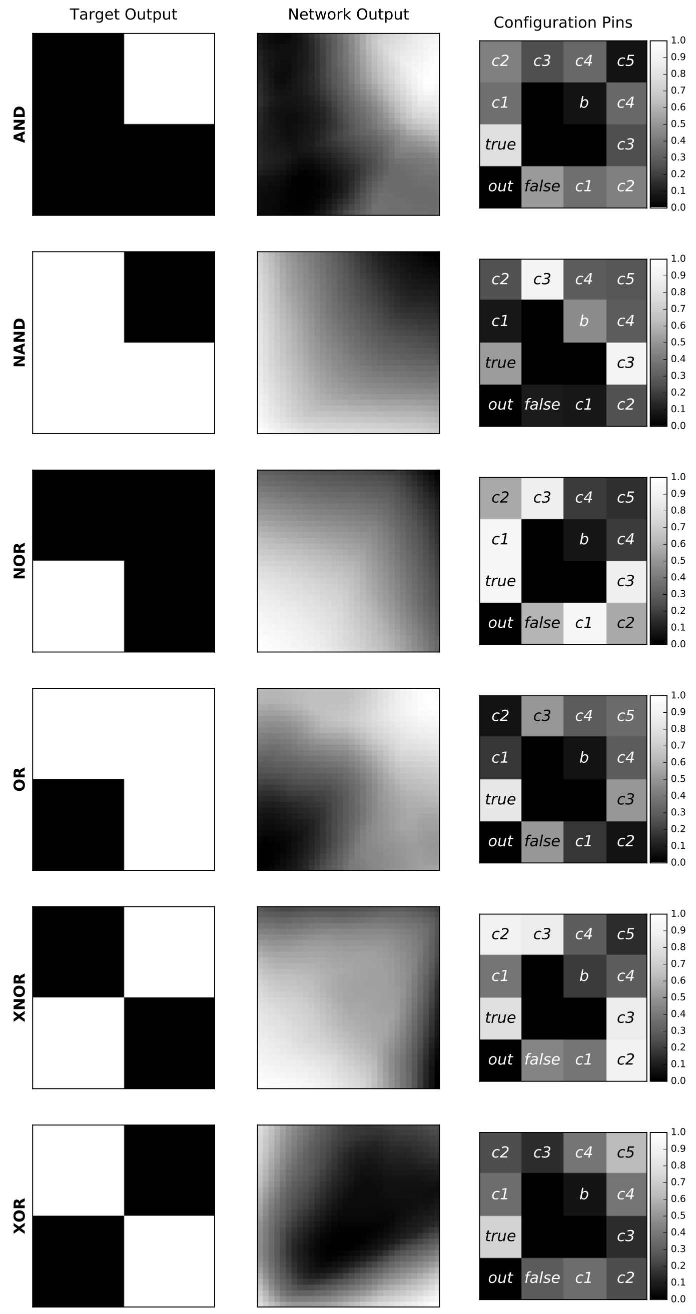

Figure 8. Results for six logic functions: desired outputs, actual responses, and the corresponding configuration

for the XOR and XNOR functions failed to produce equally satisfactory results, indicating that these functions might be difficult to perform for the nano-material under the given setup.

VI. 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 network. The training of the Neural Network was based on generated random data from a physical-model based simulation tool. 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-particle 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-particle networks has recently 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 real material samples. For this, new experiments are needed, in order to produce sufficiently many data to enable proper training of the Neural Network. This also requires new fabrication techniques, involving larger networks on micro-electrode arrays, with more contact leads to 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, films of carbon nanotubes, sheets of graphene, and mixtures of such materials. 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.

REFERENCES

[1] R. L. Anderson, "Recent advances in finding best operating conditions," *Journal of the American Statistical Association*, vol. 48, no. 264, pp. 789–798, Dec. 1953.

[2] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[3] S. K. Bose *et al.*, "Evolution of a designless nanoparticle network into reconfigurable boolean logic," *Nature Nanotechnology*, vol. 10, no. 12, pp. 1048–1052, 2015.

[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] 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)*, 2011, pp. 1918–1921.

[6] 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].

[7] 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," *Preprint*, 2015.

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

[9] J. Martens and I. Sutskever, "Training deep and recurrent networks with hessian-free optimization," in *Neural Networks: Tricks of the Trade*. Springer-Verlag, 2012, pp. 479–535.

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

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

[12] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)," *Doklady AN SSSR*, vol. 269, 1983.

[13] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[14] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014, [retrieved: Feb, 2016]. [Online]. Available: http://arxiv.org/abs/1402.1128

[15] F. J. Solis and R. J.-B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, Feb. 1981.

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

[17] ——, "Single-Electronics – How it works. How it's used. How it's simulated." in *Proceedings of the International Symposium on Quality Electronic Design*, 2012, pp. 502–507.

[18] C. Wasshuber, H. Kosina, and S. Selberherr, "A simulator for single-electron tunnel devices and circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 937–944, 1997.

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